# Module 13: Hacking Web Servers - Practical Guide

This guide focuses on the technical tools and step-by-step procedures used to identify and exploit vulnerabilities in web server software (Apache, IIS, Nginx) and their configurations.

## 1. Comprehensive Web Server Toolset

### Infrastructure & Vulnerability Scanners

* **Nikto:** The industry-standard command-line tool for web server auditing. It identifies over 6,700 potentially dangerous files/programs, checks for outdated server versions, and finds misconfigured headers.
* **OpenVAS / GVM:** A powerful open-source vulnerability scanner that provides comprehensive network and server-level security assessments.
* **Nessus:** Used for professional-grade vulnerability research, identifying unpatched server software and insecure configurations.

### CMS-Specific Tools

* **WPScan:** A specialized black-box WordPress security scanner. It is used to enumerate users, identify vulnerable plugins/themes, and perform password brute-forcing against WordPress sites.
* **JoomScan:** Similar to WPScan, but specifically designed for auditing Joomla CMS installations.
* **Droopescan:** A plugin-based scanner for Drupal and SilverStripe CMS.

### Information Gathering & OSINT

* **Netcraft:** A web-based service providing detailed information about a server's history, IP delegation, and the technology stack (OS, Web Server version) it is running.
* **Wappalyzer:** A browser extension that identifies the underlying technologies (CMS, frameworks, web servers) of a website with one click.
* **HTTrack:** A website mirroring tool. It allows you to download a complete site locally to search for hidden directories, developer comments, or configuration files offline.

## 2. Hands-On Lab Sessions

### Lab 1: Banner Grabbing (Nmap and Netcat)

**Goal:** Identify the exact version of the web server software to search for known CVEs.

1. **Using Nmap:**  
   nmap -sV --script banner [Target\_IP] -p 80
2. **Using Netcat (Manual Request):**
   * Connect: nc -nv [Target\_IP] 80
   * Send Request: Type HEAD / HTTP/1.0 and press **Enter twice**.
3. **Analysis:** Look for the Server: field in the response (e.g., Server: Apache/2.4.41 (Ubuntu)).

### Lab 2: Misconfiguration Scanning (Nikto)

**Goal:** Discover dangerous files, missing security headers, and open administrative portals.

1. **Basic Scan:**  
   nikto -h http://[Target\_IP]
2. **Analysis:** Look for results such as:
   * Directory indexing found at /admin/
   * Header X-Frame-Options is not present (Clickjacking vulnerability).
   * Server leaks inodes via ETag header.

### Lab 3: WordPress Enumeration (WPScan)

**Goal:** Harvest valid usernames and identify vulnerable plugins for potential exploitation.

1. **Enumerate Users:**  
   wpscan --url http://[Target\_URL] --enumerate u
2. **Check Vulnerable Plugins:**  
   wpscan --url http://[Target\_URL] --enumerate vp
3. **Result:** Use the discovered usernames for a targeted brute-force attack if credentials are weak.

### Lab 4: Testing for Insecure HTTP Methods (Nmap/Curl)

**Goal:** Determine if methods like PUT or DELETE are enabled, which could allow an attacker to upload a web shell.

1. **Using Nmap NSE:**  
   nmap -p 80 --script http-methods --script-args http-methods.url-path='/index.php' [Target\_IP]
2. **Manual Verification with Curl:**  
   curl -X PUT -d "test payload" http://[Target\_IP]/test\_file.txt
3. **Verification:** If you receive a 201 Created or 200 OK response, and the file exists on the server, the server is highly vulnerable.

## 3. Web Server Countermeasures (Defensive Controls)

Hardening a web server involves reducing the attack surface and ensuring the infrastructure is resilient to probes.

* **Server Masking:** Modify the "Server" header in the configuration files (e.g., ServerTokens Prod and ServerSignature Off in Apache) to provide generic information, making it harder for attackers to find specific CVEs.
* **Disable Insecure HTTP Methods:** Explicitly disable PUT, DELETE, and TRACE in the web server configuration to prevent unauthorized file manipulation and Cross-Site Tracing (XST).
* **Patch Management:** Regularly update the OS and the web server software (Apache, IIS, Nginx) to mitigate known vulnerabilities.
* **Least Privilege:** Run the web server service under a dedicated, low-privilege user account (e.g., www-data on Linux) to limit the impact of a successful compromise.
* **Remove Default Content:** Delete default pages, manual files, and example scripts that come pre-installed, as these often contain vulnerabilities or reveal server paths.
* **File Permissions:** Set strict permissions on the web root directory. The web server user should have read-only access to most files and write access only where strictly necessary (e.g., upload folders).
* **HSTS (HTTP Strict Transport Security):** Use the HSTS header to force browsers to only communicate via HTTPS, preventing protocol downgrade attacks.
* **WAF (Web Application Firewall):** Deploy a WAF to inspect incoming HTTP traffic and block signatures associated with directory traversal and server-level exploits.

## 4. CEH Exam "Checklist" for Module 13

* **Directory Traversal:** Know that ../ is the signature for this attack.
* **Insecure Methods:** Remember that **PUT** allows file uploads and **TRACE** can lead to **XST (Cross-Site Tracing)**.
* **Robots.txt:** Always check this file early in a web server engagement to find directories the owner wants to keep hidden.
* **Server Masking:** This is the process of changing the server's response banner to provide generic or false information to attackers.