Module 13

Hacking Web Servers

Learning Objectives

01 Summarize Web Server Concepts

02 Demonstrate Different Web Server Attacks

03 Explain Web Server Attack Methodology

04 Explain Web Server Attack Countermeasures

Objective 01

Summarize Web Server Concepts

Web Server Security Issues

Network and **OS level attacks** can be well defended using proper network security measures such as firewalls, IDS, etc. However, web servers can be accessed from anywhere via the Internet, which renders them **highly vulnerable** to attacks

Why are Web Servers Compromised?

- **Improper** file and directory **permissions**
- **Unnecessary** default, backup, or sample **files**
- **Misconfigurations** in web server, operating systems, and networks
- **Bugs** in server software, OS, and web applications
- Administrative or **debugging functions** that are **enabled** or accessible on web servers
- Use of **self-signed certificates** misconfigured SSL certificates, default certificates, and encryption settings
- Not using **dedicated server** for web services

Impact of Web Server Attacks

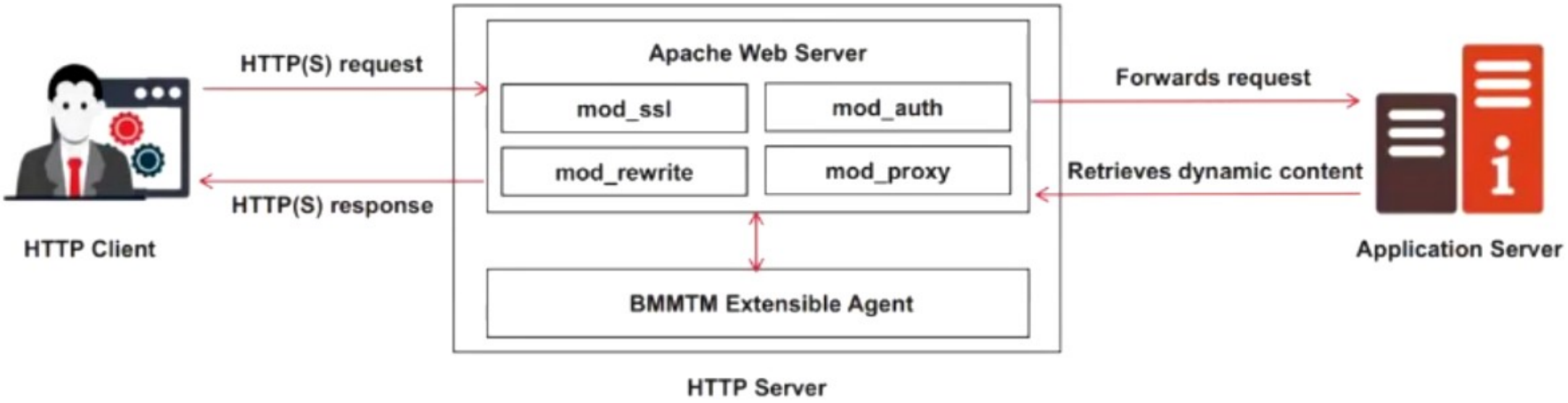
- Compromise of user accounts
- Website defacement
- Secondary attacks from the website
- Root access to other applications or servers
- Data tampering and data theft
- Reputational damage of the company

Apache Web Server Architecture

Apache web server is an **open-source** HTTP server used to **deliver web content** over the internet

Apache can **host websites**, handle HTTP requests, and serve static and dynamic content

It benefits users by providing high performance, security features, extensive customization through **modules**, and a large community for support



Apache Vulnerabilities

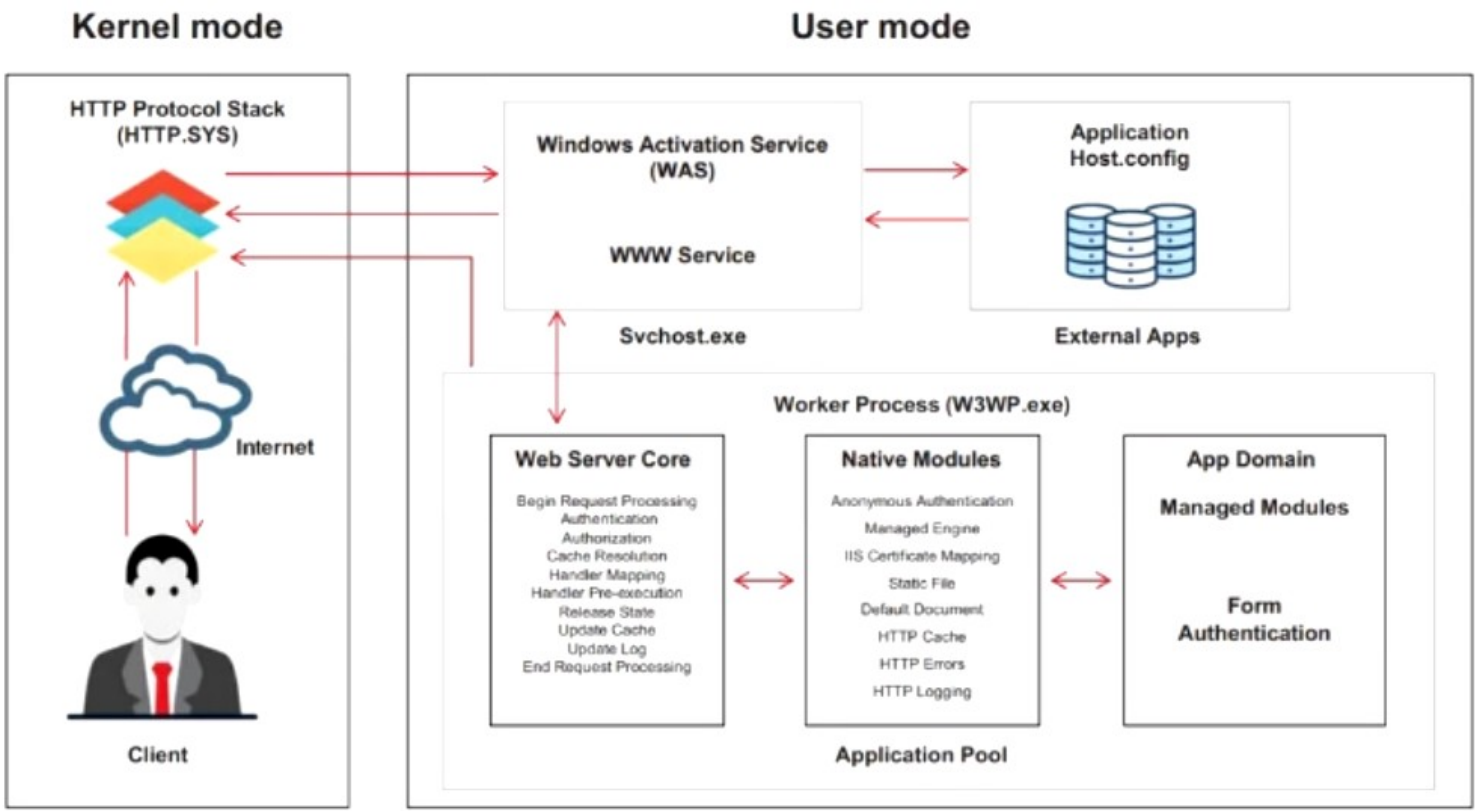
Vulnerability	Description
HTTP response splitting	This vulnerability occurs when **improperly validated input** allows attackers to **inject malicious headers into HTTP responses**, leading to XSS, cache poisoning, or sensitive information disclosure
HTTP/2 DoS by memory exhaustion on endless continuation frames	This vulnerability occurs when **attackers send continuous HTTP/2 headers**, causing excessive memory consumption and leading to server unresponsiveness or crash
mod_macro buffer over-read	This vulnerability occurs when the **mod_macro module improperly handles macro expansion**, causing **buffer over-reads** that attackers exploit with crafted requests to access sensitive adjacent memory
DoS in HTTP/2 with initial window size 0	This vulnerability arises when an attacker sets the HTTP/2 initial window size to 0, **blocking data transmission** and causing a denial of service (DoS)
HTTP/2 stream memory not reclaimed right away on RST	This vulnerability occurs when HTTP/2 stream memory is not freed on reset, allowing attackers to cause **memory exhaustion** and **denial of service** (DoS)

Vulnerability	Description
Insecure default configuration	This vulnerability arises from insecure default admin credentials, leading to **remote code execution** (RCE) when attackers use these credentials to gain **admin access**
Improper authorization	This vulnerability arises from improper authorization within the Submarine server's core components, allowing attackers to exploit **faulty checks for unauthorized access or privilege escalation**
DNS rebinding in import functionality	This vulnerability allows attackers to **manipulate DNS responses** and **access internal services** due to inadequate input validation in Apache Allura
Path traversal	Improper directory path limitations in Apache OFBiz allows attackers to **access files outside the intended directory** and potentially execute code or access sensitive data
SQL injection	This vulnerability caused by improper neutralization of SQL elements in Apache Submarine Server Core, which allows attackers to **execute arbitrary SQL queries**, leading to unauthorized access and data manipulation

https://httpd.apache.org

IIS Web Server Architecture

- Internet Information Services (IIS), a web server application developed by Microsoft, runs on a server and **responds to requests** from a browser
- IIS for Windows Server is a flexible and easy-to-manage web server for **web hosting**



IIS Vulnerabilities

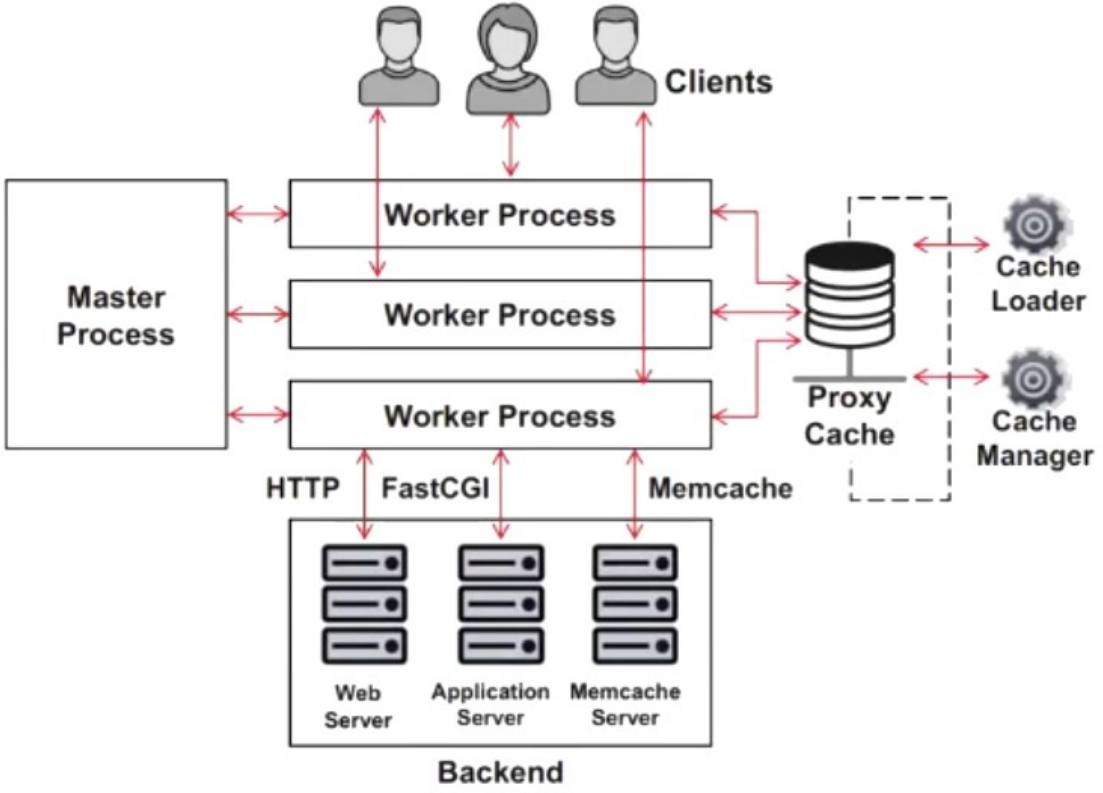
Vulnerability	Description
Trust boundary violation vulnerability	**Inadequate privilege separation** in Telerik Report Server allows unauthenticated entity to access and manipulate restricted functionalities
Authentication bypass vulnerability	An issue in the **authentication process** can allow attackers to access restricted functionality and execute arbitrary code on the server
CRLF cross-site scripting vulnerability	**Misconfigurations in the SiteMinder Web Agent** for IIS allow attackers to execute arbitrary JavaScript in a client's browser
CCURE passwords exposed to administrators	Arises due to **improper handling** and **logging** of sensitive information within the IIS Server while hosting the C•CURE 9000 Web Server
Arbitrary file path access vulnerability	**Aquaforest TIFF Server's default configuration** allows attackers to access, enumerate, or traverse directories and files, potentially bypassing authentication

Vulnerability	Description
Windows IIS server elevation of privilege vulnerability	**Improper handling of user requests** in Windows IIS server allows attackers to gain unauthorized access and control the system
File and directory permissions vulnerability	**Incorrect default permissions in Hitachi JP1/Performance Management** allows attackers to manipulate files and directories unauthorizedly
TYPO3 cross-site scripting (XSS) vulnerability	Unfiltered use of the server environment variable **PATH_INFO** in the **GeneralUtility::getIndpEnv()** allows attacker to inject malicious HTML code into uncached pages
MailEnable vulnerability	**Improper handling of file paths** allows authenticated mail users to add files with unsanitized content in public folders where the IIS user has permission to access
XSS in password manager	Improper **neutralization of user-controllable** input within the /isapi/PasswordManager.dll can allow unauthorized entity to inject malicious scripts and steal sensitive information

https://cve.mitre.org

Nginx Web Server Architecture

- Nginx is a high-performance, scalable **web server**, reverse proxy, and **load balancer** that operates on a master-worker architecture
- The architecture comprises a **master process** that oversees various worker processes responsible for handling client requests
- Each **worker process** utilizes non-blocking I/O to manage multiple connections within a single-threaded loop
- Nginx also supports advanced **load balancing**, robust caching, **SSL/TLS termination**, and detailed logging



Nginx Vulnerabilities

Vulnerability	Description
NULL pointer dereference in HTTP/3	This vulnerability occurs due to a NULL pointer dereference in **Nginx's QUIC** module, allowing attackers to launch DoS or **remote code execution**
Server-side request forgery (SSRF) vulnerability	This vulnerability in **mintplex-labs/anything-llm** allows attackers to perform port scanning, access non-public apps, and delete files
Remote code execution vulnerability	This vulnerability arises from **exposed configuration settings in Nginx-UI**, allowing attackers to perform remote code execution, privilege escalation, or information disclosure
Improper certificate validation	This vulnerability arises due to improper input validation in **Nginx-UI's Import Certificate feature**, allowing attackers to perform arbitrary file writes
SQL injection vulnerability	This vulnerability arises from **improper neutralization of SQL elements**, allowing attackers to execute arbitrary SQL queries for unauthorized access or data breaches

Vulnerability	Description
Unauthenticated private keys access	This vulnerability occurs because **Nginx does not support .htaccess files**, allowing attackers to read private keys
Excessive memory usage and CPU exhaustion in HTTP/2	This vulnerability arises from **improper memory handling** and **excessive CPU usage in HTTP/2 requests**, allowing attackers to disrupt operations by flooding the server
OS command injection in nginxWebUI	This vulnerability is caused by **improper handling of file arguments** in the upload feature, allowing attackers to inject and execute OS commands remotely
Default file permissions vulnerability	This vulnerability occurs because the Nginx Management Suite sets default file permissions allowing attackers to **modify sensitive files**

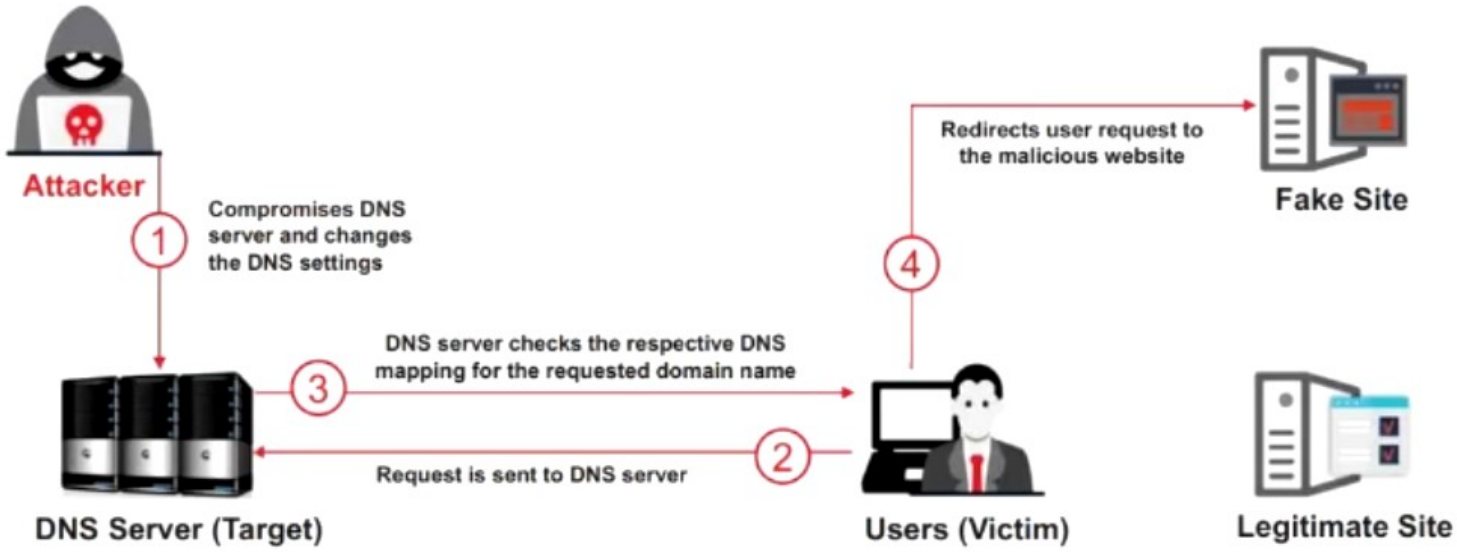
https://cve.mitre.org

Objective 02

Demonstrate Different Web Server Attacks

DNS Server Hijacking

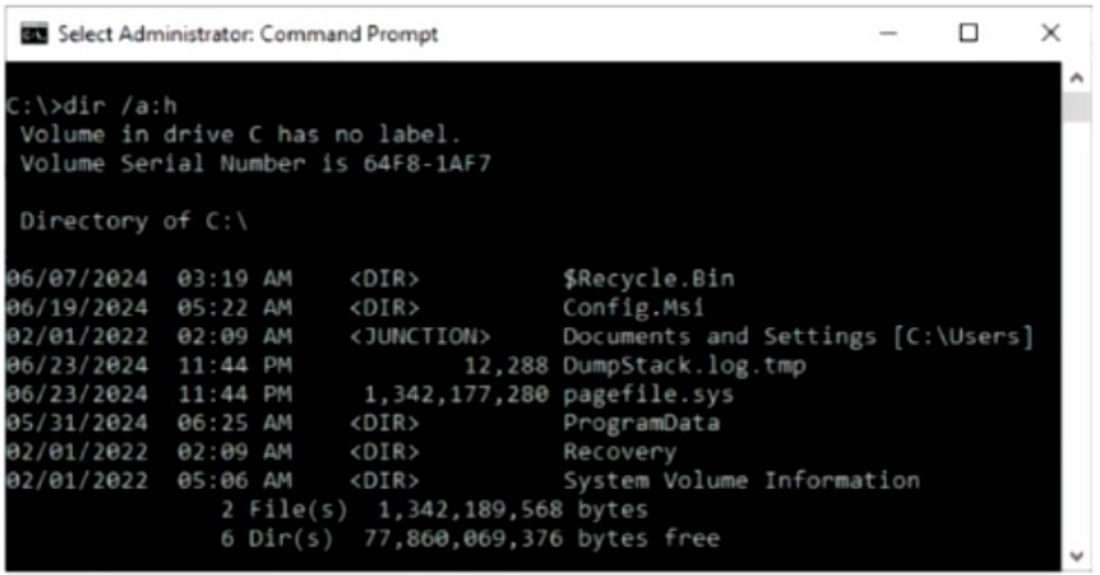
Attacker compromises the DNS server and **changes the DNS settings** so that all the requests coming towards the target web server are redirected to his/her own malicious server

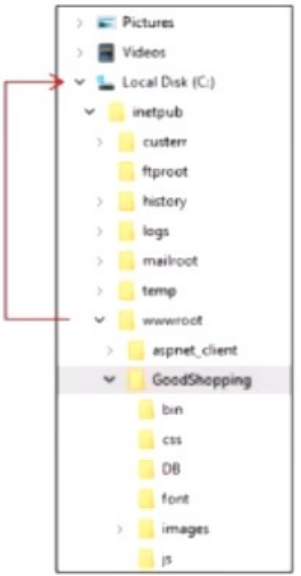


Directory Traversal Attacks

- In directory traversal attacks, attackers use the **../ (dot-dot-slash)** sequence to access restricted directories outside the web server root directory
- Attackers can use the **trial and error method** to navigate outside the root directory and access sensitive information in the system

http://server.com/scripts/..%5c../Windows/System32/cmd.exe?/c+dir+c:\





Web Server Misconfiguration

- Server misconfiguration refers to **configuration weaknesses in web infrastructure** that can be exploited to launch various attacks on web servers such as directory traversal, server intrusion, and data theft
- Some web server misconfigurations may include verbose debug/error messages, anonymous or default users/passwords, unnecessary services enabled, etc.

Web Server Misconfiguration Examples

This configuration allows anyone to view the **server status** page, which contains detailed information about the web server being currently used, including information about the **current hosts** and requests being processed

httpd.conf file on an **Apache** server

```
<Location "/server-status">
   SetHandler server-status
   Require host example.com
</Location>
```

This configuration can **enable directory browsing** leading to expose sensitive files

web.config file on an **IIS** server

```
<system.webServer>
   <directoryBrowse enabled="true"/>
</system.webServer>
```

This configuration can lead to **unsafe variable usage**

nginx.conf file

```
location / {
   set $variable $arg_user_input;
   proxy_pass http://backend/$variable;
}
```

HTTP Response-Splitting Attack

1. HTTP response splitting attack involves **adding header response data into the input field** so that the server splits the response into two responses
2. The attacker can exploit a vulnerable Apache HTTP server by leveraging **improper input validation** and injecting a **crafted request header** that elicits two responses from the server
3. The attacker can **control the first response to redirect the user to a malicious website** whereas the other responses are discarded by the web browser

Server Code

```
String author =
request.getParameter(AUTHOR_PARAM);
...
Cookie webcomic = new Cookie("author",
author); cookie.setMaxAge(cookieExpiration);
response.addCookie(cookie);
```

Input = Jason

HTTP/1.1 200 OK
...
Set-Cookie: author=Jason
...

Input = JasonTheHacker\r\nHTTP/1.1 200 OK\r\n

First Response (Controlled by Attacker)

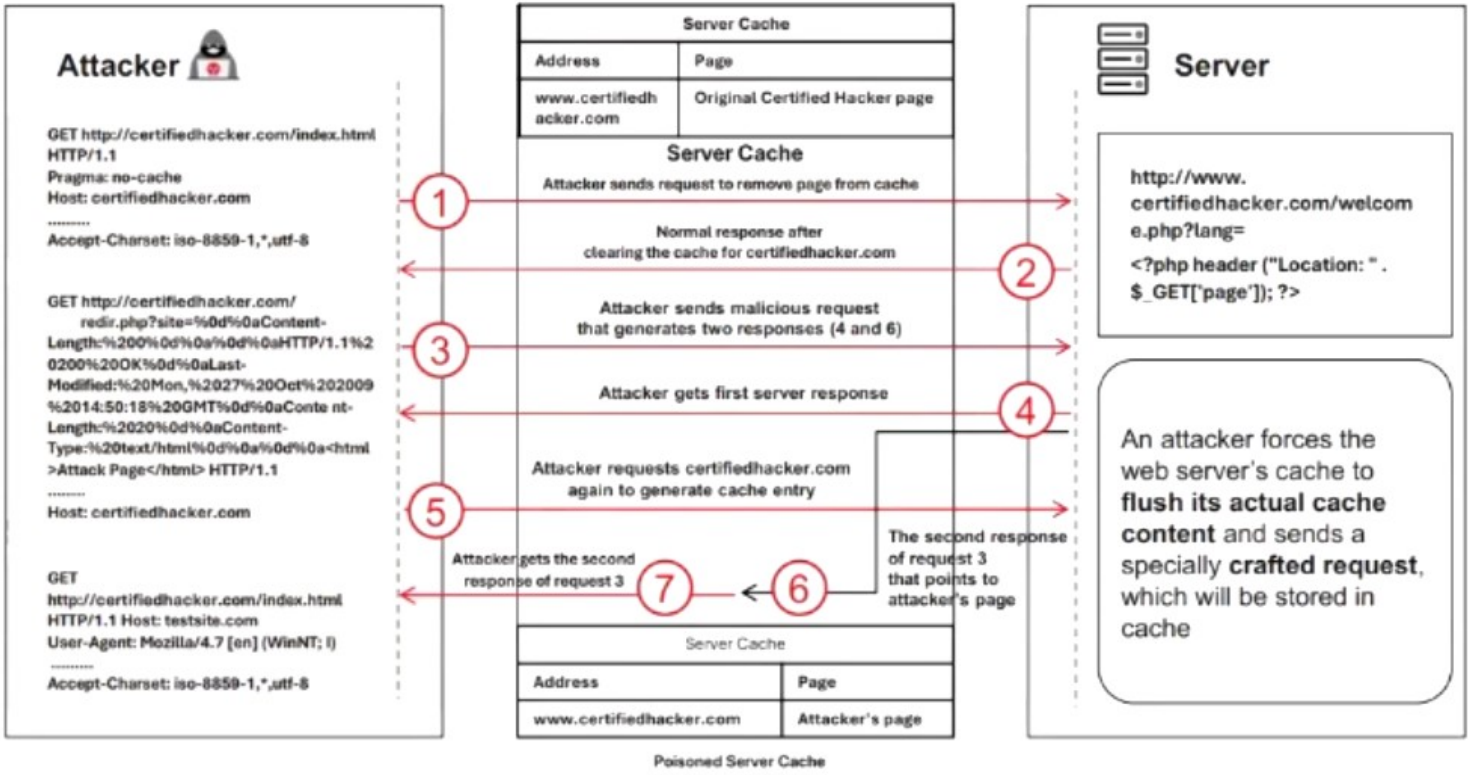
Set-Cookie: author=JasonTheHacker
HTTP/1.1 200 OK
...

Second Response

HTTP/1.1 200 OK
...

Web Cache Poisoning Attack

- Web cache poisoning attacks the **reliability of an intermediate web cache source**
- In this attack, the attackers **swap cached content** for a random URL with infected content
- Users of the web cache source can **unknowingly use the poisoned content** instead of the true and secured content when requesting the required URL through the web cache

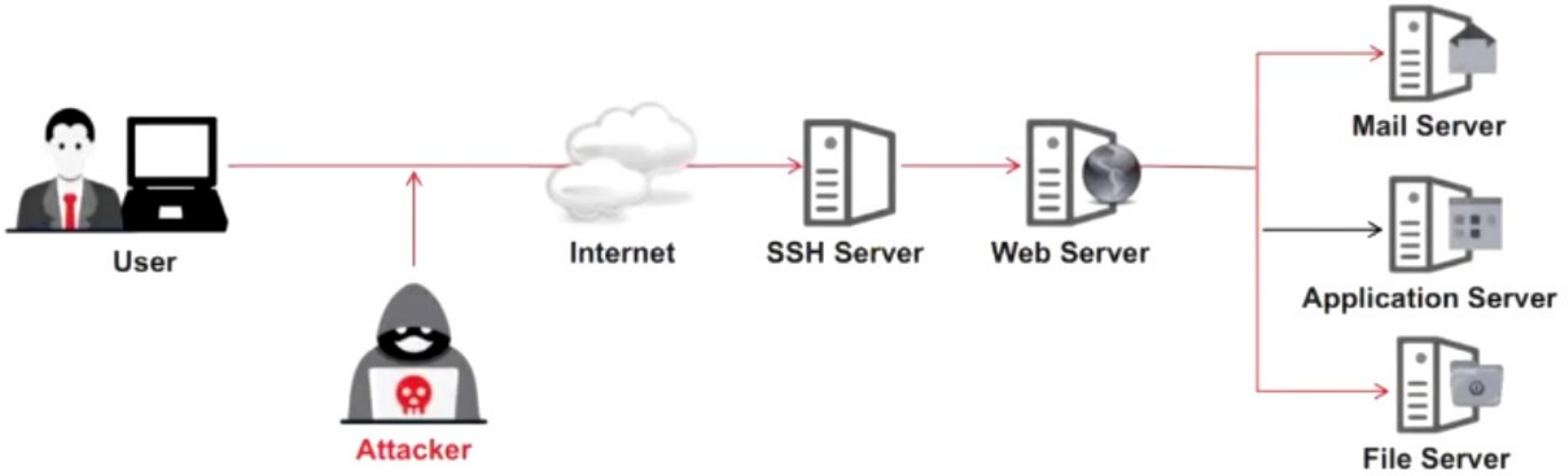


SSH Brute Force Attack

SSH protocols are used to create an **encrypted SSH tunnel** between two hosts to transfer unencrypted data over an insecure network

Attackers can brute force SSH login credentials to gain **unauthorized access to an SSH tunnel**

SSH tunnels can be used to **transmit malwares** and other exploits to victims without being detected



FTP Brute Force with AI

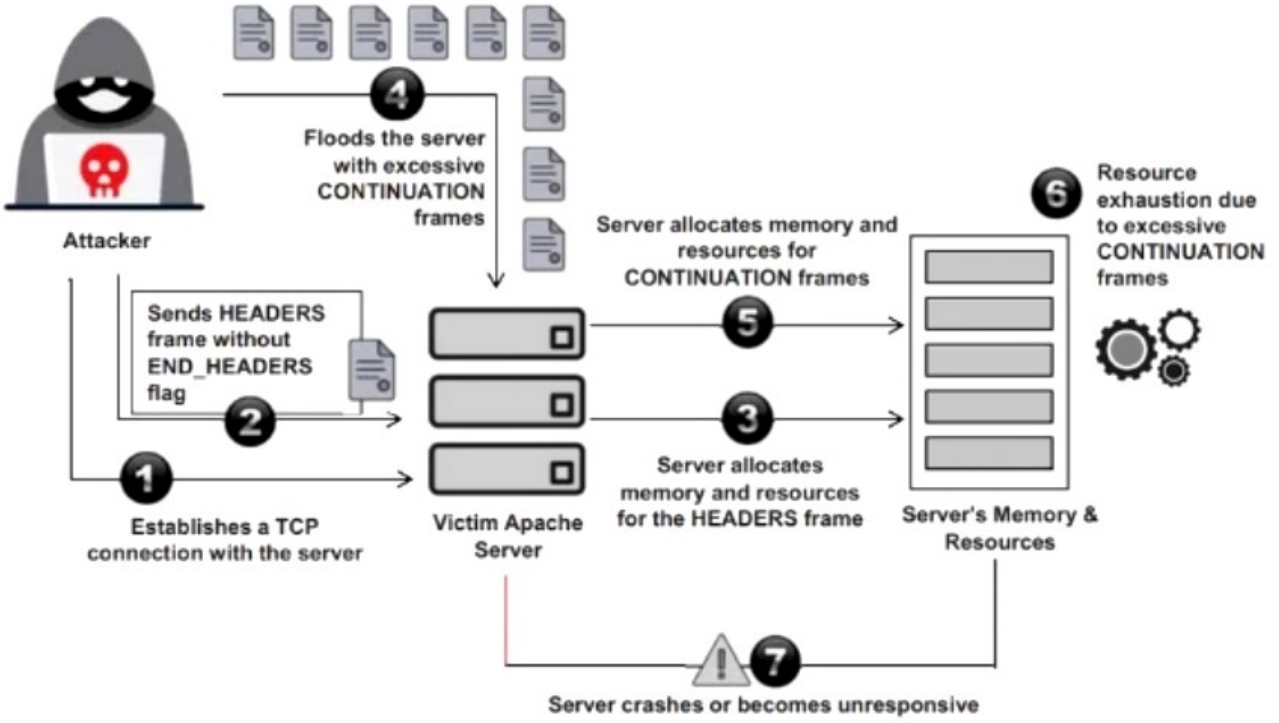
An attacker can also leverage AI-powered ChatGPT or other generative AI technology to perform this task by using an appropriate prompt such as

- ***"Attempt FTP login on target IP 10.10.1.11 with hydra using usernames and passwords from wordlists"***

```
┌──[attacker@parrot]─[~]
└─$ sgpt --shell "Atempt FTP login on target IP 10.10.1.11 with hydra using usernames and passwords
 file from wordlists"
hydra -L /usr/share/wordlists/ftp-usernames.txt -P /usr/share/wordlists/ftp-passwords.txt ftp://10.10
.1.11
[E]xecute, [D]escribe, [A]bort: E
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret serv
ice organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics any
way).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-03-14 03:06:19
[DATA] max 16 tasks per 1 server, overall 16 tasks, 2500 login tries (l:50/p:50), ~157 tries per task
[DATA] attacking ftp://10.10.1.11:21/
[21][ftp] host: 10.10.1.11   login: Martin   password: apple
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-03-14 03:06:52
```

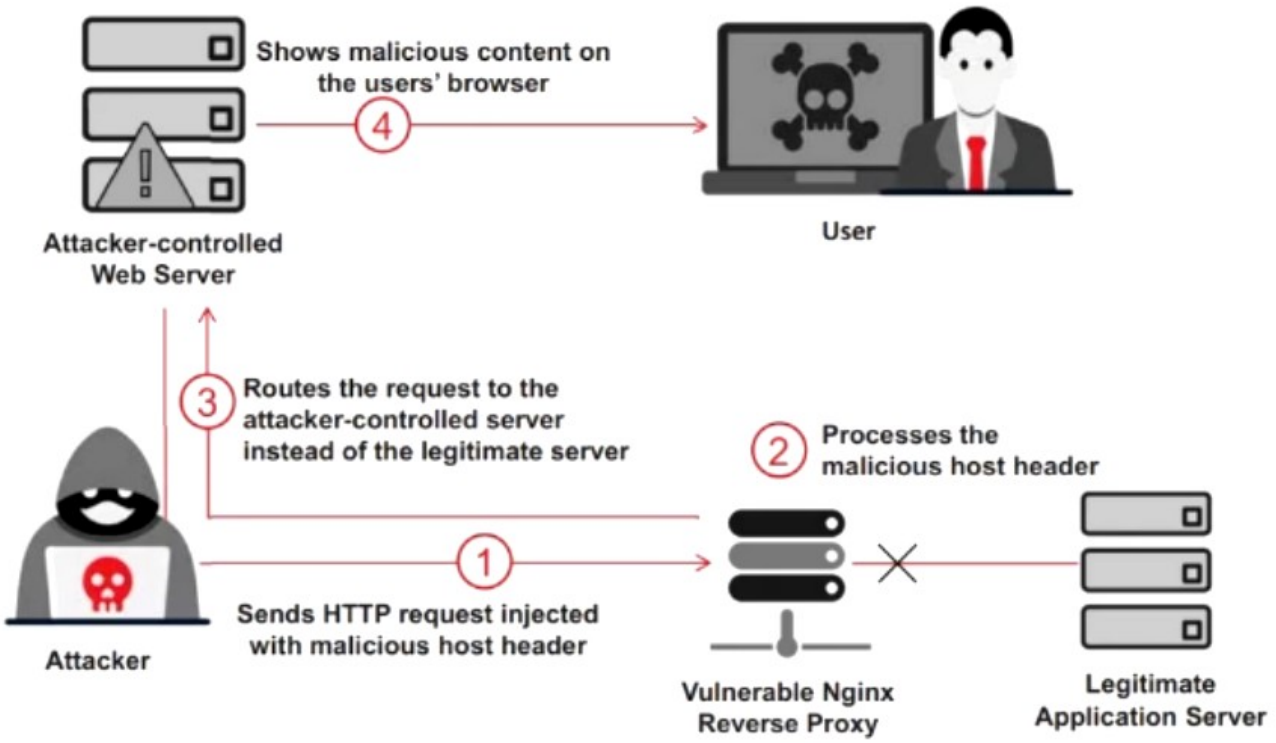
HTTP/2 Continuation Flood Attack

- The HTTP/2 continuation flood attack involves exploiting the **handling mechanism** of **HTTP/2 CONTINUATION frames**
- Attackers send numerous CONTINUATION frames over a **single TCP connection**, overwhelming the Apache server's resources, causing a **Denial-of-Service condition** on the target server



Frontjacking Attack

- FrontJacking is a web server attack in which an attacker **injects or manipulates front-end components** of a web application to hijack the user interface or user interactions
- This attack targets **poorly configured NGINX reverse proxy** servers in shared hosting environments, combining CRLF injection, HTTP request header injection, and XSS
- Attackers inject a new host header to **hijack the execution flow** of the front-end reverse proxy server and replace the accessed backend server with an attacker-controlled server



Other Web Server Attacks

Web Server Password Cracking	Attackers perform password cracking to **gain unauthorized access** to a web server by exploiting flawed and weak authentication mechanisms **Note**: For complete coverage of password cracking attacks, refer to Module 06: System Hacking
DoS/DDoS Attacks	Attackers may send numerous **fake requests** to the web server, which causes **web server crashing** or makes it unavailable to the legitimate users **Note**: For complete coverage of DoS/DDoS attacks, refer to Module 10: Denial-of-Service
Man-in-the-Middle Attack	Man-in-the-middle/manipulator-in-the-middle (MITM) attacks allow an attacker to access sensitive information by **intercepting and altering communications** between an end-user and web servers **Note**: For complete coverage of man-in-the-middle (MITM) attacks, refer to Module 11: Session Hijacking
Phishing Attacks	The attacker tricks the user to submit **login details** for a website that looks legitimate, and redirects them to the malicious website hosted on the attacker's web server **Note**: For complete coverage of phishing attacks, refer to Module 09: Social Engineering
Web Application Attacks	Vulnerabilities in **web applications** running on a web server provide a broad attack path for compromising the web servers **Note**: For complete coverage of web application attacks, refer to Module 14: Hacking Web Applications

Objective 03

Explain Web Server Attack Methodology

Web Server Attack Methodology

1. Information Gathering
2. Web Server Footprinting
3. Website Mirroring
4. Vulnerability Scanning
5. Session Hijacking
6. Web Server Passwords Hacking

Information Gathering from Robots.txt File

- The robots.txt file contains the **list of the web server directories and files** that the web site owner wants to hide from web crawlers
- An attacker can simply request the Robots.txt file from the URL and retrieve sensitive information such as the **root directory structure** and **content management system information** about the target website
- An attacker can also download the Robots.txt file of a target website using the Wget tool

Web Server Footprinting/Banner Grabbing

- Gather **valuable system-level data** such as account details, operating system, software versions, server names, and database schema details

Netcat

- This utility **reads and writes data across network connections**, using the TCP/IP protocol

  `# nc -vv www.microsoft.com 80` – press **[Enter]**

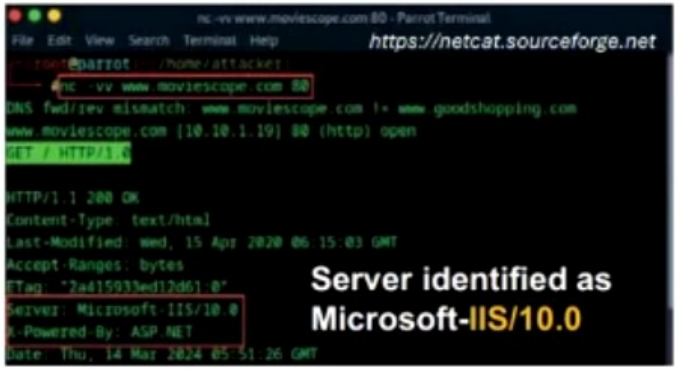
  `GET / HTTP/1.0` - Press **[Enter] twice**

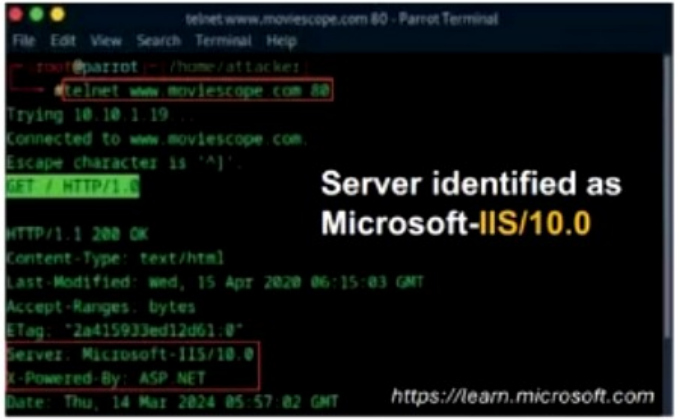
Telnet

- This technique probes **HTTP servers** to determine the **Server field** in the HTTP response header

  `telnet www.moviescope.com 80` – press **[Enter]**

  `GET / HTTP/1.0` - Press **[Enter] twice**





Web Server Footprinting with AI

An attacker can also leverage AI-powered ChatGPT or other generative AI technology to perform this task by using an appropriate prompt such as

- ***"Perform webserver footprinting on target IP 10.10.1.22"***
- ***"Perform webserver footprinting on target IP 10.10.1.22 with netcat"***

IIS Information Gathering using Shodan

Attackers can use tools such as Shodan to gather information about IIS servers, enhancing their ability to perform targeted and efficient attacks

Shodan Search Filters for Information Gathering

- Use the following filter to identify the IIS servers with **SSL certificates** issued to "Company Inc.":

  ```
  Ssl:"Company Inc." http.title:"IIS"
  ```

- Use the following filter to locate IIS servers with SSL certificates where the **common name** (CN) is "**company.in**":

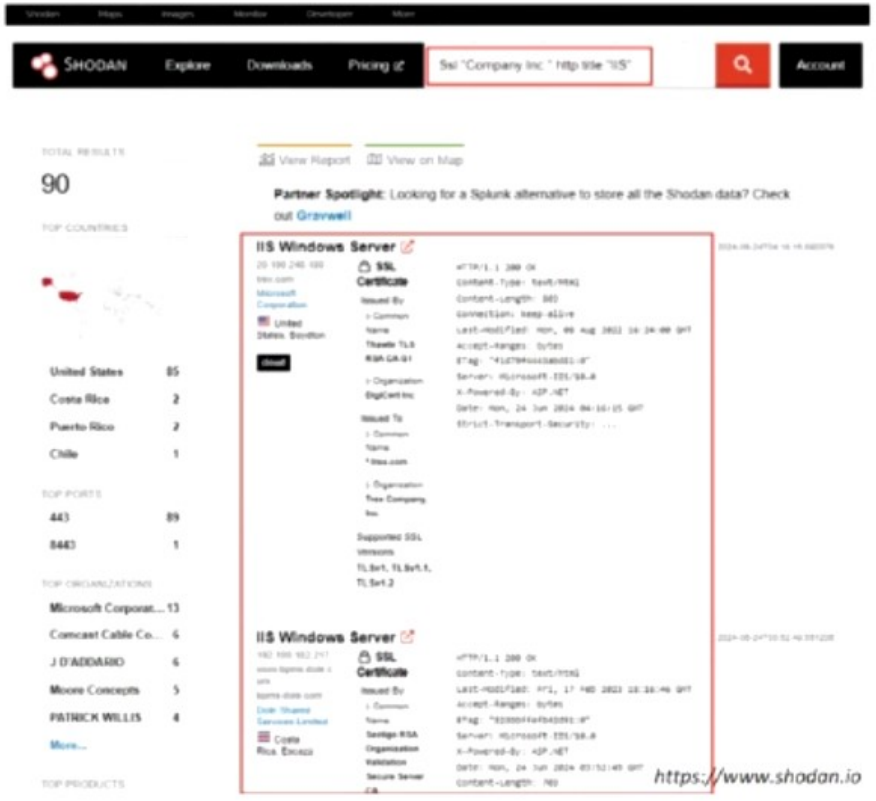
  ```
  Ssl.cert.subject.CN:"company.in" http.title:"IIS"
  ```

- Use the following filter to find **IIS servers in the US**:

  ```
  http.title:"IIS Windows Server" country:"US"
  ```

- Use the following filter to locate IIS7 servers running on **port 80**:

  ```
  http.title:"IIS7" port:80
  ```



Abusing Apache mod_userdir to Enumerate User Accounts

- Attackers can exploit Apache's **'mod_userdir'** module to enumerate user accounts using URIs such as **/~username/**
- Using **Nmap**, attackers can identify valid usernames, which can aid in brute forcing or targeted phishing

Enumerating User Accounts from the Apache Server

Perform Initial Scan to Enumerate Valid Users

- `nmap -p80 --script http-userdir-enum <target>`
- Scans port 80 using the http-userdir-enum script to list usernames matching with default word list

Perform Customized Scan

- `nmap -p80 --script http-userdir-enum --script-args userdir.users=<Wordlist.txt> <target>`
- Takes the '.txt' file as a source for usernames and tests against the target

Bypass Detection with Custom User Agent

- `nmap -p80 --script http-brute --script-args http.useragent="<User_Agent>" <target>`
- Changes the HTTP User Agent string to avoid detection by security systems

Directory Brute Forcing

When a web server receives a request for the directory, it responds to the request in the following ways:

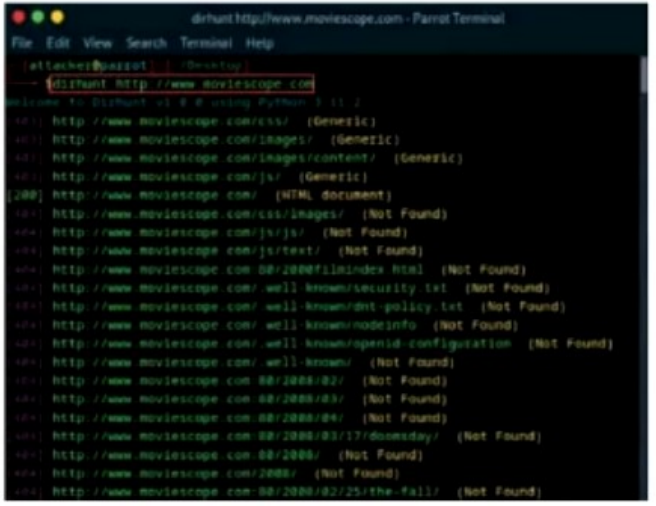
- Returns **default resource within the directory**
- Returns **error**
- Returns **listing of directory content**

Directory listings sometimes possess the following **vulnerabilities** that allow the attackers to **compromise the web server**:

- Improper **access controls**
- Unintentional **access to the web root** of servers

- After discovering the directory on the web server, **make a request** for the **same directory** and try to **access the directory listings**
- Try to **exploit vulnerable web server software** that gives **access to the directory listings**

Attackers use tools such as **Dirhunt** to search and analyze directories

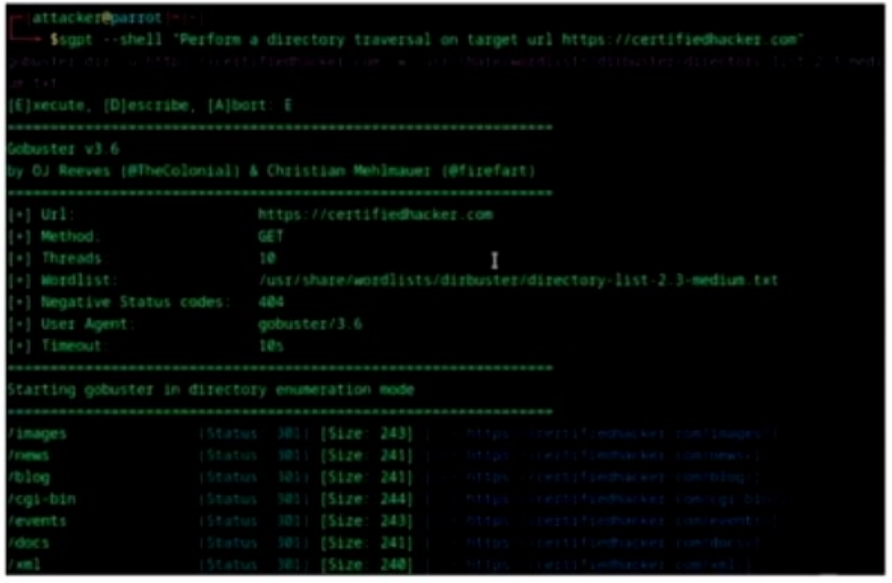


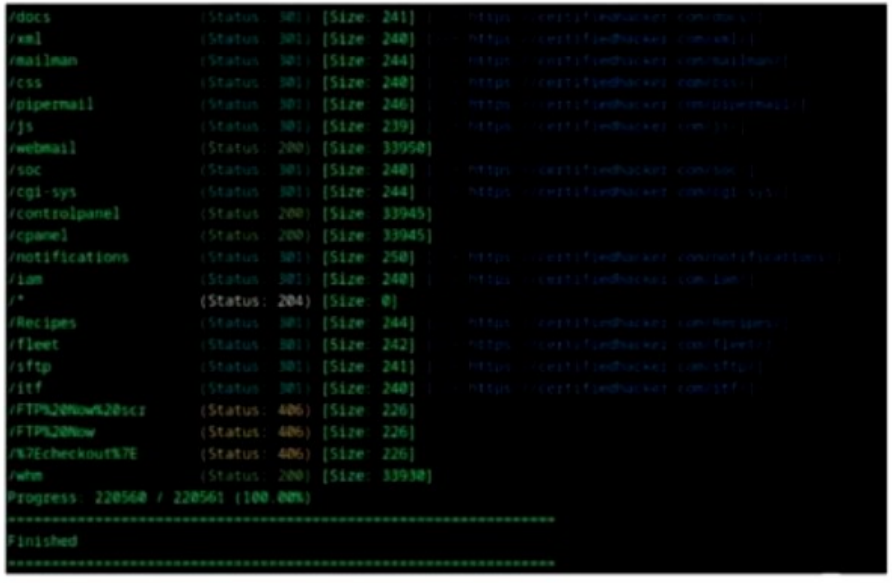
https://github.com

Directory Brute Forcing with AI

An attacker can also leverage AI-powered ChatGPT or other generative AI technology to perform this task by using an appropriate prompt such as

- ***"Perform a directory traversal on target url https://certifiedhacker.com"***





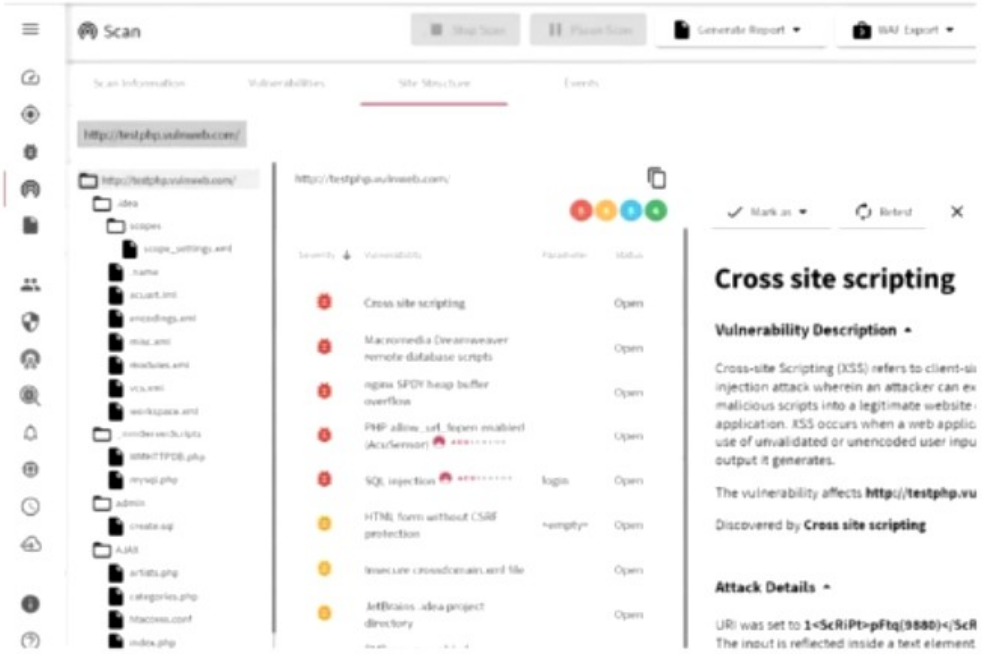
Vulnerability Scanning

Implement vulnerability scan to **identify weaknesses** in a network and determine if the system can be exploited

Use vulnerability scanners such as Acunetix Web Vulnerability Scanner, and Fortify WebInspect to find **hosts**, **services**, and **vulnerabilities**

Sniff the network traffic to find any **active systems**, **network services**, **applications**, and vulnerabilities present

Test the **web server infrastructure** for any misconfigurations, outdated content, and vulnerabilities using vulnerability scanners like **Acunetix Web Vulnerability Scanner**



Other Vulnerability Scanning Tools

OpenText Fortify WebInspect
https://www.opentext.com

Tenable.io
https://www.tenable.com

ImmuniWeb
https://www.immuniweb.com

Invicti
https://www.invicti.com

Nginx Vulnerability Scanning using Nginxpwner

Nginxpwner is a Python-based tool designed to **identify common misconfigurations** and **vulnerabilities** in Nginx web servers that could be exploited

Scanning Nginx Web Server for Vulnerabilities

- Run the following command to create a temporary file to store a list of potential URL paths

```
nano /tmp/pathlist
```

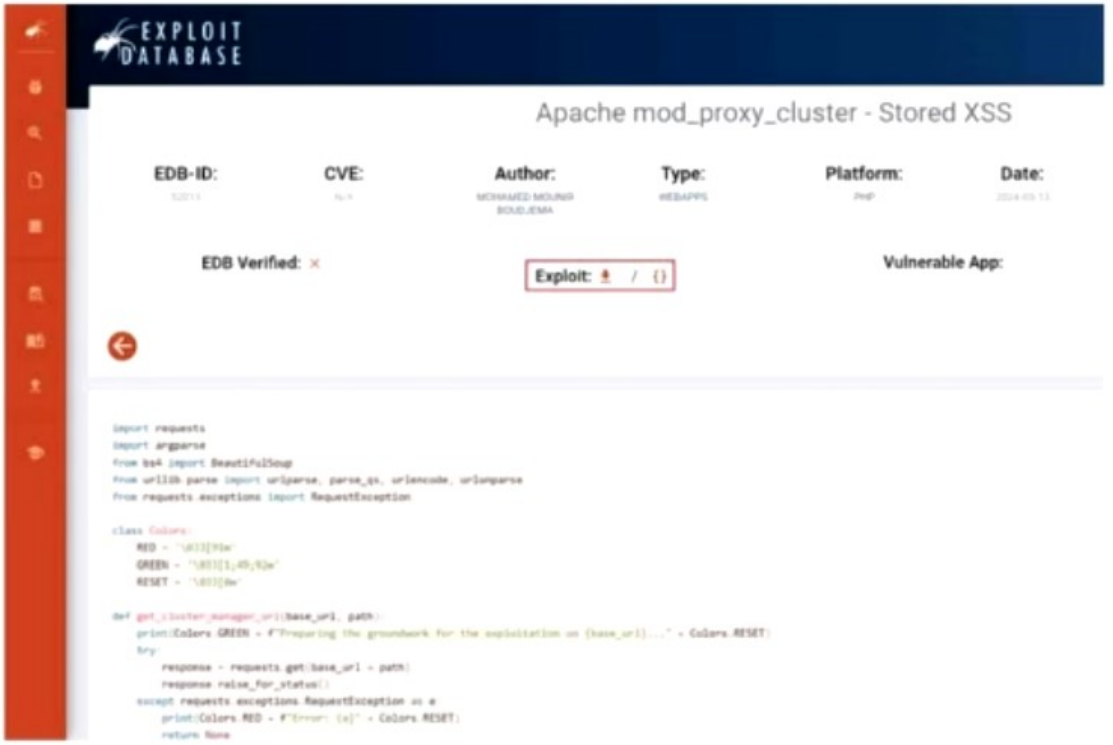
- Execute the nginxpwner script

```
python3 nginxpwner.py <target_URL>
/tmp/pathlist
```

- Analyze the output provided by Nginxpwner to identify potential vulnerabilities and misconfigurations in the target Nginx server

Finding Exploitable Vulnerabilities

- Navigate to the **Exploit Database** (*https://www.exploit-db.com*) website
- Use the search bar at the top of the page to enter keywords related to the web server you are targeting (e.g., **"Apache," "IIS," "nginx"**)
- **Apply filters** to narrow down the search results
- Review the search results to **identify relevant vulnerabilities**
- Click on an **exploit** to view detailed information, including **exploit code**, description, and references
- Cross-check the identified vulnerabilities with the **versions** and **configurations** of the target web server to ensure they are applicable
- Download the **exploit code** if it is relevant and test it on the target web server



https://www.exploit-db.com

Finding Exploitable Vulnerabilities with AI

An attacker can also leverage AI-powered ChatGPT or other generative AI technology to perform this task by using an appropriate prompt such as

- "***Identify OS and services running on the target 10.10.1.19 and then install and launch the searchsploit to find out the possible exploits associated with OS and services identified***"

```
┌──[root@parrot]─[/home/attacker]
└──╼ #sgpt --chat wsh --shell " Identify OS and services running on target 10.10
.1.19 and then install and launch the searchsploit to find out the possible expl
oits asscoaited with OS and services identified"
Please enter your OpenAI API key:
sudo apt-get update && sudo apt-get install nmap -y && nmap -sV -O 10.10.1.19 -o
X nmap_scan.xml && sudo apt-get install exploitdb -y && searchsploit --nmap nmap
_scan.xml
[E]xecute, [D]escribe, [A]bort: E
Hit:1 https://deb.parrot.sh/parrot lory InRelease
Hit:2 https://deb.parrot.sh/direct/parrot lory-security InRelease
Hit:3 https://deb.parrot.sh/parrot lory-backports InRelease
Reading package lists... Done
W: Target Packages (main/binary-amd64/Packages) is configured multiple times in
/etc/apt/sources.list:1 and /etc/apt/sources.list.d/parrot.list:19
W: Target Packages (main/binary-all/Packages) is configured multiple times in /e
tc/apt/sources.list:1 and /etc/apt/sources.list.d/parrot.list:19
```

```
Host is up (0.00049s latency).
Not shown: 990 filtered tcp ports (no-response)
PORT     STATE SERVICE       VERSION
25/tcp   open  smtp          Microsoft ESMTP 10.0.17763.1
80/tcp   open  http          Microsoft IIS httpd 10.0
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds?
1801/tcp open  msmq?
2103/tcp open  msrpc         Microsoft Windows RPC
2105/tcp open  msrpc         Microsoft Windows RPC
2107/tcp open  msrpc         Microsoft Windows RPC
3389/tcp open  ms-wbt-server Microsoft Terminal Services
MAC Address: 02:15:5D:33:88:D1 (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1 o
pen and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Microsoft Windows 2019 (97%)
Aggressive OS guesses: Microsoft Windows Server 2019 (97%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
```

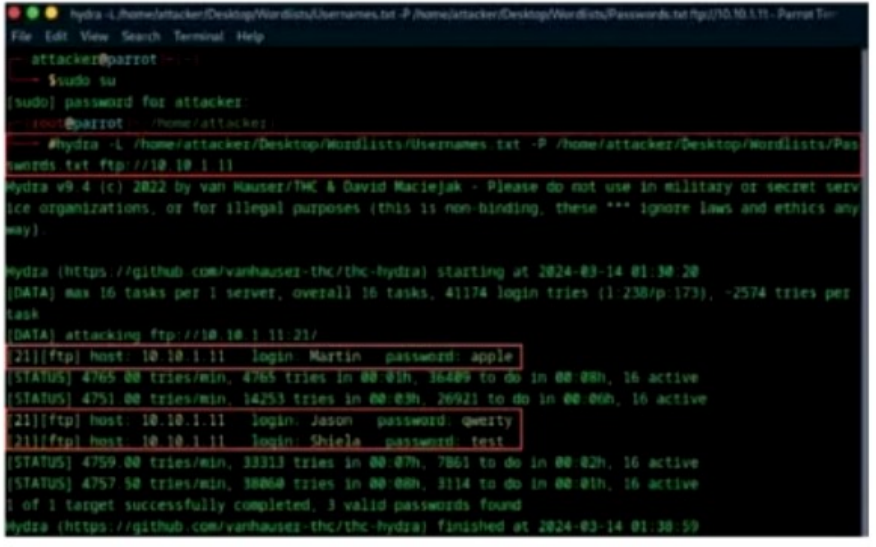
Finding Exploitable Vulnerabilities with AI (Cont'd)

```
[i] /usr/bin/searchsploit -t smtp
---------------------------------------------- ---------------------------
 Exploit Title                                |  Path
---------------------------------------------- ---------------------------
AA SMTP Server 1.1 - Crash (PoC)              | windows/dos/14990.txt
Alt-N MDaemon 6.5.1 - IMAP/SMTP Remote Buffer | windows/remote/473.c
Alt-N MDaemon 6.5.1 SMTP Server - Multiple Co | windows/remote/24624.c
Alt-N MDaemon Server 2.71 SP1 - SMTP HELO Arg | windows/dos/23146.c
Apache James Server 2.2 - SMTP Denial of Serv | multiple/dos/27915.pl
BaSoMail 1.24 - SMTP Server Command Buffer Ov | windows/dos/22668.txt
BaSoMail Server 1.24 - POP3/SMTP Remote Denia | windows/dos/594.pl
BL4 SMTP Server < 0.1 5 - Remote Buffer Overf | windows/dos/1721.pl
Blat 2.7.6 SMTP / NNTP Mailer - Local Buffer  | windows/local/38472.py
BulletProof FTP Server 2019.0.0.50 - 'SMTP Se | windows/dos/46422.py
Cisco PIX Firewall 4.x/5.x - SMTP Content Fil | hardware/remote/20231.txt
Citadel SMTP 7.10 - Remote Overflow           | windows/remote/4949.txt
Cobalt Raq3 PopRelayD - Arbitrary SMTP Relay  | linux/remote/20994.txt
CodeBlue 5.1 - SMTP Response Buffer Overflow  | windows/remote/21643.c
```

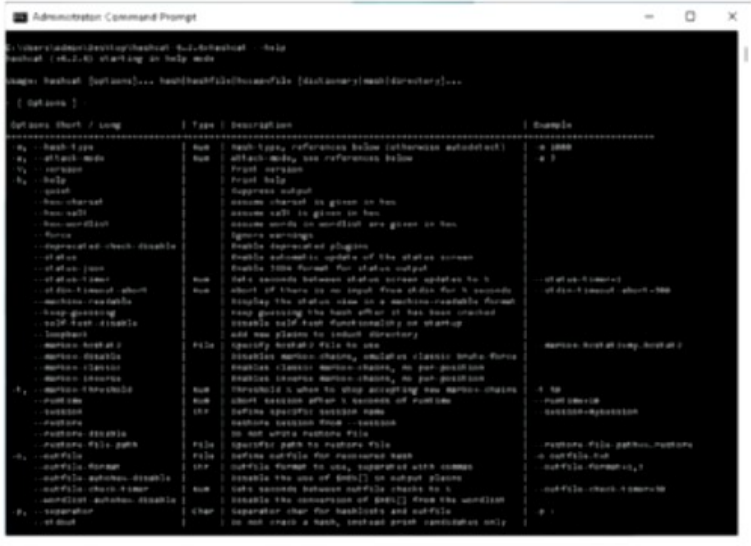
```
[i] /usr/bin/searchsploit -t microsoft esmtp
[-] Skipping term: http   (Term is too general. Please re-search manually: /usr/
bin/searchsploit -t http)

[i] /usr/bin/searchsploit -t microsoft iis httpd
[i] /usr/bin/searchsploit -t msrpc
[i] /usr/bin/searchsploit -t microsoft windows rpc
---------------------------------------- ----------------------------
 Exploit Title                          |  Path
---------------------------------------- ----------------------------
Microsoft Windows - 'Lsasrv.dll' RPC Remote B | windows/remote/293.c
Microsoft Windows - 'RPC DCOM' Long Filename  | windows/remote/100.c
Microsoft Windows - 'RPC DCOM' Remote (1)     | windows/remote/69.c
Microsoft Windows - 'RPC DCOM' Remote (2)     | windows/remote/70.c
Microsoft Windows - 'RPC DCOM' Remote (Univer | windows/remote/76.c
Microsoft Windows - 'RPC DCOM' Remote Buffer  | windows/remote/64.c
```

Web Server Password Hacking

- Use password cracking techniques such as **brute force attack**, **dictionary attack**, and **password guessing** to crack web server passwords
- Use tools such as **Hashcat**, THC Hydra, and Ncrack



https://github.com



https://hashcat.net

Path Traversal via Misconfigured NGINX Alias

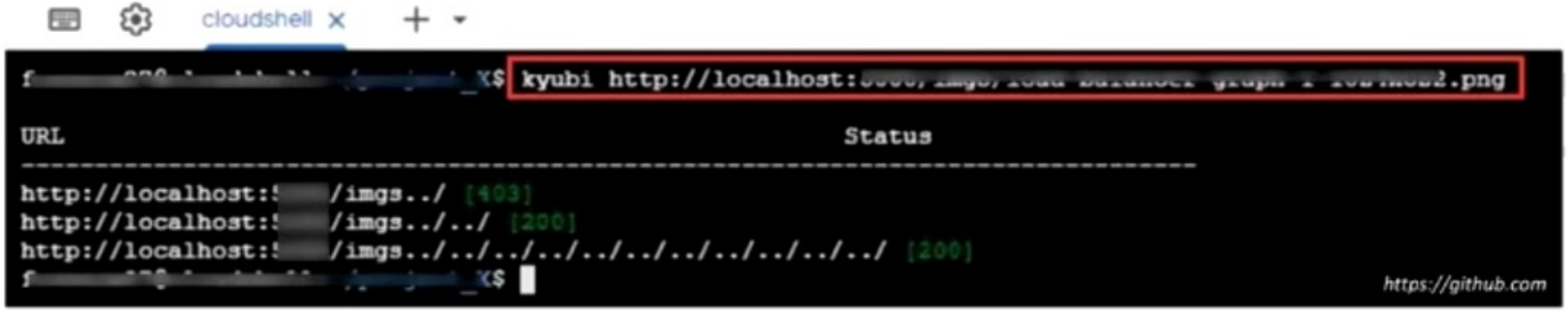
- An attacker can exploit a misconfiguration in the nginx.conf file of a targeted Nginx web server, specifically involving improper usage of the alias directive.

  Webroot misconfiguration in nginx.conf file:

  ```
  location /i {
      alias /data/w3/images/;
  }
  ```

  Lack of a trailing slash allows attackers to manipulate the URL to access directories and files outside the designated Webroot

Using Kyubi for Exploitation:



Objective 04

Explain Web Server Attack Countermeasures

Countermeasures: Patches and Updates

1. Scan for existing vulnerabilities, patch, and update the **server software regularly**
2. Before applying any service pack, hotfix, or security patch, **read and peer review** all relevant documentation
3. Apply all updates, regardless of their type on an **"as-needed"** basis
4. Test service packs and hotfixes on a representative **non-production environment** prior to their deployment in production
5. Ensure that service packs, hotfixes, and security patch levels are consistent on **all domain controllers (DCs)**
6. Ensure that **server outages** are scheduled, and that a complete set of **backup tapes** and emergency repair disks are available
7. Keep a **back-out plan** that allows the system and enterprise to return to their original state, prior to a failed implementation
8. Schedule periodic service-pack upgrades as part of operations maintenance and never trail by **more than two service packs**

Countermeasures: Protocols and Accounts

Protocols

- Block all unnecessary **ports, ICMP traffic**, and unnecessary protocols such as NetBIOS and SMB
- Harden the TCP/IP stack and consistently apply the **latest software patches** and updates to system software
- If insecure protocols such as **Telnet**, **POP3**, **SMTP**, and **FTP** are used, then take appropriate measures to provide secure authentication and communication, for example, by using IPsec policies
- If remote access is needed, ensure that remote connections are secured properly by using **tunneling and encryption protocols**
- Disable **WebDAV** if it is not used by the application or keep it secure if it is required

Accounts

- Remove all unused **modules and application extensions**
- **Disable unused default user accounts** created during the installation of an OS
- When creating a new web root directory, **grant appropriate (least possible) NTFS permissions** to anonymous users of the IIS web server to access the web content
- **Eliminate unnecessary database users** and stored procedures and follow the principle of least privilege for the database application to defend against SQL query poisoning
- Use secure web permissions, NTFS permissions, and **.NET Framework access control mechanisms** including URL authorization
- Slow down brute force and dictionary attacks with **strong password policies** and implement audits and alerts for login failures

Countermeasures: Files and Directories

1. Eliminate unnecessary files within the **.jar files**
2. Eliminate **sensitive configuration** information within the **byte code**
3. Avoid mapping **virtual directories** between two different servers, or over a network
4. Monitor and check all **network services logs**, **website access logs**, **database server logs** (e.g., Microsoft SQL Server, MySQL, Oracle), and OS logs frequently
5. Disable the serving of **directory listings**
6. **Eliminate non-web files** such as archive files, backup files, text files, and header/include files
7. Disable the serving certain **file types** by creating a resource mapping
8. Ensure that **web applications** or **website files** and **scripts** are stored in a partition or drive separate from that of the OS, logs, and any other system files

Detecting Web Server Hacking Attempts

Use a **Website Change Detection System** to detect hacking attempts on the web server

Website Change Detection System involves:

1. **Running specific script** on the server that detects any changes made in the existing executable file or new file included on the server
2. Periodically comparing the **hash values** of the files on the server with their respective master hash value to detect the changes made in codebase
3. **Alerting the user** upon any change detected on the server

For example: **DirectoryMonitor** is an automated tool that goes through all your web folders and detects any changes made to your codebase and alerts you via an email, if changes are detected

How to Defend against Web Server Attacks

Ports

- Regularly audit the ports on the server to ensure that an **insecure** or unnecessary service is not active on your web server
- Limit inbound traffic to **port 80 for HTTP** and **port 443 for HTTPS** (SSL)
- Encrypt or restrict **intranet traffic**

Server Certificates

- Ensure that **certificate data ranges** are valid and that the certificates are used for their intended purpose
- Ensure that no certificate has been revoked and the **certificate's public key** is valid all the way to a trusted root authority

Machine.config

- Ensure that protected resources are mapped to **HttpForbiddenHandler** and unused **HttpModules** are removed
- Ensure that **tracing is disabled** <trace enable="false"/> and **debug compiles** are turned off

Code Access Security

- Implement **secure coding** practices
- Restrict **code access security policy** settings
- **Configure IIS** to reject URLs with "../" and install new patches and updates

How to Defend against Web Server Attacks (Cont'd)

1. Apply **restricted ACLs** and block remote registry administration
   - Secure the **SAM** (Stand-alone servers only)
2. Ensure that security-related settings are **configured appropriately** and that access to the metabase file is restricted with hardened **NTFS permissions**
3. **Remove** unnecessary ISAPI filters from the web server
4. Remove all unnecessary file shares, including the **default administration shares**, if they are not required
   - Secure the shares with restricted **NTFS permissions**
5. Relocate sites and virtual directories to **non-system partitions** and use IIS web permissions to restrict access
6. Remove all unnecessary **IIS script mappings** for optional file extensions to avoid exploitation of any bugs in the ISAPI extensions that handle these types of files
7. Enable a **minimum level of auditing** on the web server and use NTFS permissions to protect log files

How to Defend against Web Server Attacks (Cont'd)

8. Use a **dedicated machine** as a web server
9. Create **URL mappings** to internal servers cautiously
10. Do not install the **IIS server** on a domain controller
11. Use server-side **session ID tracking** and match connections with timestamps, IP addresses, etc.
12. If a database server such as **Microsoft SQL Server** is to be used as a backend database, install it on a **separate server**
13. Use security tools provided with **web server software** and **scanners** that automate and simplify the process of securing a web server
14. Physically protect the **web server machine** in a secure machine room
15. Do not connect an IIS Server to the **Internet** until it is fully hardened
16. Do not allow anyone to **locally log in** to the machine except the administrator
17. Configure a **separate anonymous user account** for each application if multiple web applications are hosted
18. Limit the **server functionality** to support only the web technologies to be used
19. Screen and filter **incoming traffic requests**

How to Defend against HTTP Response-Splitting and Web Cache Poisoning

Server Admin

- Use the latest **web server software**
- Regularly **update/patch the OS** and web server
- Run a **web Vulnerability Scanner**

Application Developers

- Restrict web application access to **unique IPs**
- Disallow **carriage return** (%0d or \r) and line feed (%0a or \n) characters
- Comply with **RFC 2616** specifications for HTTP/1.1

Proxy Servers

- Avoid sharing **incoming TCP connections** among different clients
- Use different TCP connections with the proxy for different **virtual hosts**
- Implement "**maintain request host header**" correctly

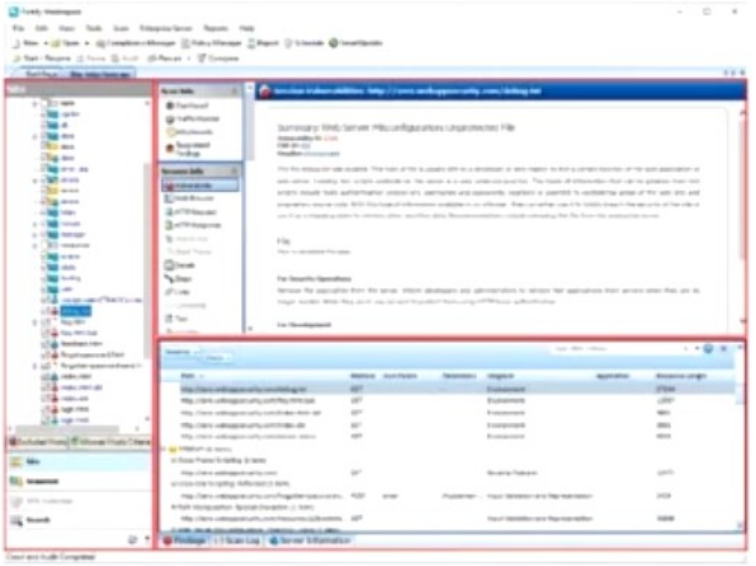
How to Defend against DNS Hijacking

1. Choose an **ICANN** accredited **registrar** and encourage them to set **Registrar-Lock** on the domain name
2. Safeguard the **registrant's account information**
3. Include DNS hijacking in **incident response and business continuity planning**
4. Use DNS monitoring tools/services to **monitor the IP address of the DNS server** and set alerts
5. Avoid downloading **audio and video codecs** and other downloaders from untrusted websites
6. Install an **antivirus** program and update it regularly
7. Change the **default router password** that comes with the factory settings

Web Server Security Tools

OpenText Fortify WebInspect

OpenText Fortify WebInspect is an **automated dynamic application security testing solution** that discovers configuration issues and identifies and prioritizes security vulnerabilities in running applications



https://www.opentext.com

Acunetix Web Vulnerability Scanner
https://www.acunetix.com

NetIQ Secure Configuration Manager
https://www.netiq.com

SAINT Security Suite
https://www.carson-saint.com

Sophos Intercept X for Server
https://www.sophos.com

UpGuard
https://www.upguard.com

Web Server Pen Testing Tools

CORE Impact

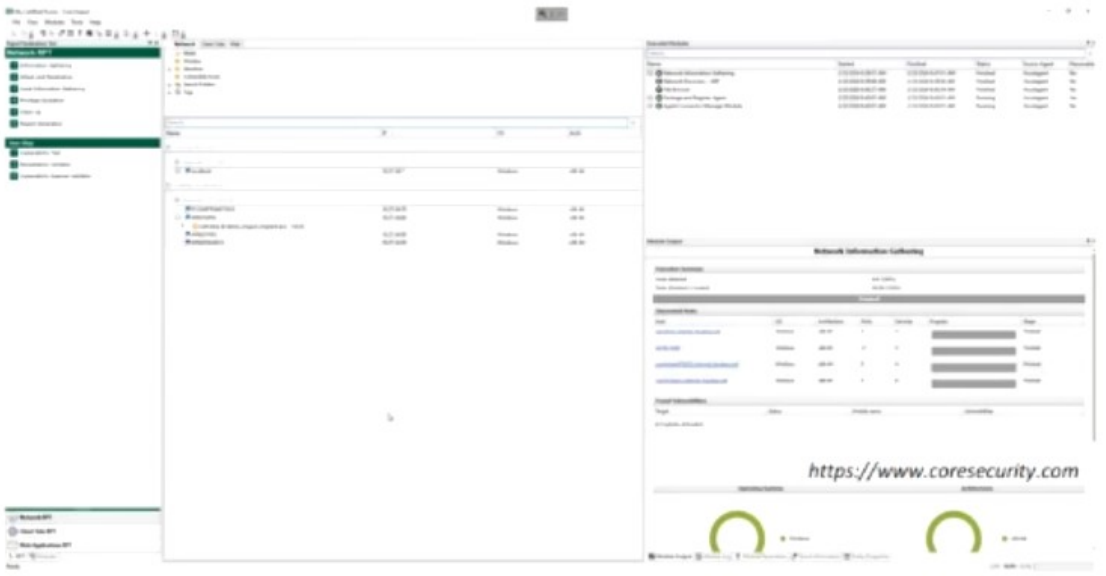
- CORE Impact **finds vulnerabilities** on an organization's web server
- This tool allows a user to **evaluate the security posture of a web server** using the present-day cybercrime techniques

Web Server Pen Testing Tools

Cobalt Strike
https://www.cobaltstrike.com

Fuxploider
https://github.com

Mitmprox
https://mitmproxy.org



Module Summary

In this module, we have discussed the following:

- Web server concepts
- Various web server threats and attacks in detail
- Web server attack methodology in detail, including information gathering, web server footprinting, vulnerability scanning, and web server passwords hacking
- Various countermeasures that are to be employed to prevent web server hacking attempts by threat actors
- Detailed discussion on securing web servers using various security tools

In the next module, we will discuss in detail how attackers, as well as ethical hackers and pen-testers, hack web applications