Chapter 13: Hacking Web Servers

Introduction

Cybercriminals are always searching for vulnerabilities, and web servers often serve as the gateway to critical systems. Poor configurations, outdated software, or inherent flaws in web server platforms can leave organizations exposed to various threats. Hacking web servers involves exploiting these vulnerabilities to gain unauthorized access, manipulate content, or compromise sensitive data. Understanding how these attacks occur is essential to securing web infrastructure.

In this chapter, you will explore:

- The fundamentals of web server operations and their role in hosting websites
- Common vulnerabilities that make web servers an attractive target
- Techniques attackers use to exploit web server weaknesses
- Tools leveraged for identifying and exploiting web server vulnerabilities
- The impacts of web server attacks on organizations
- Best practices and countermeasures to secure web servers from attacks

Web Server Concepts

Web server concepts encompass the principles of handling client requests, managing content delivery, and ensuring secure communication via HTTP/HTTPS. They facilitate both static and dynamic content processing, employ load balancing for optimal performance, and utilize caching mechanisms to enhance efficiency. Security measures, including SSL/TLS encryption, authentication, and firewall protection, safeguard data transmission. Additionally, web servers integrate with databases, APIs, and content management systems (CMS) to support dynamic applications and services.

Web Server Operations

A Web Server is a program that hosts websites based on both hardware and software. It delivers files and other content on the website over HyperText Transfer Protocol (HTTP). As the use of the internet and intranet has increased, web services have become a major part of the internet. They are used for delivering files, email communication, and other purposes. Whereas all web servers support HTML for basic content delivery, they support different types of application extensions. Web servers differ regarding security models, Operating Systems, and other factors.

Web Server Security Issues

A web server is a hardware/software application that hosts websites and makes them accessible over the Internet. A web server, along with a browser, successfully implements client–server model architecture. In this model, the web server plays the role of the server, and the browser acts as the client. To host websites, a web server stores the web pages of websites and delivers a particular web page upon request. Each web server has a domain name and an IP address associated with that domain name. A web server can host more than one website. Any computer can act as a web server if it has specific server software (a web server program) installed and is connected to the Internet. Web servers are chosen based on their capability to handle server-side programming, security characteristics, publishing, search engines, and site-building tools. Apache, Microsoft IIS, Nginx, Google, and Tomcat are some of the most widely used web server software. An attacker usually targets vulnerabilities in the software component and configuration errors to compromise web servers.

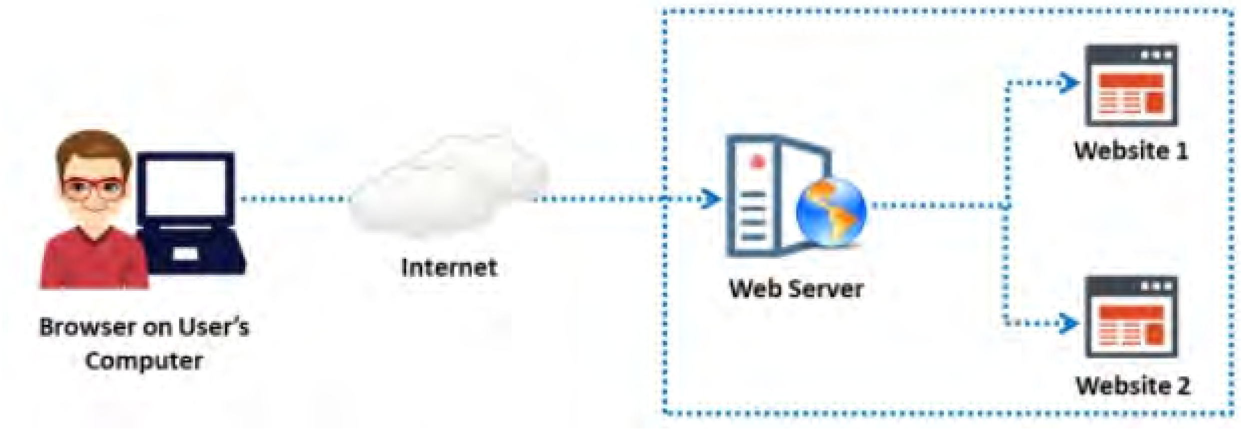


Figure 13-01: Conceptual Diagram of A Web Server

Organizations can defend most network-level and OS-level attacks by adopting network security measures such as firewalls, intrusion detection systems (IDSs), and intrusion prevention systems (IPSs) and by following security standards and guidelines. This forces attackers to turn their attention to web-server-and web-application-level attacks because a web server that hosts web applications is accessible from anywhere over the Internet. This makes web servers an attractive target. Poorly configured web servers can create vulnerabilities in even the most carefully designed firewall systems. Attackers can exploit poorly configured web servers with known vulnerabilities to compromise the security of web applications. Furthermore, web servers with known vulnerabilities can harm the security of an organization. Organizational security includes seven levels from stack 1 to stack 7.

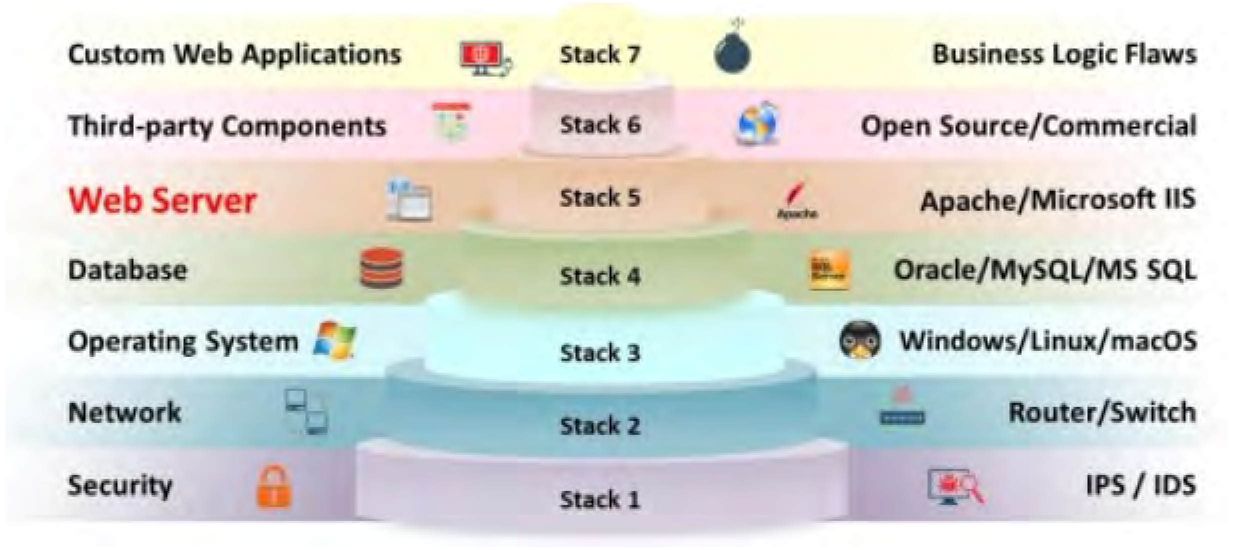


Figure 13-02: Levels of Organizational Security

Common Goals Behind Web Server Hacking

Attackers perform web server attacks with certain goals in mind. These goals may be either technical or non-technical. For example, attackers may breach the security of a web server and steal sensitive information for financial gains or merely for the sake of curiosity. The following are some common goals of web server attacks:

- Stealing credit-card details or other sensitive credentials using phishing techniques
- Integrating the server into a botnet to perform denial of service (DoS) or distributed DoS (DDoS) attacks
- Compromising a database
- Obtaining closed-source applications
- Hiding and redirecting traffic
- Escalating privileges

Some attacks are performed for personal reasons, rather than financial gains:

- For pure curiosity
- For completing a self-set intellectual challenge
- For damaging the target organization’s reputation

Dangerous Security Flaws Affecting Web Server Security

A web server configured by poorly trained system administrators may have security vulnerabilities. Inadequate knowledge, negligence, laziness, and inattentiveness toward security can pose the greatest threats to web server security.

The following are some common oversights that make a web server vulnerable to attacks:

- Failing to update the web server with the latest patches
- Using the same system administrator credentials everywhere

- Allowing unrestricted internal and outbound traffic
- Running unhardened applications and servers
- Providing complete error messages with server version information
- Using outdated SSL/TLS encryption algorithms
- Using third-party plugins in the web application

Impact Of Web Server Attacks

Attackers can cause various kinds of damage to an organization by attacking a web server. The following are some of the types of damage that attackers can cause to a web server.

- **Compromise of user accounts:** Web server attacks mostly focus on compromising user accounts. If the attacker compromises a user account, they can gain a large amount of useful information. The attacker can use the compromised user account to launch further attacks on the web server.
- **Website defacement:** Attackers can completely change the appearance of a website by replacing its original data. They deface the target website by changing the visuals and displaying different pages with messages of their own.
- **Secondary attacks from the website:** An attacker who compromises a web server can use the server to launch further attacks on various websites or client systems
- **Root access to other applications or server:** Root access is the highest privilege level to log in to a server, irrespective of whether the server is a dedicated, semi-dedicated, or virtual private server. Attackers can perform any action once they attain root access to the server.
- **Data tampering:** An attacker can alter or delete the data of a web server and even replace the data with malware to compromise users who connect to the web server
- **Data theft:** Data are among the primary assets of an organization. Attackers can attain access to sensitive data such as financial records, future plans, or the source code of a program.
- **Damage reputation of the company:** Web server attacks may expose the personal information of a company's customers to the public, damaging the reputation of the company. Consequently, customers lose faith in the company and become afraid of sharing their personal details with the company.

Why Are Web Servers Compromised?

There are inherent security risks associated with web servers, the local area networks (LANs) that host websites, and the end users who access these websites using browsers.

- **Webmaster's perspective:** From a webmaster's perspective, the greatest security concern is that a web server can expose the LAN or corporate intranet to threats posed by the Internet. These threats may be in the form of viruses, Trojans, attackers, or the compromise of data. Bugs in software programs are often sources of security lapses. Web servers, which are large and complex devices, also have these inherent risks. In addition, the open architecture of web servers allows arbitrary scripts to run on the server side while responding to remote requests. Any Common Gateway Interface (CGI) script installed in the web server may contain bugs that are potential security holes.

- **Network administrator's perspective:** From a network administrator's perspective, a poorly configured web server causes potential holes in the LAN's security. While the objective of the web server is to provide controlled access to the network, excess control can make the web almost impossible to use. In an intranet environment, the network administrator must configure the web server carefully so that legitimate users are recognized and authenticated, and groups of users are assigned distinct access privileges.
- **End user's perspective:** Usually, the end user does not perceive any immediate threat because surfing the web appears safe and anonymous. However, advanced active content such as JavaScript and Web Assembly can introduce risks by allowing malicious code to run in the browser. Harmful applications such as malware and ransomware can exploit these technologies to infiltrate user systems. Moreover, active content from a website can act as a pathway for malicious software to bypass traditional security measures such as firewalls and gain access to the local network (LAN).

The following are some oversights that can compromise a web server:

- Improper file and directory permissions
- Installing the server with default settings
- Unnecessary services enabled, including content management and remote administration
- Security conflicts with the business' ease-of-use requirements
- Lack of proper security policy, procedures, and maintenance
- Improper authentication with external systems
- Default accounts with default or no passwords
- Unnecessary default, backup, or sample files
- Misconfigurations in the web server, OS, and networks
- Bugs in server software, OS, and web applications
- Misconfigured Secure Sockets Layer (SSL) certificates and encryption settings
- Administrative or debugging functions that are enabled or accessible on web servers
- Use of self-signed certificates and default certificates
- Not using dedicated server for web services
- Granting excessive privileges to users or processes, or failing to implement the principle of least privilege.

Apache Web Server Architecture

The Apache web server is an open-source HTTP server used to deliver web content over the Internet. It is widely used owing to its robustness, flexibility, and support for various web technologies. Apache can host websites, handle HTTP requests, and serve both static and dynamic content. It benefits users by providing high performance, security features, extensive customization through modules, and a large support community. These attributes render Apache a popular choice for web hosting and development.

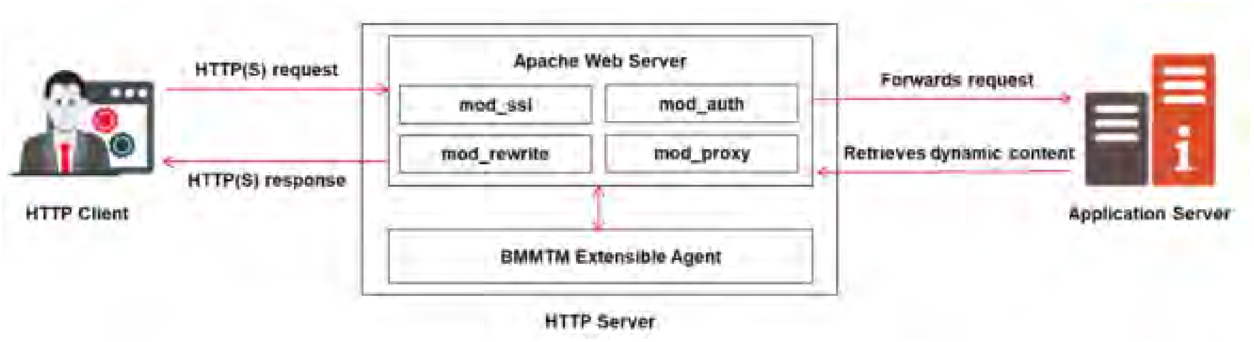


Figure 13-03: Illustration of Apache Web Server Architecture

The functionalities of the various components of the Apache webserver architecture are discussed below:

- **HTTP Client:** It is a browser or software that initiates requests to the Apache server, asking for web pages, files, or other resources.
- **HTTP Server (Core):** The core module handles HTTP(S) requests and responses, interfacing with modules such as mod_ssl, mod_rewrite, mod_proxy, and mod_auth to provide additional functionalities. o mod_auth: Manages user authentication, ensuring that only authorized users can access specific web resources based on the configured credentials.
  - **mod_ssl:** Provides SSL/TLS encryption to secure communication between the server and the clients.
  - **mod_rewrite:** Enables URL rewriting, customized URLs, and redirection based on specified rules.
  - **mod_proxy:** Functions as a proxy and gateway, enabling the forwarding of requests to other servers and load balancing.
- **BMMTM Extensible Agent:** Intercepts HTTP(S) requests and responses to gather detailed transaction data. It enhances monitoring and performance analysis by providing insights into the interactions between clients and servers.
- **Application Server:** Executes backend applications, processes data, and generates dynamic content, functioning separately from the web server that handles the HTTP requests. It is used to run applications written in various programming languages (e.g., PHP, Java, and Python), generating dynamic content that is subsequently served by the Apache web server.

Apache Vulnerabilities

The following are some of the common vulnerabilities found in Apache Servers:

Vulnerability	Description
HTTP response splitting	• This vulnerability occurs when improperly validated input allows attackers to inject malicious headers into HTTP responses • Attackers can leverage this to manipulate web traffic, potentially leading to cross-site scripting (XSS), cache poisoning attacks or sensitive information disclosure
HTTP/2 DoS by memory exhaustion on endless continuation frames	• This vulnerability occurs when attackers send continuous HTTP/2 headers, leading to excessive memory consumption and a potential denial of service (DoS) • Attackers take advantage of this by exhausting the server's memory, which results in the server becoming unresponsive or crashing
mod_macro buffer over-read	• This vulnerability occurs when the mod_macro module improperly handles macro expansion, causing it to read beyond the buffer's end • Attackers exploit this by sending specially crafted requests that trigger the over-read, potentially revealing sensitive information stored in adjacent memory locations
DoS in HTTP/2 with initial window size 0	• This vulnerability arises when an attacker sets the HTTP/2 initial window size to 0, which blocks the server from sending data • Attackers exploit this by sending requests that set the window size to 0, causing the server to wait indefinitely for window size updates, leading to a denial of service (DoS)
HTTP/2 stream memory not reclaimed right away on RST	• This vulnerability occurs when memory allocated for an HTTP/2 stream is not immediately freed upon receiving a stream reset (RST) frame • Attackers exploit this by repeatedly creating and resetting streams, causing memory to be consumed without being released, eventually leading to memory exhaustion and a denial of service (DoS)
Insecure default configuration	• This vulnerability arises from insecure default admin credentials, leading to remote code execution (RCE) • Attackers exploit this by using the default credentials to gain admin access and execute arbitrary code
Improper authorization	• This vulnerability arises from improper authorization mechanisms within the server's core components. Improper authorization • Attackers can leverage this by exploiting the faulty authorization checks to gain unauthorized access or escalate their privileges to access and perform restricted actions

Table 13-01: Apache Server Vulnerabilities

IIS Web Server Architecture

Internet Information Services (IIS) is a Windows-based service that provides a request-processing architecture. IIS's latest version is 7.x. The architecture includes Windows Process Activation

Services (WAS), Web Server Engine, and Integrated Request Processing Pipelines. IIS contains multiple components responsible for several functions, such as listening to a request, managing processes, reading configuration files, etc.

Components of IIS

IIS components include:

- ***Protocol Listeners***

  Protocol Listeners are responsible for receiving protocol-specific requests. They forward these requests to IIS for processing and return responses to requestors.

- ***HTTP.sys***

  HTTP listener is implemented as a kernel-mode device driver called the HTTP protocol stack (HTTP.sys). HTTP.sys is responsible for listening to HTTP requests, forwarding these requests to IIS for processing, and then returning processed responses to client browsers.

- ***World Wide Web Publishing Service (WWW Service)***

  For communication using the Hypertext Transfer Protocol (HTTP) on Windows 2000 and Windows NT, the World Wide Web Publishing Service is available.

  Using IIS and this protocol, users can publish Web content for usage on the Internet or on company intranets. Using a common Web browser like Microsoft Internet Explorer, published Web material may then be browsed and shown on client computers.

  Third-party apps that provide remote administration of Windows 2000 and Windows NT systems use Web browsers as their default administrative interface, and the World Wide Web Publishing Service is at the heart of all of these applications.

- ***Windows Process Activation Service (WAS)***

  World Wide Web Publishing Service (WWW Service) handled functionality in the previous version of IIS, whereas in versions 7 and later, WWW Service and WAS Service are used. These services run svchost.exe on the local system and share the same binaries.

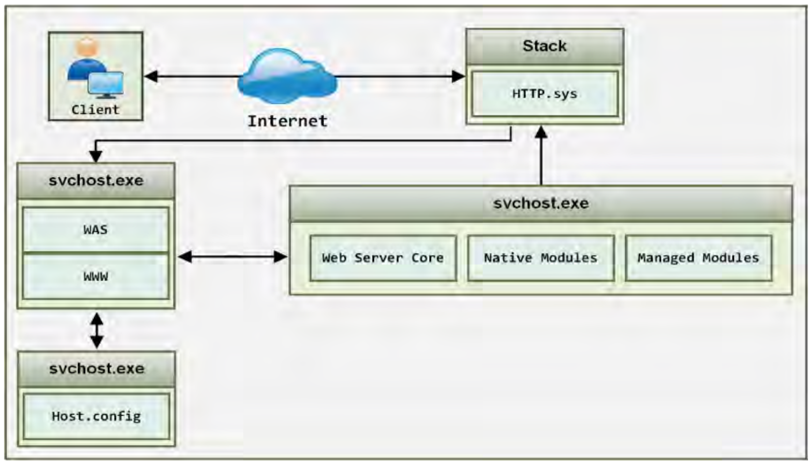


Figure 13-04: IIS Web Server Architecture

IIS Vulnerabilities

The following are some of the vulnerabilities associated with Microsoft IIS servers:

Vulnerability	Description
Trust boundary violation vulnerability	• This vulnerability results from inadequate separation of privilege boundaries, allowing an unauthenticated entity to access restricted functionality in the Telerik Report Server. • Exploiting this flaw may lead to unauthorized server operations manipulation.
Authentication bypass vulnerability	• It occurs due to specific issues in the implementation of the authentication process where an insecure endpoint allows unauthenticated access to restricted server functionality. • Attackers can leverage this vulnerability to circumvent authentication and execute arbitrary code on the server.
CRLF cross-site scripting vulnerability	• This vulnerability arises due to certain misconfigurations in the SiteMinder Web Agent for IIS Web Server. • Attackers can execute arbitrary JavaScript code in a client's browser by exploiting this vulnerability.
CCURE passwords exposed to administrators	• This vulnerability arises due to improper handling and logging of sensitive information within the Microsoft Internet Information Server (IIS) while hosting the C•CURE 9000 Web Server. • Attackers can exploit this vulnerability and access these logs to retrieve sensitive Windows credentials.
Arbitrary file path access vulnerability	• This vulnerability occurs due to the default configuration of the Aquaforest TIFF Server, which improperly restricts access to file paths. • Attackers can exploit this vulnerability to access, enumerate, or traverse directories and files, potentially bypassing authentication or accessing restricted files.
Windows IIS server elevation of privilege vulnerability	• This vulnerability occurs due to the server's improper handling of specific user requests in Windows IIS server. • Attackers can leverage this vulnerability to obtain unauthorized access and take control of the system.
File and directory permissions vulnerability	• This vulnerability arises due to incorrect default permissions in the Hitachi JP1/Performance Management software on Windows. • Attackers can leverage this vulnerability to manipulate files and directories unauthorizedly.
TYPO3 cross-site scripting (XSS) vulnerability	• This vulnerability occurs due to unfiltered use of the server environment variable PATH_INFO in the GeneralUtility::getIndpEnv() component of the TYPO3 Content Management Framework. • Attackers exploit this vulnerability by injecting malicious HTML code into uncached pages, potentially affecting other visitors.
MailEnable vulnerability	• This vulnerability occurs due to improper handling of file paths in certain conditions. • This vulnerability allows authenticated mail users to add files with unsanitized content in public folders where the IIS user has permission to access, potentially leading to arbitrary code execution.
XSS in password manager	• This vulnerability occurs due to the improper neutralization of user-controllable input within the /isapi/PasswordManager.dll ResultURL parameter. • Attackers can exploit this vulnerability to inject malicious scripts and steal sensitive information.

Table 13-02: IIS Vulnerabilities

Nginx Web Server Architecture

Nginx is a high-performance scalable web server, reverse proxy, and load balancer that operates on a master-worker architecture. It employs a single-threaded, event-driven, asynchronous, and non-blocking model to efficiently manage multiple connections. The core of Nginx's architecture comprises a master process that oversees various worker processes responsible for handling client requests.

This modular architecture allows extensive customization through various HTTP, streams, mail, and third-party modules. Nginx supports advanced load balancing, robust caching, SSL/TLS termination, and detailed logging. Its security features, including access control, authentication, and rate-limiting, make it suitable for high-traffic and mission-critical applications.

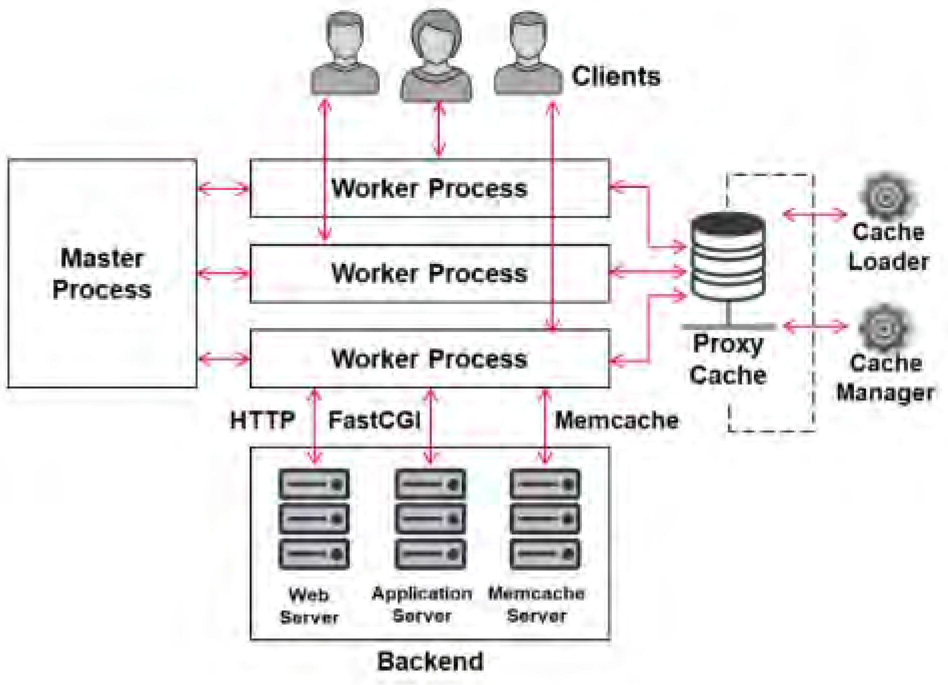


Figure 13-05: Nginx Web Server Architecture

The various components of the Nginx architecture are discussed below:

- **Master Process:** The master process in Nginx is responsible for reading and validating configuration files; creating, binding, and closing sockets; and managing worker processes. It performs administrative tasks and ensures that worker processes are properly configured and run efficiently.
- **Worker Processes:** Nginx worker processes handle client requests by accepting connections, reading/writing data, and communicating with upstream servers. Each single-threaded worker uses non-blocking I/O to handle over 1000 connections simultaneously. After processing the request, a response is sent back to the client. Shared memory zones associated with each worker process enable communication and data sharing.

- **Proxy Cache:** The proxy cache stores copies of requested content, reduces backend server load, and speeds up response times by serving frequently accessed content directly from the cache memory. The Nginx cache quickly renders pages by retrieving them from the cache instead of the server.
  - **Cache Loader:** The cache loader loads cache metadata into memory at Nginx start-up, ensuring that the cache is ready to immediately serve requests. It scans the cache directories and initializes the in-memory cache structures.
  - **Cache Manager:** The cache manager periodically checks the cache for expired content and removes old or unused cache entries into free space, ensuring that the cache does not grow beyond its configured limits.
- **Web Server:** The web server component of Nginx handles HTTP requests sent by clients, serving static content, and forwarding dynamic content requests to the application servers.
- **Application Server:** The application server component processes requests from clients by running server-side scripts or applications and delivers dynamic content to clients.
- **Memcache:** Memcache serves as a caching layer that stores data in memory for the rapid retrieval of frequently accessed data, thereby reducing the need for repeated database queries.

Nginx Vulnerabilities

Some of the common vulnerabilities found in Nginx servers are discussed below:

Vulnerabilities	Description
NULL pointer dereference in HTTP/3	• This vulnerability occurs due to a NULL pointer dereference in Nginx's QUIC module when handling HTTP/3 requests, causing worker processes to terminate. • Attackers can trigger this flaw by sending crafted requests to a vulnerable Nginx server, causing denial of service or potential remote code execution.
Server-side request forgery (SSRF) vulnerability	• This vulnerability occurs due to Server-side request forgery (SSRF) in the upload link feature of mintplex-labs/anything-llm. • Attackers can exploit this vulnerability by hosting a malicious website, allowing them to perform internal port scanning, access non-public web applications, execute arbitrary file deletion, and carry out local file inclusion.
Remote code execution vulnerability	• This vulnerability arises due to the exposed configuration settings via Nginx-UI. • Attackers can exploit this vulnerability to perform remote code execution, privilege escalation, or information disclosure.
Improper certificate validation	• This vulnerability occurs due to the improper validation of user input in the Import Certificate feature of Nginx-UI. • Attackers can exploit this vulnerability to perform arbitrary file writes.
SQL injection vulnerability	• This vulnerability occurs due to improper neutralization of special SQL elements, allowing unsensitized user-controlled parameters to be appended to queries. • Attackers can exploit this vulnerability to execute arbitrary SQL queries for unauthorized access or data breaches.
Unauthenticated private keys access	• This vulnerability occurs due to the reliance on .htaccess for security, which fails on Nginx servers that do not support .htaccess files. • Attackers can exploit this vulnerability to read private keys by accessing the site on a server that does not support .htaccess files.
Excessive memory usage and CPU exhaustion in HTTP/2	• This vulnerability arises from improper memory handling and excessive CPU usage in HTTP/2 requests. • By exploiting this vulnerability, attackers flood the server with HTTP/2 requests to consume server memory and CPU, disrupting normal operations.
OS command injection in nginxWebUI	• This vulnerability occurs due to improper handling of file arguments in the upload feature. • Attackers exploit this vulnerability to manipulate file arguments to inject and execute OS commands remotely on the server.
Default file permissions vulnerability	• This vulnerability occurs because the Nginx Management Suite sets default file permissions that allow authenticated modification of sensitive files. • Attackers leverage this vulnerability to modify sensitive files on the Nginx Instance Manager and Nginx API Connectivity Manager

Table 13-03: Nginx Vulnerabilities

Web Server Attacks

There are several Web Server Attacking techniques, some of which were defined earlier in this workbook. The remaining techniques are defined below:

DNS Server Hijacking

By compromising the DNS server, an attacker modifies the DNS configuration. Modification results in redirecting requests meant for the target webserver to the malicious server owned or controlled by the attacker.

DNS Amplification Attack

A DNS Amplification Attack is performed with the help of the DNS recursive method. An attacker takes advantage of this feature and spoofs the lookup request to the DNS server. The DNS server sends the request to the spoofed address, i.e., the target's address. Amplifying the request size and using botnets result in a distributed denial-of-service attack.

Directory Traversal Attacks

In this type of attack, attackers use the trial and error method to access restricted directories using dots and slash sequences. By accessing the directories outside the root directory, attackers can reveal sensitive information about the system.

Website Defacement

Website Defacement is a process in which attackers, after successful intrusion into a legitimate website, alter, modify, and change the appearance of the website. Accessing and defacing a website can be performed with several techniques, such as SQL injection.

Web Server Misconfiguration

Another method of attack is finding vulnerabilities in a website and exploiting them. An attacker may look for misconfigurations and vulnerabilities in the system and web server components. The attacker may identify weaknesses in the default configuration, remote functioning, misconfigurations, and default certification and then debug to exploit them.

HTTP Response Splitting Attack

HTTP Response Splitting Attacks are techniques in which an attacker sends response-splitting requests to the server. This way, an attacker can add a header response, resulting in the server splitting the response into two. The second response comes under the attacker's control so the user can be redirected to the malicious website.

Web Cache Poisoning Attack

A Web Cache Poisoning Attack is a technique in which an attacker wipes the actual cache of the web server and stores fake entries by sending a crafted request into the cache. This will redirect the users to malicious web pages.

SSH Brute-Force Attack

Brute-Forcing the SSH tunnel allows an attacker to use an encrypted tunnel. This encrypted tunnel is used for communication between hosts. An attacker can gain unauthorized access to the SSH tunnel by brute-forcing the SSH login credentials.

FTP Brute Force with AI

Attackers can leverage AI-powered technologies to enhance and automate brute-force attempts. With the aid of AI, attackers can effortlessly perform brute-force attacks on targets.

For example, an attacker can use ChatGPT 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" The output of this prompt results in the following command:

```
hydra -L/usr/share/wordlists/ftp-usernames.txt -p /usr/share/wordlists/ftp-passwords.txt ftp://10.10.1.11
```

This command will use Hydra to perform a brute-force attack against the FTP server running on the IP address "10.10.1.11", using the usernames and passwords specified in the provided wordlists.

- hydra: This is the command to execute the Hydra tool.
- -L /usr/share/wordlists/ftp-usernames. txt: Specifies the path to the file containing a list of usernames to use for the brute-force attack. The -L flag indicates that the provided file contains a list of usernames.
- -P /usr/share/wordlists/ftp-passwords. txt: Specifies the path to the file containing a list of passwords to use for the brute-force attack. The -P flag indicates that the provided file contains a list of passwords.
- ftp://10.10.1.11 : Specifies the protocol (FTP) and the target IP address (10.10.1.11) where the brute-force attack will be directed.

HTTP/2 Continuation Flood Attack

The HTTP/2 continuation flood attack involves exploiting the handling mechanism of HTTP/2 CONTINUATION frames to exhaust the target Apache server. In HTTP/2, headers that are too large to fit into a single HEADERS frame can be divided into multiple frames. The initial portion of the headers is sent in a HEADERS frame, and the remaining parts are sent in CONTINUATION frames. Attackers exploit this mechanism by sending numerous CONTINUATION frames over a single TCP connection without completing the headers, overwhelming the Apache server's resources, and causing a denial-of-service (DoS) condition on the server.

How HTTP/2 Continuation Flood Attack Works

- The attacker first establishes a TCP connection with the target Apache server. ▪ The attacker then sends a legitimate HEADERS frame. This frame contains headers for a request, and more headers follow in the subsequent CONTINUATION frames.
- Upon receiving the HEADERS frame, the Apache server allocates its memory and resources to process the frame sent by the attacker.
- Instead of completing the header sequence by setting the END_HEADERS flag, the attacker sends several CONTINUATION frames. Each CONTINUATION frame indicates that additional header data are yet to be obtained.
- For each received CONTINUATION frame, the server allocates additional memory and processing resources to handle the incoming header data. The server remained in a state of anticipation and waited for the header sequence to be completed.
- As CONTINUATION frames increase, the server memory and CPU resources become overwhelmed, leading to resource exhaustion.
- The Apache server exhausts its memory or processing capacity, causing it to slow down, crash, or become unresponsive.

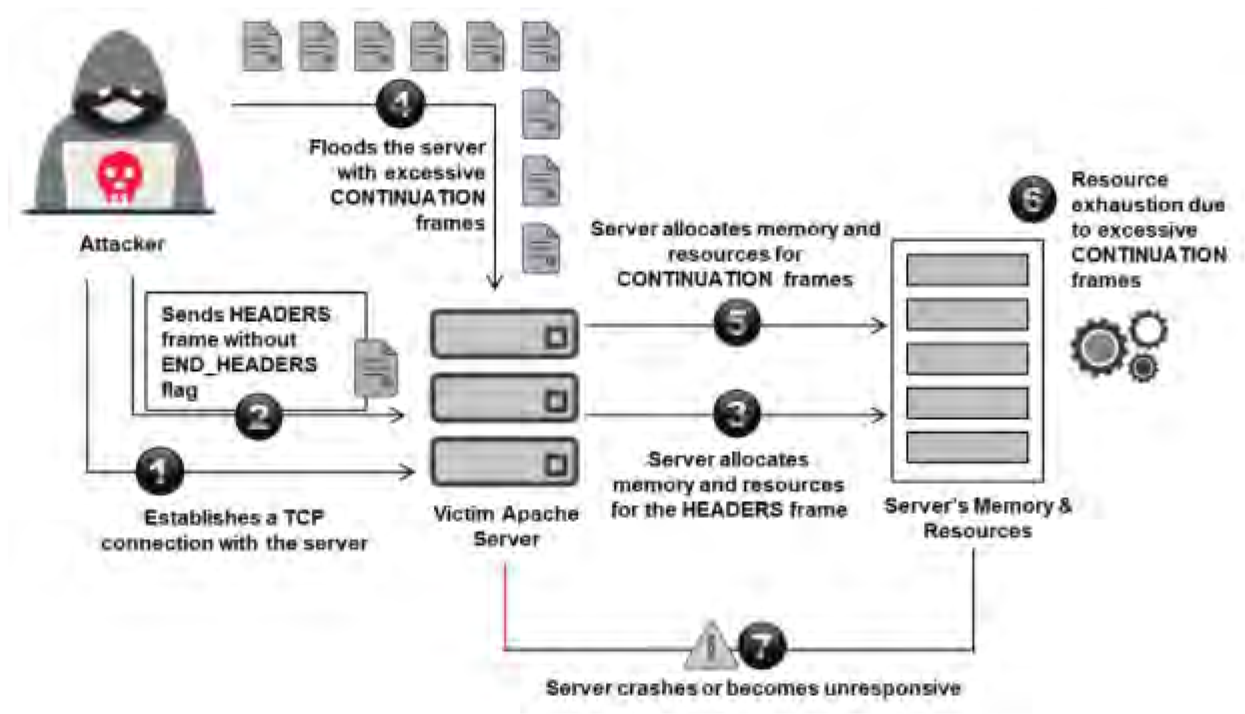

Attacker
4 Floods the server with excessive CONTINUATION frames
Sends HEADERS frame without END_HEADERS flag
2
1 Establishes a TCP connection with the server
Victim Apache Server
Server allocates memory and resources for CONTINUATION frames
5
3
Server allocates memory and resources for the HEADERS frame
6 Resource exhaustion due to excessive CONTINUATION frames
Server's Memory & Resources
7
Server crashes or becomes unresponsive

Figure 13-06: Illustration of HTTP/2 Continuation Flood Attack

Frontjacking Attack

Frontjacking is a type of web server attack in which an attacker injects or manipulates the front-end components of a web application, such as scripts or HTML elements, to hijack a user interface or user interactions. This attack often targets poorly configured Nginx reverse proxy servers in shared hosting environments by combining CRLF injection, HTTP request header injection, and XSS.

Attackers exploit flaws in the target reverse proxy configuration, such as improper sanitization of $uri and $document_uri variables, to inject a new host header. This hijacks the execution flow of the front-end reverse proxy server and consequently replaces the accessed backend server with an attacker-controlled server. This allows attackers to display malicious content, redirect users to fake websites, execute reflected XSS and phishing payloads, and inject harmful scripts.

How A Frontjacking Attack Works

- The attacker creates an HTTP request containing CRLF characters in the URI to inject a malicious host header and sends the request to the vulnerable reverse proxy server.
- The vulnerable reverse proxy accepts a request containing the malicious host header injected via the CRLF injection.
- Once the reverse proxy processes the injected host header, it routes the request to the attacker-controlled server instead of the legitimate backend server.
- The attacker-controlled server responds with malicious content such as phishing pages, malware, or other harmful scripts.

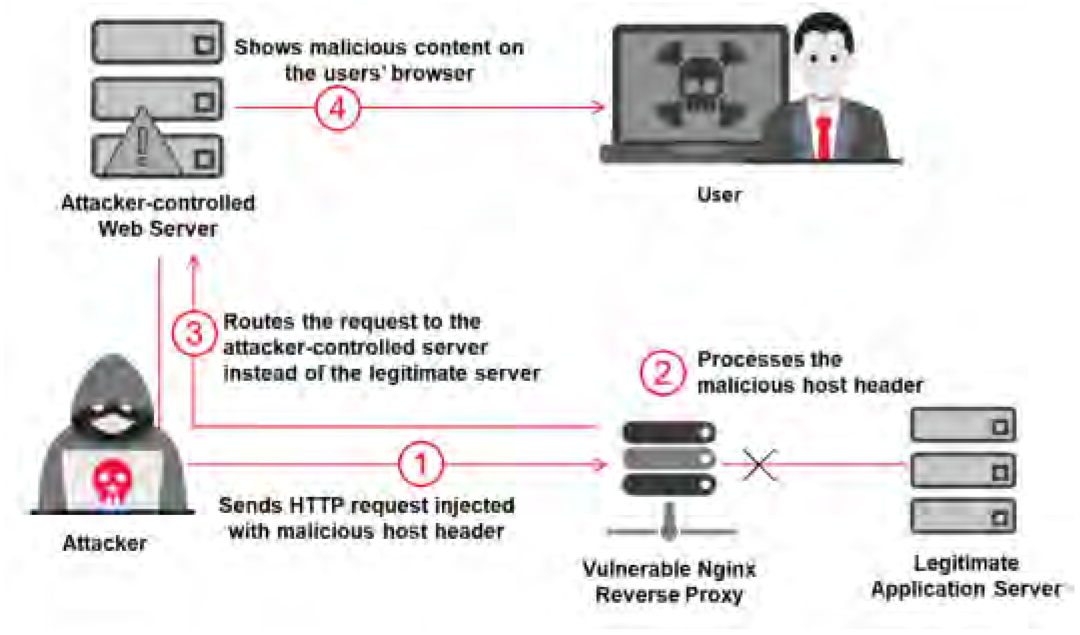

Shows malicious content on the users' browser
4
User
Attacker-controlled Web Server
3
Routes the request to the attacker-controlled server instead of the legitimate server
2
Processes the malicious host header
1
Sends HTTP request injected with malicious host header
Attacker
Vulnerable Nginx Reverse Proxy
Legitimate Application Server

Figure 13-07: Illustration of Frontjacking Attack

Other Web Server Attacks

Web Server Password Cracking

An attacker attempts to exploit weaknesses to hack well-chosen passwords. The most common passwords found are password, root, administrator, admin, demo, test, guest, qwerty, pet names, and so on. The attacker mainly targets the following through web server password cracking:

- SMTP and FTP servers
- Web shares
- SSH tunnels
- Web form authentication

Attackers use different methods such as social engineering, spoofing, phishing, a Trojan horse or virus, wiretapping, and keystroke logging to perform web server password cracking. In many hacking attempts, the attacker starts with password cracking to prove to the web server that they are a valid user.

Web Server Password Cracking Techniques

Password cracking is the most common method of gaining unauthorized access to a web server by exploiting flawed and weak authentication mechanisms. Once the password is cracked, an attacker can use the password to launch further attacks.

We present some details of various tools and techniques used by attackers to crack passwords. Attackers can use password cracking techniques to extract passwords from web servers, FTP servers, SMTP servers, and so on. They can crack passwords either manually or with automated tools such as THC Hydra, and Ncrack. The following are some techniques attackers use to crack passwords:

- **Guessing:** This is the most common method of cracking passwords. In this method, the attacker guesses possible passwords either manually or by using automated tools provided with dictionaries. Most people tend to use their pets' names, loved ones' names, license plate numbers, dates of birth, or other weak passwords such as "QWERTY," "password," "admin," etc. so that they can remember them easily. The attacker exploits this human behavior to crack passwords.
- **Dictionary attack:** A dictionary attack uses a predefined file containing various combinations of words, and an automated program enters these words one at a time to check if any of them are the password. This might not be effective if the password includes special characters and symbols. If the password is a simple word, then it can be found quickly. Compared to a brute-force attack, a dictionary attack is less time-consuming.
- **Brute-force attack:** In the brute-force method, all possible character combinations are tested; for example, the test may include combinations of uppercase characters from A to Z, numbers from 0 to 9, and lowercase characters from a to z. This method is useful for identifying one-word or two-word passwords. If a password consists of uppercase and lowercase letters as well as special characters, it might take months or years to crack the password using a brute-force attack.
- **Hybrid attack:** A hybrid attack is more powerful than the above techniques because it uses both a dictionary attack and brute-force attack. It also uses symbols and numbers. Password cracking is easier with this method than with the above methods.

DoS/DDoS Attacks

DoS and DDoS attack techniques are defined in detail in Chapter 9. These DoS/DDoS attacks are used to flood fake requests toward the web server resulting in crashing, unavailability, or denial of service for all users.

Man-in-the-Middle/Sniffing Attack

As defined in previous chapters, by using a Man-in-the-Middle Attack, an attacker places themself between the client and server and sniffs the packets. They extract sensitive information from the communication by intercepting and altering the packets.

Phishing Attacks

By using Phishing Attacks, an attacker attempts to extract login details from a fake website that appears to be legitimate. The attacker tries to impersonate a legitimate user on the target server using stolen information, usually credentials.

Web Application Attacks

Other web application-related attacks include:

- Cookie Tampering
- DoS Attack
- SQL Injection
- Session Hijacking

- Cross-Site Request Forgery (CSRF) Attack
- Cross-Site Scripting (XSS) Attack
- Buffer Overflow

Web Server Attack Methodology

Information Gathering

Information gathering is the first and one of the most important steps toward hacking a target web server. In this step, an attacker collects as much information as possible about the target server by using various tools and techniques. The information obtained from this step helps the attacker in assessing the security posture of the web server. Attackers may search the Internet, newsgroups, bulletin boards, and so on for gathering information about the target organization. Attackers can use tools such as who.is and Whois Lookup to extract information such as the target's domain name, IP address, and autonomous system number.

who.is

who.is is designed perform a variety of whois lookup functions. It lets the user perform a domain whois search, whois IP lookup, and whois database search for relevant information on domain registration and availability.

Figure 13-08: Screenshot of who.is

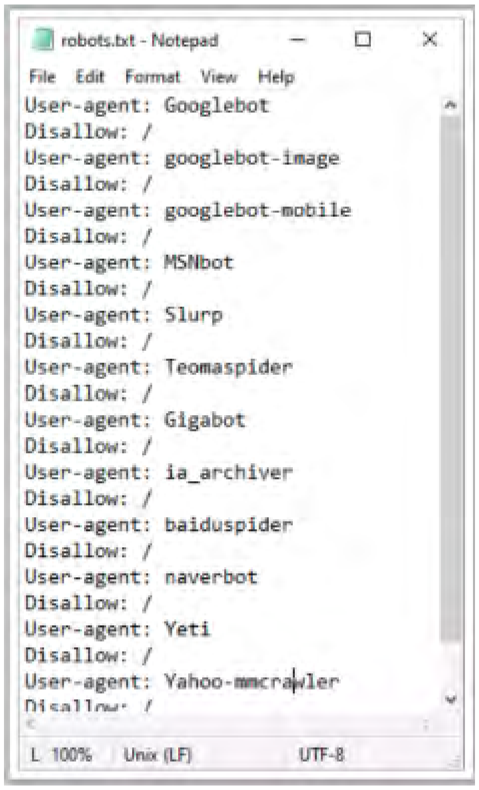
The following are some additional information-gathering tools:

- Whois Lookup (https://whois.domaintools.com)
- Whois (https://www.whois.com)
- Domain Dossier (https://centralops.net)
- Subdomain Finder (https://pentest-tools.com)

Information Gathering from Robots.txt File

A website owner creates a robots.txt file to list the files or directories a web crawler should index for providing search results. Poorly written robots.txt files can cause the complete indexing of website files and directories. If confidential files and directories are indexed, an attacker may easily obtain information such as passwords, email addresses, hidden links, and membership areas.

If the owner of the target website writes the robots.txt file without allowing the indexing of restricted pages for providing search results, an attacker can still view the robots.txt file of the site to discover restricted files and then view them to gather information. An attacker types URL/robots.txt in the address bar of a browser to view the target website's robots.txt file. An attacker can also download the robots.txt file of a target website using the Wget tool.



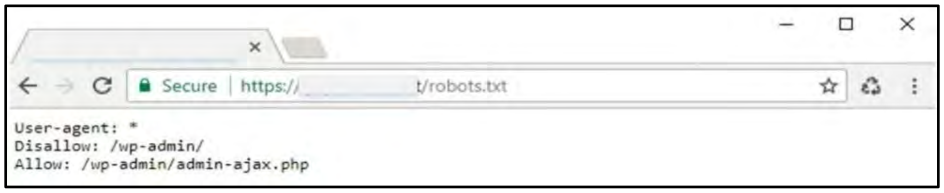
```
User-agent: Googlebot
Disallow: /
User-agent: googlebot-image
Disallow: /
User-agent: googlebot-mobile
Disallow: /
User-agent: MSNbot
Disallow: /
User-agent: Slurp
Disallow: /
User-agent: Teomaspider
Disallow: /
User-agent: Gigabot
Disallow: /
User-agent: ia_archiver
Disallow: /
User-agent: baiduspider
Disallow: /
User-agent: naverbot
Disallow: /
User-agent: Yeti
Disallow: /
User-agent: Yahoo-mmcrawler
Disallow: /
```

Figure 13-09: Screenshot Displaying A Robots.Txt File

Information Gathering/Banner Grabbing

Information gathering is the first and one of the most crucial steps in hacking a target web server. During this phase, an attacker collects as much information as possible about the target server using various tools and techniques. The data obtained in this step is essential for assessing the security posture of the web server. Attackers may search the internet, newsgroups, bulletin boards, and other sources to gather information about the target organization. Tools like who.is and Whois Lookup can be used to extract details such as the target's domain name, IP address, and autonomous system number.

An attacker may use different tools and networking commands to extract information. They may also navigate to the robot.txt file to extract information about internal files.



```
User-agent: *
Disallow: /wp-admin/
Allow: /wp-admin/admin-ajax.php
```

Figure 13-10: Robots.txt File

Web Server Footprinting

This includes footprinting focused on the webserver using different tools such as Netcraft, Maltego, httprecon, etc. The results of web server footprinting can include the server name, type, Operating System, running application, and other information about the target website.

Web Server Footprinting Tools

The following are some additional footprinting tools:

- Netcraft (https://www.netcraft.com)
- ID Serve (https://www.grc.com)
- Nmap (https://nmap.org)
- Ghost Eye (https://github.com)
- Skipfish (https://code.google.com)

Netcat

Netcat (https://netcat.sourceforge.net) is a networking utility that reads and writes data across network connections by using the TCP/IP protocol. It is a reliable "back-end" tool used directly or driven by other programs and scripts. It is also a network debugging and exploration tool.

The following are the commands used to perform banner grabbing for www.moviescope.com as an example to gather information such as server type and version

- # nc –vv www. moviescope.com 80 – press [Enter]
- GET / HTTP/1.0 - press [Enter] twice

Telnet

Telnet is a client–server network protocol that is widely used on the Internet or LANs. It provides login sessions for a user on the Internet. A single terminal attached to another computer emulates the session by using Telnet. The primary security issues with Telnet are the following.

- It does not encrypt data sent through the connection.
- It lacks an authentication scheme.

Telnet enables an attacker to perform a banner-grabbing attack. It probes HTTP servers to determine the server field in the HTTP response header.

For instance, the following procedure is utilized to enumerate a host running on HTTP (TCP 80).

- Request Telnet to connect to a host on a specific port with the command # telnet www.moviescope.com 80 and press Enter. A blank screen appears.
- Type GET / HTTP/1.0 and press Enter twice.

httprecon

httprecon is a tool for advanced web server fingerprinting. This tool performs banner-grabbing attacks, status code enumeration, and header ordering analysis on the target web server and provides accurate web server fingerprinting information.

httprecon performs the following header analysis test cases on the target web server:

- A legitimate GET request for an existing resource
- An exceedingly long GET request (a Uniform Resource Identifier (URI) of >1024 bytes)
- A common GET request for a non-existing resource
- A common HEAD request for an existing resource
- Enumeration with OPTIONS, which is allowed
- The HTTP method DELETE, which is usually not permitted
- The HTTP method TEST, which is not defined
- The protocol version HTTP/9.8, which does not exist
- A GET request including attack patterns (e.g., : ../ and %%)

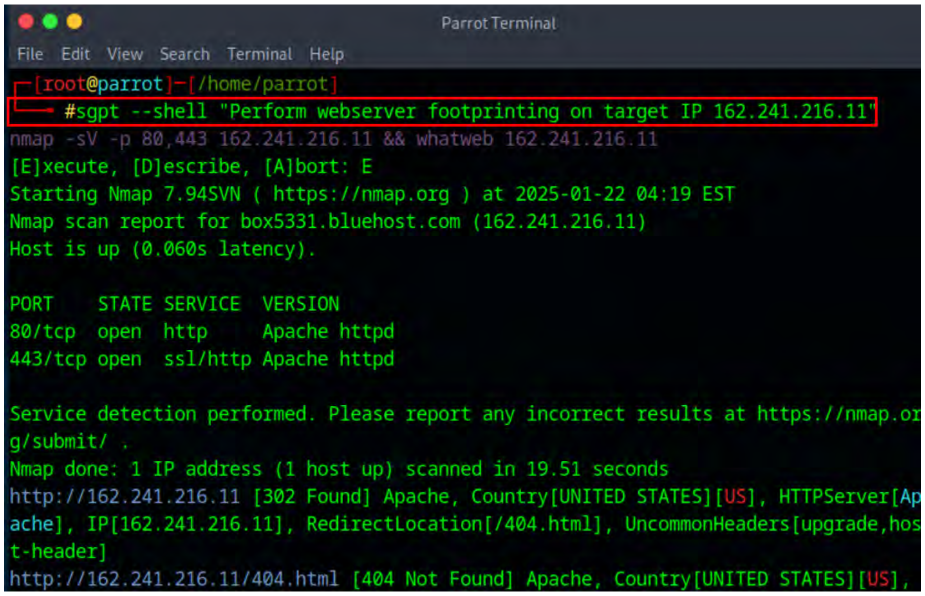
Uniscan

Uniscan is a versatile server fingerprinting tool that not only performs simple commands such as ping, traceroute, and nslookup, but also conducts static, dynamic, and stress checks on web servers. In addition to scanning websites, Uniscan performs automated Bing and Google searches for specific IPs. It compiles all this data into a comprehensive report file.

Web Server Footprinting with AI

Attackers can leverage AI-powered technologies to enhance and automate hacking. With the aid of AI, attackers can effortlessly perform web server footprinting on the target servers.

For example, an attacker can use ChatGPT to perform this task by using an appropriate prompt such as: “Perform webserver footprinting on target IP 162.241.216.11”



```
┌─[root@parrot]─[/home/parrot]
└──╼ #sgpt --shell "Perform webserver footprinting on target IP 162.241.216.11"
nmap -sV -p 80,443 162.241.216.11 && whatweb 162.241.216.11
[E]xecute, [D]escribe, [A]bort: E
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-22 04:19 EST
Nmap scan report for box5331.bluehost.com (162.241.216.11)
Host is up (0.060s latency).

PORT    STATE SERVICE  VERSION
80/tcp  open  http     Apache httpd
443/tcp open  ssl/http Apache httpd

Service detection performed. Please report any incorrect results at https://nmap.or
g/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.51 seconds
http://162.241.216.11 [302 Found] Apache, Country[UNITED STATES][US], HTTPServer[Ap
ache], IP[162.241.216.11], RedirectLocation[/404.html], UncommonHeaders[upgrade,hos
t-header]
http://162.241.216.11/404.html [404 Not Found] Apache, Country[UNITED STATES][US],
```

Figure 13-11: Web Server Footprinting with AI

The output of the prompt results in the following command.

```
nmap -SV -p 80,443 162.241.216.11 && whatweb 162.241.216.11
```

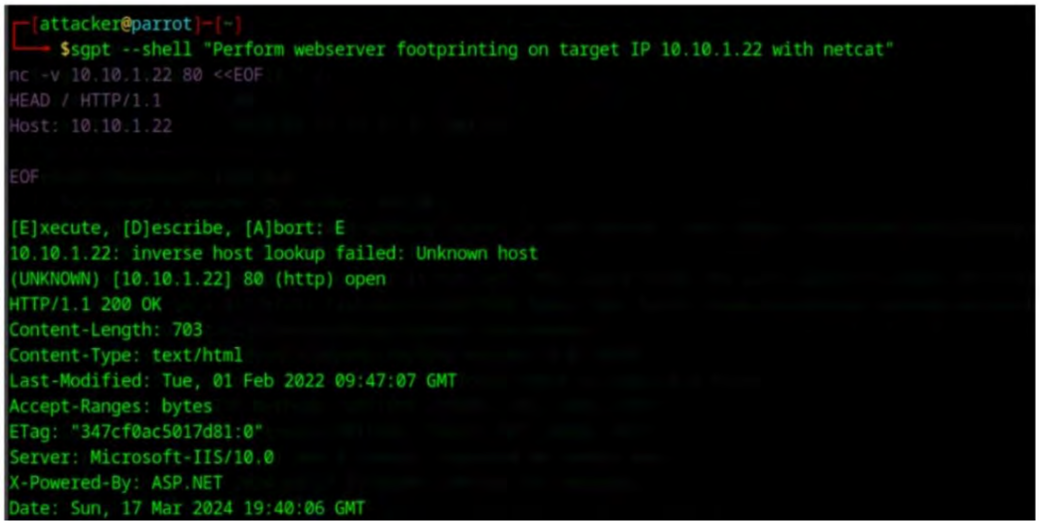
This is a compound command consisting of different network scanning and reconnaissance tools that are Nmap, and WhatWeb.

1. Nmap:
   - -sV: This option enables version detection, which attempts to determine the version of services running on open ports.
   - 162.241.216.11: Specifies the target IP address to scan
     Therefore, the Nmap command scans the target IP address 162.241.216.11 to detect open ports and determine the versions of the services running on those ports
   - -p 80,443 : Scans only ports 80 (HTTP) and 443 (HTTPS)
2. WhatWeb:
   - 162.241.216.11: Specifies the target IP address to scan.
   - WhatWeb is a web scanner that identifies websites built with. By providing the target IP address "162.241.216.11", WhatWeb will attempt to gather information about the web server and technologies used on the target website.

Web Server Footprinting using Netcat with AI

Attackers can leverage AI-powered technologies to enhance and automate hacking. With the aid of AI, attackers can effortlessly perform footprinting on the target Web server with the help of the Netcat tool.

For example, an attacker can use ChatGPT to perform this task by using an appropriate prompt such as: Perform webserver footprinting on target IP 10.10.1.22 with netcat.



```
┌─[attacker@parrot]─[~]
└──╼ $sgpt --shell "Perform webserver footprinting on target IP 10.10.1.22 with netcat"
nc -v 10.10.1.22 80 <<EOF
HEAD / HTTP/1.1
Host: 10.10.1.22

EOF

[E]xecute, [D]escribe, [A]bort: E
10.10.1.22: inverse host lookup failed: Unknown host
(UNKNOWN) [10.10.1.22] 80 (http) open
HTTP/1.1 200 OK
Content-Length: 703
Content-Type: text/html
Last-Modified: Tue, 01 Feb 2022 09:47:07 GMT
Accept-Ranges: bytes
ETag: "347cf0ac5017d81:0"
Server: Microsoft-IIS/10.0
X-Powered-By: ASP.NET
Date: Sun, 17 Mar 2024 19:40:06 GMT
```

Figure 13-12: Web Server Footprinting using Netcat with AI

The command you provided is using 'nc' (netcat) to establish a connection to the target IP address "10.10.1.22" on port 80, and then sending an HTTP request using the 'HEAD' method.

```
nc -v 10.10.1.22 80 <<EOF
HEAD/HTTP/1.1
Host: 10.10.1.22
EOF
```

- nc -v 10.10.1.22 80: This initiates a connection to the target IP address "10.10.1.22" on port 80. The -v option makes 'nc' operate in verbose mode, providing more information about the connection.
- <<EOF: This is a heredoc syntax in Bash, indicating the start of a block of text to be sent to the standard input of 'nc.' Everything between <<EOF and the next occurrence of EOF will be sent over the established connection.
- HEAD/HTTP/1.1: This is the HTTP request line, indicating the HTTP method (HEAD), protocol version (HTTP/1.1), and the path of the resource being requested (which should typically follow the HTTP method, not precede it). This line contains syntactic errors. It should be HEAD / HTTP/1.1 istead of HEAD/HTTP/1.1.
- Host: 10.10.1.22: This is a header indicating the hostname of the server to which the client is attempting to connect. In this case, it is the IP address "10.10.1.22".
- EOF: This signals the end of the heredoc block

IIS Information Gathering using Shodan

Gathering information on IIS servers can provide attackers with crucial details that assist in the planning and execution of further attacks. By identifying the version of IIS used, attackers can match known vulnerabilities to a specific version, making it easier to exploit them. In addition, information such as open ports, running services, and server configurations can reveal weak points in the security posture of the server. With a detailed knowledge of the IIS server environment, attackers can plan their attacks to exploit specific weaknesses. For example, if the gathered data show that a particular server uses an outdated version of IIS, attackers may deploy exploits targeting vulnerabilities specific to that version.

For this purpose, attackers can use tools such as Shodan, which can help them gather the necessary information about IIS servers, thereby significantly enhancing their ability to perform targeted and efficient attacks. Shodan search filters that can be used by attackers to gather information on IIS servers are as follows:

- Use the following filter to search for any IIS server and to provide a list of instances running IIS:
  http.title:"IIS"
- Use the following filter to identify IIS servers with SSL certificates issued to "Company_name" that can assist in finding servers associated with a specific organization:
  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"
  This can assist in identifying servers associated with a specific domain or subdomain.
- Use the following filter to identify IIS servers in the US that can aid in geographically targeted attacks:
  http.title:"IIS Windows Server" country:"US"
- Use the following filter to locate IIS7 servers running on port 80:
  http.title:"IIS7" port:80
- Use the following filter to search for IIS7 servers within a specific IP range to assist in network-specific information gathering:
  http.title:"IIS7" net:"<IP_address>/24"

Additional Shodan search filters that can be used by attackers to gather information on IIS servers are as follows:

- http.title:"IIS7" – Identifies IIS servers running version 7, useful for finding specific version vulnerabilities.
- http.title:"IIS Windows Server" - Targets IIS servers on Windows, helpful for Windows-specific exploitation.
- http.title:"Internet Information Services" - Broad search for any IIS servers, useful for general information gathering.

Attackers can now review the search results to gather information about IIS servers, such as their IP addresses, open ports, running services, and version details. They can also click on individual results to obtain detailed information about the server, including HTTP headers, SSL certificates, and additional metadata. In addition, they can use Shodan's export features to save data in formats, such as CSV or JSON, for further analysis.

Abusing Apache mod_userdir to Enumerate User Accounts

Attackers can exploit Apache's 'mod_userdir' module to enumerate user accounts on a web server. This module allows access to user directories using URIs formatted as '/~username/'. Using the Nmap tool, attackers can identify valid usernames to obtain valuable information that can be used to perform further attacks, such as brute-forcing or targeted phishing.

Enumerating User Accounts from the Apache Server

Perform Initial Scan to Enumerate Valid Users

Run the following command to enumerate valid users from the target web server with mod_userdir enabled:

```
nmap -p80 --script http-userdir-enum <target>
```

This command scans the target on port 80 using the 'http-userdir-enum' script. If 'mod_userdir' is enabled, the script uses the default word list 'usernames.lst' located at '/nselib/data/' and lists any found usernames.

Perform Customized Scan

A customized list of potential usernames can be used to improve the accuracy and effectiveness of the attacks. Run the following command with the custom word list '<Wordlist>.txt':

```
nmap -p80 --script http-userdir-enum userdir.users=<Wordlist>.txt <target>
```

This command takes the '.txt' file as a source for the usernames and tests against the target.

Bypass Detection with Custom User Agent

Some security systems may detect and block requests from Nmap because of its default user-agent string. To bypass such detection, a custom user agent should be specified. Run the following command with the specified HTTP user-agent string to avoid detection:

```
nmap-p80 --script http-brute --script-args http.useragent="<User_Agent>" <target>
```

This command changes the HTTP user-agent string used by Nmap to a specified string, making the traffic appear to originate from a standard web browser rather than a scanning tool.

Enumerating Web Server Information Using Nmap

Nmap (https://nmap.org), along with the Nmap Scripting Engine (NSE), can extract a large amount of valuable information from the target web server. In addition to Nmap commands, NSE provides scripts that reveal various types of useful information about the target server to an attacker.

An attacker uses the following Nmap commands and NSE scripts to extract information.

- Discover virtual domains with hostmap:
  `$nmap --script hostmap-bfk <host>`
- Detect a vulnerable server that uses the TRACE method:
  `nmap --script http-trace -p80 localhost`
- Harvest email accounts with http-google-email:
  `$nmap --script http-google-email <host>`
- Enumerate users with http-userdir-enum:

nmap -p80 --script http-userdir-enum localhost

- Detect HTTP TRACE:
  $nmap -p80 --script http-trace <host>
- Check if the web server is protected by a web application firewall (WAF) or IPS:
  $nmap -p80 --script http-waf-detect --script-args="http-waf-detect.uri=/testphp.vulnweb.com/artists.php,http-waf-detect.detectBodyChanges" www.modsecurity.org
- Fingerprint a WAF
  nmap --script=http-waf-fingerprint -p80,443 <host>
- Enumerate common web applications
  $nmap --script http-enum -p80 <host>
- Obtain robots.txt
  $nmap -p80 --script http-robots.txt <host>

The following are some additional Nmap commands used to extract web server information:

- nmap -sV –O –p target IP address
- nmap -sV --script http-enum target IP address
- nmap target IP address -p 80 --script = http-frontpage-login
- nmap --script http-passwd --script-args http-passwd.root =/ target IP address

Finding Default Credentials of Web Server

Administrators or security personnel use administrative interfaces to securely configure, manage, and monitor web application servers. Many web server administrative interfaces are publicly accessible and located in the root directory. Often, these administrative interface credentials are not properly configured and remain set to default. Attackers attempt to identify the running application interface of the target web server by performing port scanning. Once the running administrative interface is identified, the attacker uses the following techniques to identify the default login credentials:

- Consult the administrative interface documentation and identify the default passwords
- Use Metasploit's built-in database to scan the server
- Use online resources such as cirt.net (https://cirt.net/passwords) and FortyPoundHead.com (https://www.fortypoundhead.com) to identify the default passwords
- Attempt password-guessing and brute-forcing attack

These default credentials can grant access to the administrative interface, compromising the web server and allowing the attacker to exploit the main web application.

cirt.net

cirt.net (https://cirt.net/passwords) is a lookup database for default passwords, credentials, and ports.

The following are some additional websites for finding the default passwords of web server administrative interfaces:

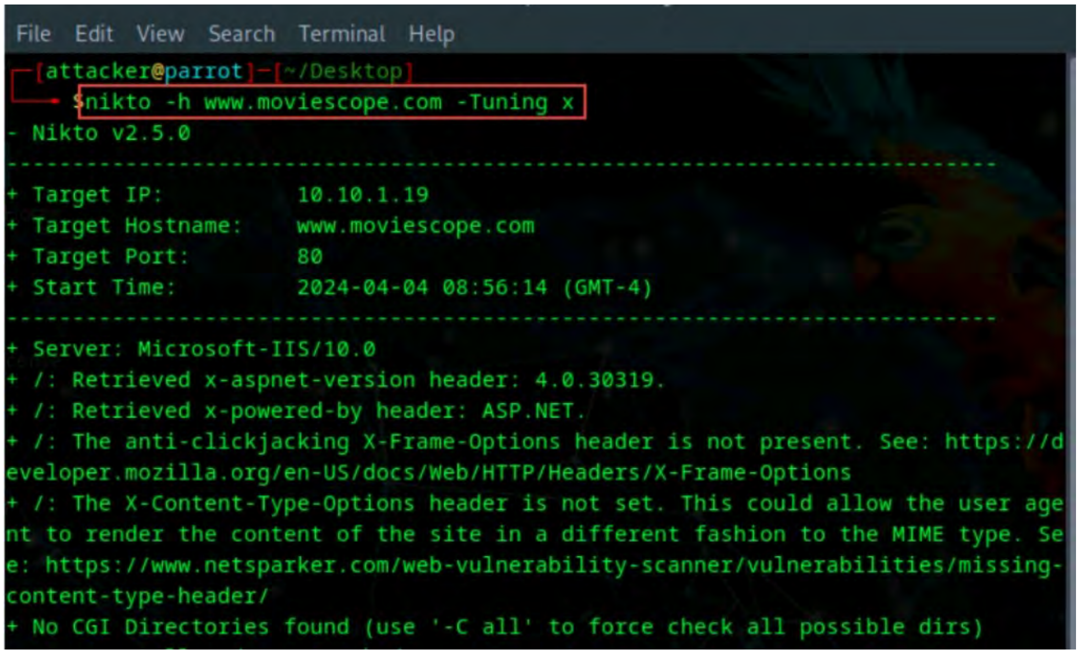
- https://www.fortypoundhead.com
- https://www.defaultpassword.com
- https://default-password.info
- https://www.routerpasswords.com

Finding Default Content of Web Server

Most servers of web applications have default contents and functionalities that allow attackers to launch attacks. Tools such as Nikto2 can be used to identify default contents.

Nikto2:

Nikto is a vulnerability scanner used extensively to identify potential vulnerabilities in web applications and web servers.



```
┌─[attacker@parrot]─[~/Desktop]
└──╼ $nikto -h www.moviescope.com -Tuning x
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          10.10.1.19
+ Target Hostname:    www.moviescope.com
+ Target Port:        80
+ Start Time:         2024-04-04 08:56:14 (GMT-4)
---------------------------------------------------------------------------
+ Server: Microsoft-IIS/10.0
+ /: Retrieved x-aspnet-version header: 4.0.30319.
+ /: Retrieved x-powered-by header: ASP.NET.
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://d
eveloper.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user age
nt to render the content of the site in a different fashion to the MIME type. Se
e: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-
content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
```

Figure 13-13: Screenshot of Nikto2

The following are some common default contents and functionalities that an attacker attempts to identify in web servers.

Administrators Debug and Test Functionality

Functionalities designed for administrators to debug, diagnose, and test web applications and web servers contain useful configuration information and the runtime state of both the server and its running applications. Hence, these functionalities are the main targets for attackers.

Sample Functionality to Demonstrate Common Tasks

Many servers contain various sample scripts and pages designed to demonstrate certain application server functions and application programming interfaces (APIs). Often, web servers fail to secure these scripts from attackers, and these sample scripts either contain vulnerabilities that can be exploited by attackers or implement functionalities that allow attackers to exploit.

Publicly Accessible Powerful Functions

Some web servers include powerful functionalities that are intended for administrative personnel and restricted from public use. However, attackers attempt to exploit such powerful functions to compromise the server and gain access. For example, some application servers allow web archives to be deployed over the same HTTP port as that used by the application. An attacker may use common exploitation frameworks such as Metasploit to perform scanning to identify default passwords, upload backdoors, and gain command-shell access to the target server.

Server Installation Manuals

An attacker attempts to identify server manuals, which may contain useful information about configuration and server installation. Accessing this information allows the attacker to prepare an appropriate framework to exploit the installed web server.

Directory Brute Forcing

When a web server receives a request for a directory, rather than a file, the web server responds to the request in the following ways.

- **Return Default Resource within the directory:** The server may return a default resource within the directory, such as index.html.
- **Return Error:** The server may return an error, such as the HTTP status code 403, indicating that the request is not permitted.
- **Return listing of directory content:** The server may return a listing showing the contents of the directory. A sample directory listing is shown in the screenshot.

Though directory listings do not have significant relevance from a security perspective, they occasionally possess the following vulnerabilities that allow attackers to compromise web applications:

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

In general, after discovering a directory on a web server, an attacker makes a request for that directory and attempts to access the directory listing. Attackers also attempt to exploit vulnerable web server software that grants access to directory listings. Attackers use tools such as Dirhunt and Sitechecker Website Directory Scanner to find directory listings of the target web server.

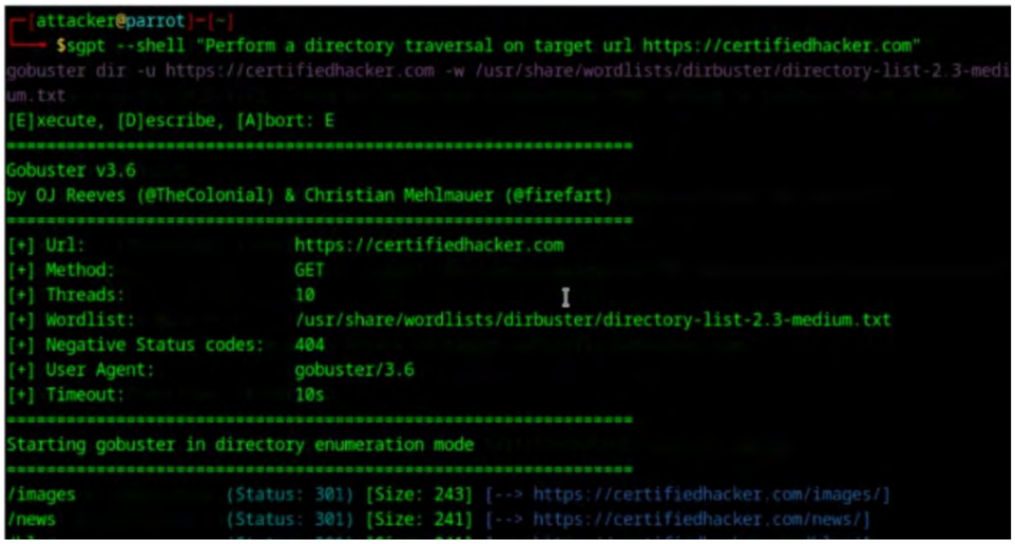
Dirhunt

Dirhunt is a web crawler optimized for searching and analyzing directories. This tool can find interesting results if the server has the "index of" mode enabled. Dirhunt is also useful if the directory listing is not enabled. It detects directories with false 404 errors, directories where an empty index file has been created to hide things, and so on.

Directory Brute Forcing With AI

Attackers can leverage AI-powered technologies to enhance and automate hacking. With the aid of AI, attackers can effortlessly perform directory-traversal attacks on the target domain.

For example, an attacker can use ChatGPT to perform this task by using an appropriate prompt such as: “Perform a directory traversal on target url https://certifiedhacker.com”.



```
┌─[attacker@parrot]─[~]
└──╼ $sgpt --shell "Perform a directory traversal on target url https://certifiedhacker.com"
gobuster dir -u https://certifiedhacker.com -w /usr/share/wordlists/dirbuster/directory-list-2.3-medi
um.txt
[E]xecute, [D]escribe, [A]bort: E
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     https://certifiedhacker.com
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/images               (Status: 301) [Size: 243] [--> https://certifiedhacker.com/images/]
/news                 (Status: 301) [Size: 241] [--> https://certifiedhacker.com/news/]
```

Figure 13-14: Directory Traversal using ChatGPT

The output of prompt results in following command is:

```
gobuster dir -u https://certifiedhacker.com -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

This command is attempting to discover directories on the website.

- gobuster dir: This is the gobuster tool command to perform directory/file brute-forcing
- -u https://certifiedhacker.com: Specifies the target URL which is "https://certifiedhacker.com"
- -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt: Specifies the wordlist to be used for directory and file brute-forcing.

This command systematically tests different directories and files on the target website using the wordlist provided in an attempt to find any that might be accessible.

Vulnerability Scanning

Vulnerability scanning is performed to identify vulnerabilities and misconfigurations in a target web server or network. Vulnerability scanning reveals possible weaknesses in a target server to exploit in a web server attack. In the vulnerability-scanning phase, attackers use sniffing techniques to obtain data on the network traffic to determine active systems, network services, and applications. Automated tools such as Acunetix Web Vulnerability Scanner are used to perform vulnerability scanning on a target server and find hosts, services, and vulnerabilities.

Acunetix Web Vulnerability Scanner

Acunetix (https://www.acunetix.com) Web Vulnerability Scanner (WVS) scans websites and detects vulnerabilities. Acunetix WVS checks web applications for SQL injections, XSS, and so on. It includes advanced pen testing tools to ease manual security audit processes and creates professional security audit and regulatory compliance reports based on AcuSensor Technology. It supports the testing of web forms and password-protected areas, pages with CAPTCHA, single sign-on, and two-factor authentication mechanisms. It detects application languages, web server types, and smartphone-optimized sites. Acunetix crawls and analyzes different types of websites, including HTML5, Simple Object Access Protocol (SOAP), and Asynchronous JavaScript and Extensible Markup Language (AJAX). It supports the scanning of network services running on the server and the port scanning of the web server.

The following are some additional vulnerabilities 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 on Nginx web servers. NginxPwner automates the process of checking for security issues, such as misconfigured HTTP methods, improper file permissions, outdated software versions, and insecure default settings. Using this tool, attackers can quickly gather detailed information on the potential entry points and weaknesses in the Nginx setup, making it easier to launch targeted attacks.

Steps To Scan the Target Nginx Web Server for Vulnerabilities

- Run the following command to create a temporary file to store a list of potential URL paths acquired via crawling or other methods:

  ```
  nano /tmp/pathlist
  ```

  Insert the paths into this file, then save and exit using CTRL+X.

- Execute the nginxpwner script, specifying the target URL and the path file:

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

  Replace <target_URL> and /tmp/<pathlist> with the actual target URL and path to your list file.

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

Finding Exploitable Vulnerabilities

Flaws and programming errors in software design led to security vulnerabilities. Attackers take advantage of these vulnerabilities to perform various attacks on the confidentiality, availability, or integrity of a system. Software vulnerabilities such as programming flaws in a program, service, or within the OS software or kernel can be exploited to execute malicious code.

Many public vulnerability repositories that are available online allow access to information about various software vulnerabilities. Attackers search on exploit sites such as Packet Storm (https://packetstormsecurity.com) and Exploit Database (https://www.exploit-db.com) for exploitable vulnerabilities of a web server based on its OS and software applications. Attackers use the information gathered in the previous stages to find the relevant vulnerabilities by using Exploit Database.

Exploiting these vulnerabilities allows attackers to execute a command or binary on a target machine to gain higher privileges than existing ones or to bypass security mechanisms. Attackers using these exploits can even access privileged user accounts and credentials.

Finding Exploitable Web Server Vulnerabilities Using Exploit Database

Visit Exploit Database Website

Navigate to the Exploit Database (https://www.exploit-db.com) website.

Use the Search Functionality

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").

Filter Search Results

Apply filters to narrow down the search results. You can filter by:

- Type: Choose "Webapps" for web server-related exploits.
- Platform: Select the appropriate platform (e.g., Windows, Linux).
- Date: Specify a date range to find recent vulnerabilities.

Analyze the Results

Review the search results to identify the relevant vulnerabilities. Each result includes details such as the vulnerability description, affected software version, and type of exploit.

Review Exploit Details

Click on an exploit to view detailed information, including:

- Exploit Code: The actual code or script used to exploit the vulnerability.
- Description: Detailed explanation of the vulnerability and how it can be exploited.
- References: Links to additional resources or advisories.

Verify Vulnerability

Cross-check the identified vulnerabilities with the versions and configurations of the target web server to ensure they are applicable.

Download and Test Exploits

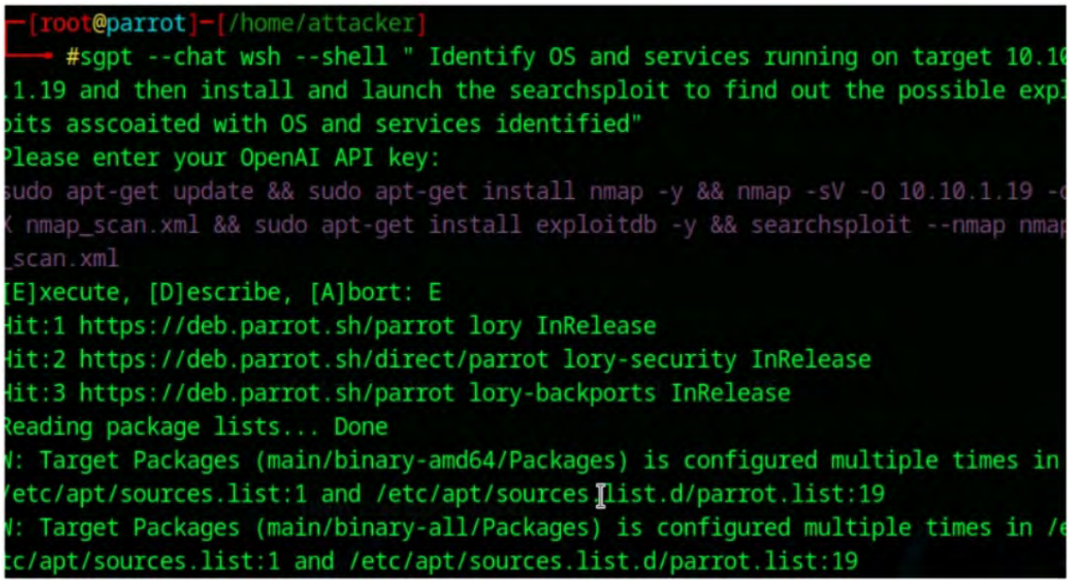
Download the exploit code if it is relevant and tested in the target environment to verify its effectiveness.

Finding Exploitable Vulnerabilities with AI

Attackers can leverage AI-powered technologies to enhance and automate hacking. With the aid of AI, attackers can effortlessly identify potential vulnerabilities and search for exploits associated with them.

For example, an attacker can use ChatGPT 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 10.1
1.19 and then install and launch the searchsploit to find out the possible exp
its asscoaited with OS and services identified"
lease enter your OpenAI API key:
udo apt-get update && sudo apt-get install nmap -y && nmap -sV -O 10.10.1.19 -
 nmap_scan.xml && sudo apt-get install exploitdb -y && searchsploit --nmap nma
scan.xml
E]xecute, [D]escribe, [A]bort: E
it:1 https://deb.parrot.sh/parrot lory InRelease
it:2 https://deb.parrot.sh/direct/parrot lory-security InRelease
it:3 https://deb.parrot.sh/parrot lory-backports InRelease
eading package lists... Done
: 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
: Target Packages (main/binary-all/Packages) is configured multiple times in /
c/apt/sources.list:1 and /etc/apt/sources.list.d/parrot.list:19
```

Figure 13-15: Finding Exploitable Vulnerabilities with AI

The output of the prompt results in the following command:

```
sudo apt-get update && sudo apt-get install nmap -y && nmap -SV-O 10.10.1.19 -oX nmap_scan.xml
&& sudo apt-get install exploitdb -y && searchsploit-nmap nmap_scan.xml
```

- The command first updates the package repository and installs the nmap package to perform a scan on the target IP address 10.10.1.19.

- The nmap command with options -SV-o is used to perform service-version detection without OS detection on the target IP address, and the results are saved in an XML file named nmap_scan.xml.
- Next, the script installs the exploitdb package, which provides the search-sploit utility for searching for possible exploits associated with the identified services.
- The search-sploit-nmap command is used to search for exploits associated with the OS and services identified during the scan.

Session Hijacking

Valid session IDs can be intercepted, allowing an attacker to gain unauthorized access to a web server and view its data. Techniques like session token prediction, session replay, session fixation, sidejacking, and XSS can be used to steal or hijack active session content. These methods help attackers capture session cookies and IDs from established sessions. Tools like Burp Suite, JHijack, and Ettercap can automate the session hijacking process.

Burp Suite

Burp Suite is a web security testing tool capable of hijacking session IDs in active sessions. Its Sequencer tool analyzes the randomness of session tokens, allowing an attacker to predict the next possible session ID and potentially take control of a valid session.

The following are some additional session hijacking tools:

- JHijack (https://sourceforge.net)
- Ettercap (https://www.ettercap-project.org)

Web Server Password Hacking

In this phase of web server hacking, an attacker attempts to crack web server passwords. The attacker may employ all possible techniques of password cracking to extract passwords, including password guessing, dictionary attacks, brute-force attacks, hybrid attacks, precomputed hashes, rule-based attacks, distributed network attacks, and rainbow attacks. The attacker needs patience to crack passwords because some of these techniques are tedious and time-consuming. The attacker can also use automated tools such as Hashcat, THC Hydra, and Ncrack to crack web passwords and hashes.

Hashcat

Hashcat is a cracker compatible with multiple OSs and platforms and can perform multi-hash (MD4, 5; SHA – 224, 256, 384, 512; RIPEMD-160; etc.), multi-device password cracking. The attack modes of this tool are straight, combination, brute force, hybrid dict + mask, and hybrid mask + dict.

THC Hydra

THC Hydra is a parallelized login cracker that can attack numerous protocols. This tool is a proof-of-concept code that provides researchers and security consultants the possibility to demonstrate how easy it would be to gain unauthorized remote access to a system.

Currently, this tool supports the following protocols: Asterisk; Apple Filing Protocol (AFP); Cisco Authentication, Authorization, and Accounting (AAA); Cisco auth; Cisco enable; Concurrent Versions System (CVS); Firebird; FTP; HTTP-FORM-GET; HTTP-FORM-POST; HTTP-GET; HTTP-HEAD; HTTP-POST; HTTP-PROXY; HTTPS-FORM-GET; HTTPS-FORM-POST; HTTPS-GET; HTTPS-HEAD; HTTPS-POST; HTTP-Proxy; ICQ; Internet Message Access Protocol (IMAP); Internet Relay Chat (IRC); Lightweight Directory Access Protocol (LDAP); Memcached; MongoDB; Microsoft SQL Server; MySQL; Network Control Protocol (NCP); Network News Transfer Protocol (NNTP); Oracle Listener; Oracle system identifier (SID); Oracle; PC-Anywhere; personal computer Network File System (PC-NFS); POP3; Postgres; Radmin; Remote Desktop Protocol (RDP); Rexec; Rlogin; Rsh; Real Time Streaming Protocol (RTSP); SAP R/3; Session Initiation Protocol (SIP); Server Message Block (SMB); Simple Mail Transfer Protocol (SMTP); SMTP Enum; Simple Network Management Protocol (SNMP) v1+v2+v3; SOCKS5; SSH (v1 and v2); SSH key; Subversion; TeamSpeak (TS2); Telnet; VMware-Auth; Virtual Network Computing (VNC); and Extensible Messaging and Presence Protocol (XMPP).

The following are some additional password cracking tools:

- Ncrack (https://nmap.org)
- Rainbow crack (https://project-rainbowcrack.com)
- Wfuzz (http://www.edge-security.com)
- Wireshark (https://www.wireshark.org)

Using Application Server as a Proxy

Web servers are occasionally configured to perform functions such as forwarding or reverse HTTP proxy. Web servers with these functions enabled are employed by attackers to perform the following attacks:

- Attacking third-party systems on the Internet
- Connecting to arbitrary hosts on the organization's internal network
- Connecting back to other services running on the proxy host itself

Attackers use GET and CONNECT requests to use vulnerable web servers as proxies to connect to and obtain information from target systems through these web servers.

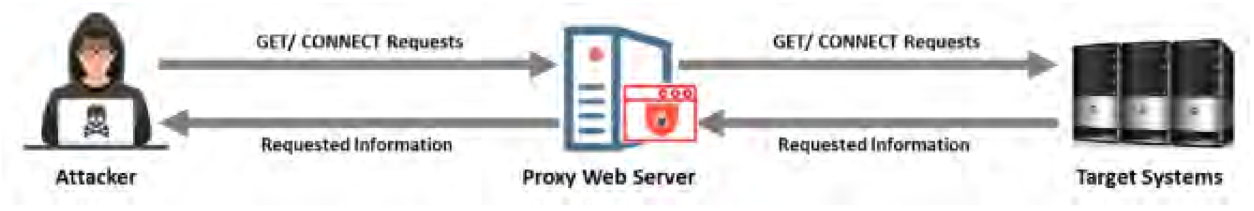


Figure 13-16: Illustration Of The Use Of An Application Server As A Proxy

Path Traversal via Misconfigured NGINX Alias

An attacker can take advantage of a misconfiguration in the nginx.conf file of a targeted Nginx web server, particularly due to improper use of the alias directive. This vulnerability occurs when the alias directive lacks a trailing slash, as shown in the following configuration example:

```
Webroot Misconfiguration in nginx.conf File
location /i
{ alias /data/w3/images/;}
```

In this scenario, the missing trailing slash enables attackers to manipulate the URL to access files and directories outside the designated Webroot. By appending paths like `../`, they can traverse the file system, potentially exposing sensitive data or system files.

Exploiting This Vulnerability Using Kyubi

Kyubi is a Python-based security tool designed to detect and exploit path traversal vulnerabilities in Nginx web servers resulting from misconfigured alias directives. It automates the testing of URL paths to identify potential traversal issues.

Run the following command to scan for misconfigurations/vulnerabilities:

```
kyubi -v <target_URL>
```

Kyubi recursively brute forces the URL path to detect and list files or directories that are potentially exposed owing to misconfigurations.

Exploiting this misconfiguration allows attackers to access sensitive files outside the Webroot, including configuration files and credentials. This can lead to severe consequences such as data theft, website defacement, and even total web server compromise.

Web Server Attack Tools

Immunity's CANVAS

Immunity's CANVAS offers penetration testers and security professionals a robust exploit development framework with hundreds of exploits and an automated exploitation system. Its features include client-side exploitation, privilege escalation, HTTP-tunneled privilege escalation, remote kernel exploitation, advanced backdoor techniques, and sophisticated web attack capabilities.

The following are some additional web server attack tools:

- OpenVAS (https://www.openvas.org)
- THC Hydra (https://github.com)
- HULK DoS (https://github.com)
- MPack (https://sourceforge.net)

Countermeasures

The basic recommendation for securing a web server from internal and external attacks and other threats is to place the webserver in a secure zone where security devices such as firewalls, IPS, and IDS are deployed to filter and inspect the traffic destined for the webserver constantly. Placing the web server in an isolated environment such as a DMZ protects it from threats.

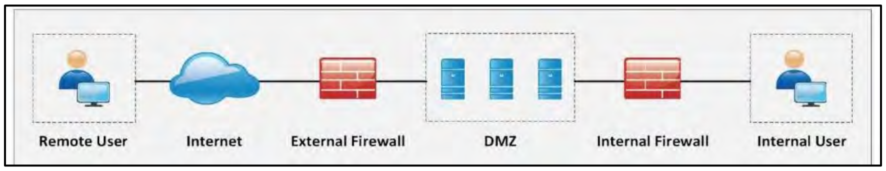


Figure 13-17: Web Server Deployment

Place Web Servers in Separate Secure Server Security Segment on Network

An ideal web hosting network should be designed with three segments: an Internet segment; a secure server security segment, which is often called the demilitarized zone (DMZ); and an internal network. The first step in securing web servers is to place them separately in the DMZ, which is isolated from the public network and from the internal web-hosting network. Placing web servers in a separate segment adds security barriers between the web servers and the internal network as well as between the web servers and the outside public network. This separation allows the administrator to place firewalls and apply access control based on security rules for the internal network as well as for Internet traffic toward the DMZ. Such a web-hosting network can prevent attacks on the web server by outside attackers or malicious insiders.

Network segmentation divides a network into different segments, each having its own hub or switch. It allows network administrators to protect one segment from others by enforcing firewalls and security rules depending on the level of security desired. In a segmented network, an attacker who compromises one segment of the network will not be able to compromise the security of other segments of the network. Let us example a sample web-hosting network that is segmented by the administrator in such a manner that the web server is placed in a DMZ.

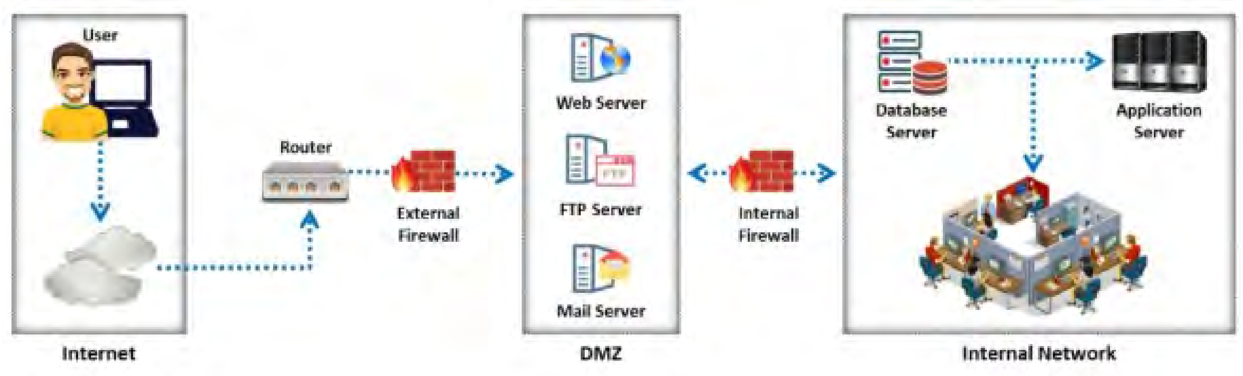


Figure 13-18: Illustration of Three Different Segments in a Web-Hosting Network

Countermeasures: Patches and Updates

The following are various countermeasures for secure update and patch management of web servers:

- Scan for existing vulnerabilities; patch and update the server software regularly.
- Before applying any hotfix, or security patch, read and peer review all relevant documentation.
- Apply all updates, regardless of their type, on an "as-needed" basis.
- Test service packs and hotfixes on a representative non-production environment prior to their deployment in production.
- Ensure that hotfixes, and security patch levels are consistent on all domain controllers (DCs).
- Ensure that server outages are scheduled and that a complete set of backup tapes and emergency repair disks are available.
- Keep a back-out plan that allows the system and enterprise to return to their original state, prior to a failed implementation.
- Disable all unused script extension mappings. ▪ Avoid using default configurations dispatched with web servers.
- Use virtual patches in the organization because they provide additional identification/logging capabilities.
- Establish a disaster recovery plan to handle patch management failures.
- Conduct an extensive risk assessment to determine which segments of the network are most vulnerable or at high risk that need to be patched first.
- Make a detailed inventory of all the endpoints, services, and dependencies.
- While deploying a patch to the full system, ensure that it is performed in testing environment first.
- Deploy an alerting system for patches.
- Use a patch management application or system such as SolarWinds Patch Manager to automate the procedure.
- Regularly conduct monitoring and reporting to ensure your patch management processes and updates are performing effectively.
- Reduce your exposure to third-party risks by limiting the number of software versions you employ.
- All patch and update operations should be validated and documented for accessibility, analysis, and confirmation.
- Make a standardized patch management and security update methodology as part of the SDLC.

Countermeasures: Protocols and Accounts

Countermeasures: Protocols

The following are various countermeasures for using secure protocols on web servers:

- Block all unnecessary ports, ICMP traffic, and unnecessary protocols.
- Harden the TCP/IP stack and consistently apply the latest software patches and updates to system software.
- If insecure protocols such as Telnet, 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.
- Use secure protocols such as Transport Layer Security (TLS)/SSL for communicating with the web server.
- Ensure that unidentified FTP servers operate in an innocuous part of the directory tree that is different from the web server's tree.
- Ensure that the HTTP service banner is properly configured to cover the details of the host device like OS version and type.
- Isolate the supporting servers such as LDAP servers from the local subnet to filter out the traffic through a firewall before entering the local network.'
- Ensure that all the file transfer applications through the web server are done through FTPS for better data encryption and protection.
- Redirect all HTTP traffic to HTTPS to ensure data is encrypted in transit.
- Use HSTS headers to force browsers to use secure connections, preventing downgrade attacks.
- Automate the renewal process for SSL/TLS certificates to avoid the use of expired certificates.
- Implement rate-limiting to mitigate DDoS attacks that target the SSL/TLS handshake process.

Countermeasures: Accounts

The following countermeasures can be adopted to secure user accounts on a web server:

- 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 the 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.
- Run processes using least privileged accounts as well as least privileged services and user accounts.

- Limit the administrator or root-level access to the minimum number of users and maintain a record of the same.
- Maintain logs of all user activity in an encrypted form on the web server or in a separate machine on the intranet.
- Disable all noninteractive accounts that should exist but do not require an interactive login.
- Use secure VPN networks such as OpenVPN while accessing multi-server platforms or accessing data from cross-server network models, which helps use one account for multiple server access.
- Use password managers such as KeePass to maintain a proper password policy for multiple user accounts.
- Enable the Separation of Duties (SoD) feature on the server config settings.
- Force users to periodically change passwords for their accounts by creating a password expiry policy.
- Enable the user account locking feature by setting a limit on the number of failed login attempts.
- Implement 2FA or MFA as an additional layer of security for user accounts.
- Use CAPTCHA challenges on login and registration pages to prevent automated bot attacks.
- Use security questions with unpredictable answers as an additional authentication factor or for account recovery.
- Use strong, one-way hashing algorithms such as bcrypt, scrypt, or Argon2 to securely store passwords.
- Design secure account recovery processes that verify a user's identity without exposing the account to takeover risks.

Countermeasures: Files and Directories

The following countermeasures can be adopted for securing files and directories on a web server:

- Eliminate unnecessary files within .jar files.
- Eliminate sensitive configuration information within the byte code.
- Avoid mapping virtual directories between two different servers or over a network.
- Monitor and check all network services logs, website access logs, database server logs (e.g., Microsoft SQL Server, MySQL, and Oracle), and OS logs frequently.
- Disable the serving of directory listings.
- Eliminate non-web files such as archive files, backup files, text files, and header/include files.
- Disable the serving of certain file types by creating a resource map.
- 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.
- Run the web server within a sandbox directory for preventing access to system files.
- Avoid all non-web file types from being referenced in a URL.

- Run the web server processes with the least required privileges and give access only to the necessary directories.
- Exclude meta characters while processing user inputs to ensure proper filtering of inputs.
- Employ file integrity checkers to verify web content and intrusion detection.
- If an application allows file uploads, the uploaded files are scanned for malware and stored outside the web root.
- Use WAF to protect against common web-based attacks such as SQL injection, which can lead to unauthorized file access.
- Use SFTP instead of FTP to encrypt file transfers.
- Ensure that the configuration files (e.g., .htaccess, web.config) are secure and not accessible from the Web.
- Implement version control for web application files to track changes and revert to previous versions if necessary.

Detecting Web Server Hacking Attempts

An attacker who gains access to a web server by compromising security through known vulnerabilities present in the web server may attempt to plant backdoors (scripts). These backdoors allow the attacker to gain access, launch phishing attacks, or send spam emails. The victim remains unaware of the web server attack until the server is blacklisted on spam mails or until the attacker redirects the visitors of a target site hosted on the web server to some other site. Thus, a web server attack is difficult to detect unless such malicious events occur. By the time these events occur, it may be too late to react because the attacker would have already succeeded. Therefore, a mechanism to detect a web server hacking attempt in its early stages is required to prevent harm to the web server.

When an attacker installs a backdoor on a web server, the size of files infected with the backdoor automatically increases. A website change detection system (WDS) is a script that runs on the server to detect changes made to any executable file or the presence of any new file on the web server, such as HTML, JavaScript (JS), PHP, Active Server Pages (ASP), Perl, and Python files. It works by periodically comparing the hash values of the files on the server with their respective master hash values to detect any changes to the codebase. If it detects any change on the server, it alerts the user to take necessary action. Thus, WDS helps in detecting web server hacking attempts in the early stages of an attack. For example, DirectoryMonitor is an automated tool that goes through entire web folders, detects any changes made to the codebase, and alerts the user through an email.

How to Defend against Web Server Attacks

Defenses against web server attacks include the following.

Ports

Monitor all ports on the web server regularly to prevent unnecessary traffic toward the target web server. If traffic is not monitored, the target web server will be vulnerable to malware attacks. Do not allow public access to port 80 for HTTP or to port 443 for HTTPS; traffic to these ports should be limited. If port 80 is kept open, the server will be vulnerable to DoS attacks, which consume server resources. Intranet traffic should either be encrypted or restricted to secure the web server.

Attackers attempt to hide their identity by spoofing the IP address of a legitimate user. By processing the security log file, either using the "deny this IP address" rule in the firewall ruleset file or by creating a "routed blackhole" command, the target system can defend against web server attacks.

Server Certificates

Server certificates guarantee security and are signed by a trusted authority. However, an attacker may compromise certified servers using forged certificates to intercept secure communications by performing MITM attacks. There are various techniques to avoid such MITM attacks. The following are some of them.

- Use the direct validation of certificates.
- Use a novel protocol that does not depend on third parties for certificate validation.
- Allow domains to directly and securely examine their certificates by using previously established user authentication credentials.
- Use a robust cryptographic construction that enhances server identity validation and resolves the limitations of third-party solutions.
- Ensure that the certificate data ranges are valid and that certificates are used for their intended purpose.
- Ensure that the certificate has not been revoked and that the certificate's public key is valid all the way to a trusted root authority.

Machine.config

The machine.config file provides a mechanism of securing information by changing machine-level settings. It affects all other applications. The machine.config file includes machine settings for the .Net framework, which affect the security. The following can be performed with the machine.config file:

- Ensure that protected resources are mapped to HttpForbiddenHandler and that unused HttpModules are removed
- Ensure that tracing is disabled <trace enable="false"/> and debug compiles are turned off
- Verify that ASP.NET errors are not reverted to the client
- Verify session state settings

Code Access Security

The following measures can be adopted to ensure code access security.

- Implement secure coding practices to avoid source-code disclosure and input validation attacks.
- Restrict code access security policy settings to ensure that there are no permissions to execute code downloaded from the Internet or intranet.

- Configure IIS to reject URLs with "../" to prevent path traversal, lockdown system commands and utilities with restrictive access control lists (ACLs), and install new patches and updates.
- If targets do not implement code access security in their web servers, then there is a possibility of execution of malicious code.

The following are some other measures to defend against web server attacks:

- Apply restricted ACLs and block remote registry administration.
- Secure the SAM (stand-alone servers only).
- Ensure that security-related settings are configured appropriately and that access to the metabase file is restricted with hardened NTFS permissions.
- Remove unnecessary Internet Server Application Programming Interface (ISAPI) filters from the web server.
- Remove all unnecessary file shares, including the default administration shares, if they are not required.
- Secure the shares with restricted NTFS permissions. ▪ Relocate sites and virtual directories to non-system partitions and use IIS web permissions to restrict access.
- 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.
- Enable a minimum level of auditing on the web server and use NTFS permissions to protect log files.
- Use a dedicated machine as a web server.
- Create URL mappings to internal servers cautiously.
- Do not install the IIS server on a domain controller.
- Use server-side session ID tracking and match connections with timestamps, IP addresses, etc.
- If a database server such as Microsoft SQL Server is to be used as a backend database, install it on a separate server.
- Use security tools provided with web server software and scanners that automate and simplify the process of securing a web server.
- Physically protect the web server machine in a secure machine room.
- Do not connect an IIS Server to the Internet until it is fully hardened.
- Do not allow anyone to locally log in to the machine except the administrator.
- Configure a separate anonymous user account for each application if multiple web applications are hosted.
- Limit the server functionality to support only the web technologies to be used.
- Screen and filter incoming traffic requests.
- Implement firewalls to control incoming and outgoing network traffic based on security rules.
- Store website files and scripts on a separate partition or drive.

- Use an effective anti-bot mitigation service such as DataDome to detect botnets in real time.
- Use network segmentation, VPNs, and secure protocols (such as HTTPS) to limit unauthorized access and ensure encrypted communication.
- Implement role-based access control (RBAC) and the principle of least privilege to minimize the risk of unauthorized access.
- Deploy IDPS to monitor network traffic and detect unusual or malicious activity.
- Implement centralized log monitoring to track server and application activities. Analyze logs for signs of suspicious behavior.
- Set up alerts for security events and establish an incident response plan to address security incidents promptly.
- Restrict directory listings and enforce proper file permissions to prevent unauthorized access to sensitive files.
- Turn off unused features, such as server-side scripting or directory browsing, to reduce potential attack vectors.

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

While setting cookies, remove carriage returns (CRs) and linefeeds (LFs) before inserting data into an HTTP response header. The best practice is to use third-party products to test for the existence of security holes and defend against CRLF injection. Ensure that data application engines are up to date.

The User Datagram Protocol (UDP) source port randomization technique defends servers against blind response forgery. Limit the number of simultaneous recursive queries and increase the times-to-live (TTLs) of legitimate records.

The following are some methods to defend against HTTP response-splitting attacks 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 the web application's access to unique IPs
- Disallow CR (%0d or \r) and LF (%0a or \n) characters
- Comply with RFC 2616 specifications for HTTP/1.1
- Parse all user inputs or other forms of encoding before using them in HTTP headers

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

The following techniques can be used to defend against DNS hijacking:

- Choose a registrar accredited by the Internet Corporation for Assigned Names and Numbers (ICANN) and encourage them to set REGISTRAR-LOCK on the domain name.
- Safeguard the registrant's account information.
- Include DNS hijacking in incident response and business continuity planning.
- Use DNS monitoring tools/services to monitor the IP address of the DNS server and set alerts.
- Avoid downloading audio and video codecs and other downloaders from untrusted websites.
- Install an antivirus program and update it regularly.
- Change the default router password.
- Restrict zone transfers and use script blockers in the browser.
- Domain Name System Security Extensions (DNSSEC) adds an extra layer to DNS that prevents it from being hacked.
- Strong password policies and user management further enhances security.
- When signing up for DNS servers with DNS service providers, learn who to contact when an issue occurs, how to receive good-quality reception and support, and whether the DNS server's infrastructure is hardened against attacks.
- Use a master–slave DNS and configure the master without Internet access. Maintain two slave servers so that even if an attacker hacks a slave, it will update only when it receives an update from the master.
- The constant monitoring of DNS servers ensures that a domain name returns the correct IP address.
- Change the default username and password of the router. Keep the firmware up-to-date for ensuring safety from new vulnerabilities.
- VPN service establish virtual private network (VPN)-encrypted tunnels for secure private communication over the Internet. This feature protects messages from eavesdropping and unauthorized access.
- Use firewall protection services to safeguard the original DNS resolvers and filter out the rogue DNS resolver traffic.
- Maintain proper protection systems such as MFA and hardware security to ensure controlled access to the DNS server.
- Install script blocker extensions in the browser.
- Use only secured and reputed VPN networks instead of free VPN services, which can track your activities and record them for future use.
- Use ACLs to restrict who can query DNS servers, thereby reducing the risk of malicious queries leading to hijacking

Web Application Security Scanners

Syhunt Hybrid

The Syhunt Hybrid scanner automates web application security testing and guards the organization's web infrastructure against web application security threats. Syhunt Dynamic crawls websites and detects XSS, directory transversal problems, fault injection, SQL injection, attempts to execute commands, and several other attacks. Syhunt Hybrid creates signatures to detect application vulnerabilities and prevents logout. It analyzes JavaScript (JS), logs suspicious responses, and tests errors for review.

N-Stalker X

N-Stalker is a web application security scanner that searches for vulnerabilities to attacks such as clickjacking, SQL injection, and XSS. It allows spider crawling throughout the application and the creation of web macros for form authentication. It also provides proxy capabilities for "drive-thru" attacks and identifies components through reverse proxies that distribute different platforms in the same application URL.

The following are some additional web application security scanners:

- Invicti (https://www.invicti.com)
- Burp Suite (https://www.portswigger.net)
- Wapiti (https://wapiti-scanner.github.io)
- WebScarab (https://owasp.org)
- WPSec (https://wpsec.com)
- Tinfoil Web Scanner (https://www.tinfoilsecurity.com)
- Skipfish (https://code.google.com)
- Detectify (https://detectify.com)
- OpenTextTM FortifyTM On Demand (https://www.opentext.com)
- OWASP Zed Attack Proxy (ZAP) (https://www.zaproxy.org)
- SonarQube (https://www.sonarqube.org)
- Arachni (https://ecsypno.com)

Web Server Security Scanners

Qualys Community Edition

Qualys Community Edition (https://www.qualys.com)discovers IT assets, manages vulnerabilities, scans web applications, and maintains cloud assets inventory. It offers vulnerability management to identify dangerous bugs and remediate them immediately. Qualys can also assess vulnerabilities on all internal IT infrastructure as well as external-facing assets to ensure a secure state.

The following are some additional web server security scanners:

- Observatory (https://observatory.mozilla.org)
- WordPress Security Scan (https://hackertarget.com)
- Web Vulnerability Scanner (https://pentest-tools.com)
- Nikto (https://cirt.net)

- ImmuniWeb (https://www.immuniweb.com)

Web Server Malware Infection Monitoring Tools

QualysGuard Malware Detection

QualysGuard Malware Detection (https://www.qualys.com)allows organizations to proactively scan their websites for malware and provides automated alerts and in-depth reporting to enable prompt identification and resolution. It enables organizations to protect their customers from malware infections and safeguard their brand reputation.

The following are some additional web server malware infection monitoring tools:

- Sucuri Site Check (https://sitecheck.sucuri.net)
- SiteLock Malware Removal (https://www.sitelock.com)
- Quttera (https://quttera.com)
- Web Inspector (https://www.webinspector.com)
- SiteGuarding (https://www.siteguarding.com)

Web Server Security Tools

OpenText Fortify WebInspect

OpenText Fortify WebInspect (https://www.opentext.com)is an automated dynamic application security testing solution that discovers configuration issues as well as identifies and prioritizes security vulnerabilities in running applications. It mimics real-world hacking techniques and provides a comprehensive dynamic analysis of complex web applications and services. WebInspect dashboards and reports provide organizations with visibility and an accurate risk posture of its applications.

The following are some additional web server security tools:

- 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(https://www.coresecurity.com) finds vulnerabilities in an organization's web server. This tool allows a user to evaluate the security posture of a web server by using the same techniques currently employed by cyber criminals. It scans for possible vulnerabilities in the web server, imports scan results, and runs exploits to test the identified vulnerabilities. It can also scan network servers, workstations, firewalls, routers, and various applications for vulnerabilities; identify which vulerabilities pose real threats to the network; determine the potential impact of exploited vulnerabilities; and prioritize and execute remediation efforts.

The following are some additional web server pen testing tools:

- Cobalt Strike (https://www.cobaltstrike.com)
- Fuxploider (https://github.com)
- Mitmprox (https://mitmproxy.org)

Patch Management

Developers always attempt to find bugs in a web server and fix them. Bug fixes are distributed in the form of patches, which provide protection against known vulnerabilities. Unpatched or vulnerable patches can create a security loophole in the web server. This section describes the role of patches, upgrades, and hotfixes in securing web servers. This section also provides guidance for choosing proper patches, upgrades, hotfixes, and their appropriate sources for secure patch management.

Patches and Hotfixes

A patch is a small piece of software designed to fix problems, security vulnerabilities, and bugs as well as improve the usability or performance of a computer program or its supporting data. A patch can be considered a repair job for a programming problem. A software vulnerability is the weakness of a software program that makes it susceptible to malware attacks. Software vendors provide patches that prevent exploitations and reduce the probability of threats exploiting a specific vulnerability. Patches include fixes and updates for multiple known bugs or issues. A patch is a publicly released update that is available for all customers. A system without patches is much more vulnerable to attacks than a regularly patched system. If an attacker can dentify a vulnerability before it is fixed, then the system might be susceptible to malware attacks.

A hotfix is a package used to address a critical defect in a live environment and contains a fix for a single issue. It updates a specific product version. Hotfixes provide quick solutions and ensure that the issues are resolved. Apply hotfixes to software patches on production systems.

Vendors update users about the latest hotfixes through email or make them available on their official website. Hotfixes are updates that fix a specific customer issue and are not always distributed outside the customer organization. Vendors occasionally deliver hotfixes as a set of fixes called a combined hotfix or service pack.

What is Patch Management?

Patch management is an area of systems management that involves acquiring, testing, and installing multiple patches (code changes) in an administered computer system. Patch management is a method of defense against vulnerabilities that cause security weaknesses or corrupt data. It is a process of scanning for network vulnerabilities, detecting missed security patches and hotfixes, and then deploying the relevant patches as soon as they are available to secure the network. It involves the following tasks:

- Choosing, verifying, testing, and applying patches
- Updating previously applied patches with current patches
- Listing patches applied previously to the current software
- Recording repositories or depots of patches for easy selection
- Assigning and deploying the applied patches

An automated patch management process includes the following steps. ‘

- **Detect:** Use tools to detect missing security patches.
- **Assess:** Asses the issue(s) and its associated severity by mitigating the factors that may influence the decision.
- **Acquire:** Download the patch for testing.
- **Test:** Install the patch first on a test machine to verify the consequences of the update.
- **Deploy:** Deploy the patch to computers and ensure that applications are not affected.
- **Maintain:** Subscribe to receive notifications about vulnerabilities when they are reported.

Installation of a Patch

The installation of a patch entails the following tasks.

Identifying Appropriate Sources For Updates And Patches

It is important to identify appropriate sources for updates and patches. Patches and updates that are not installed from trusted sources can render the target server even more vulnerable to attacks, instead of hardening its security. Thus, the selection of appropriate sources for updates and patches plays a vital role in securing web servers.

The following are some methods for identifying appropriate sources for updates and patches.

- Create a patch management plan that fits the operational environment and business objectives
- Find appropriate updates and patches on the home sites of the applications or OS vendors
- The recommended method of tracking issues relevant to proactive patching is to register to the home sites to receive alerts

Installation of a Patch

Users can access and install security patches via the World Wide Web. Patches can be installed in two ways.

- **Manual Installation:** In this method, the user downloads the patch from the vendor and installs it
- **Automatic Installation**: In this method, applications use an auto update feature to update themselves

Implementation and Verification of a Security Patch or Upgrade

- Before installing any patch, verify the source

- Use a proper patch management program to validate file versions and checksums before deploying security patches
- The patch management tool must be able to monitor the patched systems
- The patch management team should check for updates and patches regularly

Patch Management Best Practices

- Define roles, responsibilities, and procedures for patch management, including timelines for applying patches and handling emergencies
- Develop a comprehensive policy outlining procedures for evaluating, testing, approving, and deploying patches
- Outline systems, applications, and devices, including servers, workstations, and networking equipment, require patching
- Maintain an updated inventory of all hardware and software assets to ensure that no device or application is overlooked during the patching process
- Assess and prioritize patches based on the severity of the vulnerabilities they address
- Group assets by criticality, application type, or other relevant criteria for prioritizing patching efforts
- Use tools to automate the discovery of applicable patches in the systems and ensure timely updates
- Implement a testing process to verify that patches do not cause issues in existing systems
- Leverage patch management software to streamline the patching process, from discovery to deployment
- Create and follow a regular schedule for patching to ensure updates are applied in a timely manner
- Create a procedure for fast-tracking the deployment of patches for critical vulnerabilities that are being actively exploited
- Stay informed about new vulnerabilities and available patches by subscribing to security advisories and feeds
- Limit access to patch management tools and systems to authorized personnel
- Ensure that systems are backed up before applying patches to facilitate recovery in case of issues
- Ensure that patch management is part of standard IT operations
- Verify that patches have been successfully applied after deployment and are functioning as intended
- Use tools or systems to track patch status to ensure that all assets are appropriately patched
- Regularly perform vulnerability scanning to identify unpatched vulnerabilities and verify patch effectiveness
- Protect the patch management system itself against attacks to prevent compromise
- Regularly review and refine patch management processes to improve efficiency
- Include third-party applications patches in the organization's patch management strategy
- Have a plan for quickly rolling back patches in case they cause unforeseen issues

- Use a controlled test environment to verify patches before deploying them to production systems
- Ensure patches do not cause compatibility issues with existing applications and configurations
- Implement scheduled patching to minimize disruption and align with maintenance windows

Patch Management Tools

GFI LanGuard

The GFI LanGuard patch management software (https://www.gfi.com) scans the user's network automatically as well as installs and manages security and non-security patches. It supports machines across Microsoft®, MAC OS X®, and Linux® operating systems, as well as many third-party applications. It allows auto-downloads of missing patches as well as patch rollback, resulting in a consistently configured environment that is protected from threats and vulnerabilities.

The following are some additional patch management tools:

- Symantec Client Management Suite (https://www.broadcom.com)
- Solarwinds Patch Manager (https://www.solarwinds.com)
- Kaseya Patch Management (https://www.kaseya.com)
- Software Vulnerability Manager (https://www.flexera.com)
- Ivanti Patch for Endpoint Manager (https://www.ivanti.com)

Summary

Web servers are essential to modern digital infrastructure, making them attractive targets for cyberattacks. Threat actors exploit vulnerabilities resulting from misconfigurations, outdated software, and inherent security flaws in popular web servers such as Apache, IIS, and Nginx. Common attack techniques include directory traversal, which allows unauthorized access to restricted files, brute force attacks to crack login credentials, and session hijacking to take control of active user sessions. Attackers utilize tools like Nmap, Nikto, and Burp Suite to identify and exploit these weaknesses. The consequences of web server attacks can be severe, leading to data breaches, website defacement, denial-of-service (DoS) attacks, and significant reputational harm. To mitigate these threats, organizations implement security measures such as firewalls, intrusion detection and prevention systems (IDS/IPS), secure configurations, timely patching, and encryption. Additionally, advanced strategies like network segmentation, strong access controls, and web application firewalls (WAFs) provide an extra layer of protection against web server attacks.

Mind Map

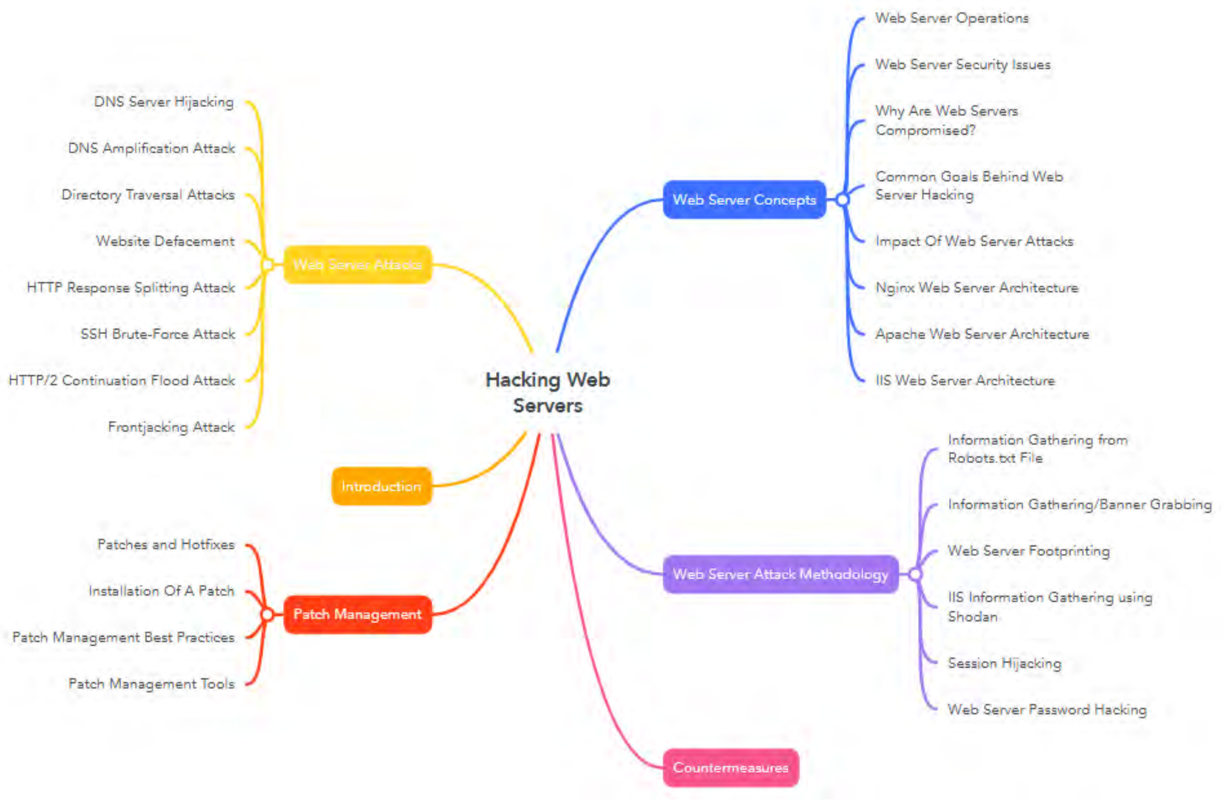

Hacking Web Servers
Web Server Attacks
DNS Server Hijacking
DNS Amplification Attack
Directory Traversal Attacks
Website Defacement
HTTP Response Splitting Attack
SSH Brute-Force Attack
HTTP/2 Continuation Flood Attack
Frontjacking Attack
Introduction
Patch Management
Patches and Hotfixes
Installation Of A Patch
Patch Management Best Practices
Patch Management Tools
Web Server Concepts
Web Server Operations
Web Server Security Issues
Why Are Web Servers Compromised?
Common Goals Behind Web Server Hacking
Impact Of Web Server Attacks
Nginx Web Server Architecture
Apache Web Server Architecture
IIS Web Server Architecture
Web Server Attack Methodology
Information Gathering from Robots.txt File
Information Gathering/Banner Grabbing
Web Server Footprinting
IIS Information Gathering using Shodan
Session Hijacking
Web Server Password Hacking
Countermeasures

Figure 13-19: Mind Map

Practice Questions

1. What is the primary role of a web server?

A. Manage DNS requests
B. Host and deliver web content over HTTP
C. Authenticate users for application access
D. Encrypt database records

2. Which web server software is known for its open-source nature and flexibility?

A. Microsoft IIS
B. Nginx
C. Apache
D. Google Web Server

3. What is the main cause of vulnerabilities in web servers?

A. Using open-source software
B. Configuration errors and outdated software

C. Poor website design
D. High server load

4. What is website defacement?

A. Encrypting website files for a ransom
B. Replacing website content with unauthorized content
C. Deleting website data permanently
D. Using a website to perform distributed denial of service attacks

5. What vulnerability arises from using default admin credentials?

A. Directory traversal
B. Buffer overflow
C. Insecure default configuration
D. SQL injection

6. Which protocol is commonly targeted in web server brute-force attacks?

A. FTP
B. ICMP
C. UDP
D. DNS

7. What is a botnet's role in web server attacks?

A. Hosting legitimate websites
B. Performing DDoS attacks
C. Encrypting web server data
D. Managing user authentication

8. What does a directory traversal attack involve?

A. Injecting SQL commands
B. Accessing files outside the root directory
C. Spoofing DNS requests
D. Cracking passwords

9. Which tool is commonly used for web server footprinting?

A. Nessus
B. Nmap
C. Metasploit
D. Hydra

10. Which type of vulnerability allows attackers to execute JavaScript on a client's browser?

A. SQL injection

B. Directory traversal
C. Cross-site scripting (XSS)
D. Session fixation

11. What is the main objective of HTTP response splitting attacks?

A. Crashing the server
B. Injecting malicious headers into HTTP responses
C. Bypassing authentication mechanisms
D. Manipulating DNS records

12. What is the purpose of mod_rewrite in the Apache server?

A. Manage SSL certificates
B. Rewrite and redirect URLs based on rules
C. Authenticate users
D. Cache web pages

13. Which feature of Nginx helps with load balancing?

A. Proxy cache
B. Worker processes
C. Application server
D. SSL termination

14. What is the result of a DNS amplification attack?

A. Stealing user credentials
B. Redirecting web traffic to malicious servers
C. Overloading the DNS server with traffic
D. Accessing sensitive server data

15. What is a countermeasure for ARP spoofing attacks?

A. Using HTTPS
B. Configuring static ARP tables
C. Implementing CAPTCHA challenges
D. Enabling directory listing

16. What is the function of HTTP.sys in Microsoft IIS?

A. Encrypt user data
B. Handle HTTP requests at the kernel level
C. Execute server-side scripts
D. Perform vulnerability scanning

17. Which vulnerability involves excessive memory consumption through HTTP/2?

A. Buffer overflow
B. HTTP/2 continuation flood attack
C. CRLF injection
D. Cross-site request forgery (CSRF)

18. What is the primary goal of a Man-in-the-Middle (MITM) attack?

A. Exploit SQL queries
B. Intercept and alter communication
C. Steal session IDs
D. Deface websites

19. Which tool is best for identifying web server vulnerabilities?

A. Burp Suite
B. Acunetix Web Vulnerability Scanner
C. Wireshark
D. JHijack

20. What does the SameSite cookie attribute help prevent?

A. Brute-force attacks
B. CSRF attacks
C. Directory traversal
D. SQL injections

21. Which HTTP header forces browsers to use secure connections?

A. Content-Security-Policy
B. Strict-Transport-Security (HSTS)
C. X-Frame-Options
D. Referrer-Policy

22. What is the purpose of a demilitarized zone (DMZ. in web server deployment?

A. Encrypt server data
B. Prevent unauthorized internal access
C. Isolate the web server from public and internal networks
D. Enable secure remote administration

23. What is the principle of least privilege?

A. Allowing all users maximum access to reduce workload
B. Giving users access only to resources they need for their job
C. Restricting access to only administrators
D. Creating separate servers for different users

24. Which vulnerability allows unauthorized access to restricted files by manipulating URLs?

A. SQL injection
B. Path traversal
C. XSS
D. Remote code execution

25. What is a key advantage of Nginx's worker processes?

A. Single-threaded processing for efficiency
B. Load balancing across clients
C. Handling multiple connections simultaneously
D. Simplified logging

Answers

1. **Answer:** B

Explanation: Web servers host and deliver web content, such as files, applications, and data, over HTTP/HTTPS. They are the backbone of websites, making them accessible to users.

2. **Answer:** C

Explanation: Apache is a widely used open-source web server known for its flexibility, extensive module support, and strong community. It supports a variety of platforms and applications.

3. **Answer:** B

Explanation: Most vulnerabilities arise from configuration errors (e.g., misconfigured permissions. and outdated software, leaving the server exposed to exploits.

4. **Answer:** B

Explanation: Website defacement involves attackers modifying a website's appearance, often to spread propaganda, harm reputation, or display malicious content.

5. **Answer:** C

Explanation: Insecure default configurations, such as default admin credentials, allow attackers to gain unauthorized access and control over the web server.

6. **Answer:** A

Explanation: FTP is often targeted in brute-force attacks because it lacks modern security features, and its credentials are often transmitted in plaintext.

7. **Answer:** B

Explanation: A botnet consists of infected machines that attackers use to perform large-scale DDoS attacks, overwhelming web servers with traffic.

8. **Answer:** B

Explanation: Directory traversal exploits vulnerabilities to access restricted directories and files by manipulating file paths, such as using "../" to move up directory levels.

9. **Answer:** B

Explanation: Nmap is a versatile tool for scanning and discovering details about web servers, including open ports, services, and vulnerabilities.

10. **Answer:** C

Explanation: Cross-site scripting (XSS. allows attackers to inject malicious JavaScript into web pages viewed by users, often stealing cookies or session data.

11. **Answer:** B

Explanation: HTTP response splitting occurs when attackers manipulate HTTP headers to inject malicious responses, leading to cache poisoning or redirections.

12. **Answer:** B

Explanation: The mod_rewrite module in Apache allows URL rewriting and redirections based on defined rules, enhancing flexibility and control.

13. **Answer:** A

Explanation: Nginx's proxy cache stores frequently requested content to reduce backend server load and balance traffic effectively.

14. **Answer:** C

Explanation: DNS amplification attacks exploit open DNS resolvers to generate large volumes of traffic directed at a target, overwhelming its resources.

15. **Answer:** B

Explanation: Configuring static ARP tables ensures devices communicate only with authorized MAC addresses, preventing ARP spoofing attacks.z

16. **Answer:** B

Explanation: HTTP.sys in Microsoft IIS is a kernel-mode driver that processes HTTP requests, providing faster and more secure request handling.

17. **Answer:** B

Explanation: HTTP/2 continuation flood attacks exploit the continuation frame mechanism, causing excessive memory and CPU consumption, leading to server unresponsiveness.

18. **Answer:** B

Explanation: MITM attacks aim to intercept and alter communication between two parties, often to steal sensitive data or manipulate transactions.

19. **Answer:** B

Explanation: Acunetix Web Vulnerability Scanner detects vulnerabilities like SQL injection, XSS, and weak configurations, making it a comprehensive web application security tool.

20. **Answer:** B

Explanation: The SameSite cookie attribute prevents cookies from being sent with cross-site requests, mitigating CSRF attacks.

21. **Answer:** B

Explanation: HSTS headers enforce HTTPS connections, preventing browsers from accessing the site over unsecured HTTP and protecting against downgrade attacks.

22. **Answer:** C

Explanation: A DMZ isolates the web server from both public and internal networks, allowing controlled access and reducing security risks.

23. **Answer:** B

Explanation: The principle of least privilege restricts users' access to only the resources and actions necessary for their roles, reducing exposure to attacks.

24. **Answer:** B

Explanation: Path traversal attacks manipulate URLs to access files outside the intended directories, potentially exposing sensitive data.

25. **Answer:** C

Explanation: Nginx worker processes efficiently handle multiple connections using non-blocking I/O, enabling the server to support thousands of simultaneous requests.