Nikto: The Definitive Guide to Web Server Reconnaissance

Chapter 1: The Sentinel of the Web - An Introduction to Nikto

In the vast and complex ecosystem of cybersecurity, few tools have demonstrated the longevity and persistent relevance of Nikto. For over two decades, it has served as a foundational instrument for security professionals, system administrators, and penetration testers. It is a sentinel, tirelessly probing the public-facing interfaces of web servers for known weaknesses and misconfigurations. This guide provides an exhaustive exploration of Nikto, from its core principles and historical origins to its practical application in modern security assessments. It aims to equip the reader with a deep, nuanced understanding of not just how to operate the tool, but how to interpret its findings, integrate it into a larger security workflow, and wield it with the ethical and legal responsibility that its power demands.

1.1 What is Nikto? A Foundational Overview

Nikto is an open-source web server scanner distributed under the GNU General Public License (GPL).[1] It is implemented as a command-line utility written in the Perl programming language, a choice that has ensured its cross-platform compatibility for decades.[3] The primary function of Nikto is to perform comprehensive vulnerability scanning against web servers and web applications.[5] It is not an exploitation framework designed to gain access to a system, but rather a reconnaissance tool engineered to identify potential vectors of attack. The power of Nikto resides in its extensive and frequently updated databases of known vulnerabilities and misconfigurations. Its scanning logic is built upon a vast repository of checks designed to uncover a wide spectrum of security issues. These checks fall into several key categories:

- **Dangerous Files and Programs:** Nikto searches for thousands of potentially harmful files and Common Gateway Interface (CGI) scripts that may have been left on a server, such as old administration scripts, default installation files, or known malicious web shells.[1]
- **Outdated Server Software:** The scanner cross-references the target server's software banner against a database of known versions to identify outdated and unpatched web

servers, which are often susceptible to public exploits.[2]

- **Server-Specific Misconfigurations:** Nikto performs checks for configuration errors that are specific to certain server types, such as Apache, Microsoft IIS, or Nginx.[3] This can include issues like the presence of multiple index files, improperly configured HTTP server options, or enabled debugging methods that could leak sensitive information.[3]

The sheer breadth of its knowledge base is a core component of its effectiveness. At present, Nikto can detect over 7,000 potentially dangerous files or programs, check for outdated versions of more than 1,250 server types, and identify version-specific problems on over 270 server variants.[9] This comprehensive database allows it to quickly perform a wide-ranging initial assessment of a web server's security posture.

1.2 The History and Philosophy of Nikto

Nikto's journey began on December 27, 2001, when its creator, Chris Sullo, released the 1.00 Beta version.[3] It rapidly gained popularity within the nascent information security community, evolving into one of the most widely used free web vulnerability scanners. A major milestone was the release of version 2.0 in November 2007, which represented several years of accumulated improvements and a significant enhancement of its capabilities.[11] Development has since transitioned to a more collaborative, community-driven model, hosted on GitHub. This evolution was marked by David Lodge joining the development team in 2008 and Sullo's return to the project in 2009, cementing its status as an enduring open-source project.[11]

The origin of the tool's name is a frequent point of curiosity and is explicitly tied to classic science fiction cinema. The name "Nikto" is taken from the iconic phrase "Klaatu barada nikto" from the 1951 film *The Day the Earth Stood Still*, a command used to stop a destructive robot. The name was also famously used by the character Ash in the 1992 cult classic film *Army of Darkness*.[11] While the word has appeared coincidentally in other popular franchises, such as *Star Wars* and *Call of Duty*, these are unrelated to the scanner's identity and history.[13]

A fundamental aspect of Nikto's design philosophy is the deliberate prioritization of speed over stealth. Nikto is engineered to be fast and efficient, performing thousands of checks in a relatively short amount of time.[4] This efficiency, however, comes at the cost of being exceptionally "noisy." A standard Nikto scan generates a massive volume of HTTP requests, many of which are for files that do not exist, resulting in thousands of 404 Not Found errors in the target server's logs.[10] This makes a Nikto scan trivial to detect for any administrator monitoring their logs or using a basic Intrusion Detection System (IDS).[4]

While some documentation notes that Nikto can be used in an "evasive manner" to elude certain types of IDS [6], this capability is limited. The evasion techniques modify the signature of individual requests but do not reduce the sheer volume, which remains its most obvious footprint. This trade-off is a core characteristic that every user must understand before deploying the tool.

1.3 Why Nikto Still Matters in a Modern Security Stack

In an era of sophisticated Dynamic Application Security Testing (DAST) frameworks and advanced threat intelligence platforms, a two-decade-old command-line tool might seem anachronistic. Yet, Nikto remains a standard utility in the arsenal of security professionals and is included by default in leading security-focused operating systems like Kali Linux.[5] Its enduring relevance is also underscored by its inclusion in resources from government cybersecurity agencies like the U.S. Cybersecurity and Infrastructure Security Agency (CISA).[1] Nikto's value in a modern context stems from its specialized role: it is arguably the best tool for rapidly identifying "low-hanging fruit." These are the common, often-overlooked vulnerabilities that provide an initial foothold for an attacker. It excels at discovering outdated server components, default configuration files left over from development, and insecure server settings that more complex, application-focused DAST tools might not prioritize.[7] Beyond its use as an offensive assessment tool, Nikto serves a crucial function in defensive validation and compliance. A web server's configuration is a complex tapestry of settings, files, and software versions. Nikto's extensive database of checks represents a curated list of what *not* to do—a benchmark of common security failings. An organization can therefore leverage Nikto not just to simulate an attack, but to perform a rapid audit of its own infrastructure. Running Nikto against a production server and receiving a "clean" report provides a baseline level of assurance that fundamental security hygiene practices are being followed. It confirms that default scripts have been removed, that software is up to date, and that dangerous HTTP methods are disabled. This positions Nikto as a vital tool for system administrators and DevOps teams, allowing them to quickly verify their hardening efforts against a globally recognized standard of common vulnerabilities, a practice encouraged by its inclusion in CISA's best practice resources.[2]

Chapter 2: Getting Started - Installation and Configuration

A successful Nikto deployment begins with a proper installation and an understanding of its core dependencies. As a Perl-based tool, its setup process varies across different operating systems, with some requiring more manual configuration than others. This chapter provides detailed, platform-specific instructions to ensure a fully functional Nikto environment, ready for scanning.

2.1 Core Prerequisite: The Perl Environment

At its heart, Nikto is a collection of Perl scripts.[3] Therefore, a functional Perl interpreter is the

absolute primary prerequisite for running the tool. Most Linux distributions and macOS come with Perl pre-installed, making the setup process on these platforms relatively straightforward. Windows, however, does not include a native Perl interpreter, making its installation a mandatory first step for users on that platform.[18]
Furthermore, for Nikto to perform one of its most critical functions—scanning servers that use SSL/TLS encryption (HTTPS)—specific Perl modules must be available. On Debian-based systems like Ubuntu, this dependency is typically the libnet-ssleay-perl package.[25] On Red Hat-based systems, the equivalent is
perl-Net-SSLeay.[26] For Windows users, this functionality is usually provided by installing a comprehensive Perl distribution like ActivePerl in conjunction with the OpenSSL libraries.[9] Without these SSL-related modules, Nikto will be unable to scan HTTPS websites.

2.2 Installation on Linux (Debian/Ubuntu/Kali)

For users of Debian, Ubuntu, Kali Linux, and other Debian-based distributions, there are two primary methods for installing Nikto.

Using Package Managers (Recommended)

The simplest and most reliable method is to use the Advanced Package Tool (APT). This approach ensures that Nikto and all its necessary dependencies, including the required Perl modules, are installed and configured correctly. The command is executed from the terminal:

Bash

```
sudo apt update
sudo apt install nikto
```

This command first updates the local package lists and then installs the Nikto package.[7] For users of Kali Linux, Nikto is typically included in the default installation and can be found under the "Vulnerability Analysis" category of tools.[5]

From Source (Advanced)

For users who require the absolute latest version of Nikto, which may not yet be available in their distribution's repositories, installation from the source code on GitHub is the preferred method. This is accomplished using the git version control tool.

1. Clone the official repository:

Bash
git clone https://github.com/sullo/nikto.git

2. Navigate into the program directory:
   Bash
   cd nikto/program/

3. Run the scanner directly using the Perl interpreter:
   Bash
   perl nikto.pl -h <target>

This method gives the user direct access to the latest features and plugin updates as soon as they are pushed to the master branch of the repository.[7]

2.3 Installation on Windows

Installing Nikto on Windows is a more involved process due to the lack of a native Perl environment and other dependencies. It requires several manual steps to create a functional environment.

- **Step 1: Install a Perl Distribution.** The most common choice for Windows is ActivePerl, a free distribution from ActiveState.[7] After downloading and running the installer, it is crucial to add the Perl binary directory (e.g., C:\Perl\bin) to the system's PATH environment variable. This allows the perl command to be run from any command prompt.[24]
- **Step 2: Install OpenSSL and Build Dependencies.** To enable SSL/TLS scanning, a more complex chain of dependencies is required. This often includes installing OpenSSL for Windows, a C/C++ compiler like MinGW, and the Microsoft nmake utility. This process can be convoluted and requires careful attention to setting multiple PATH variables to ensure all components can find each other.[24]
- **Step 3: Clone the Nikto Repository.** Using a command-line terminal that supports git, such as Git Bash (which is highly recommended for this purpose), clone the official Nikto repository [7]:
  Bash
  git clone https://github.com/sullo/nikto.git

- **Step 4: Run Nikto.** Once all prerequisites are in place, navigate to the Nikto directory in the command prompt and execute the scan using the perl command [24]:
  Bash
  cd nikto/program/
  perl nikto.pl -h <target>

2.4 Installation on macOS

For macOS users, the installation process is greatly simplified by the Homebrew package manager. Homebrew is the de facto standard for installing open-source software on macOS and handles all dependencies automatically.

1. **Ensure Homebrew is installed.** If not already present, it can be installed by running the command provided on the official Homebrew website.
2. **Install Nikto.** With Homebrew installed, a single command in the terminal is all that is required [5]:
   Bash
   brew install nikto

Homebrew will download Nikto and all necessary Perl modules and dependencies, making it ready to use immediately.[28]

2.5 Initial Configuration: The nikto.conf File

While Nikto can be controlled entirely through command-line flags, its behavior can also be customized through a central configuration file. In older versions this was config.txt, but in modern versions it is nikto.conf.[11] This file allows users to set persistent options, which is particularly useful for settings that are used frequently, such as proxy configurations or custom user agents.
Key variables that can be defined in nikto.conf include:

- USERAGENT: Changes the User-Agent string sent in Nikto's HTTP requests. This can be useful for mimicking a standard browser to be slightly less conspicuous.
- PROXYHOST, PROXYPORT, PROXYUSER, PROXYPASS: These variables configure Nikto to route all its traffic through an HTTP proxy, which is essential for testing from behind a corporate firewall or for use with interception proxies.[17]
- DEFAULTHTTPVER: Sets the default HTTP version to use for initial requests.[19]

By editing nikto.conf, users can create a customized, persistent scanning configuration that aligns with their specific testing environment and requirements, saving them from having to re-enter the same command-line options for every scan.

Chapter 3: The Art of the Scan - Mastering Nikto's Command Line

The true power of Nikto is unlocked through its versatile and comprehensive command-line

interface. While a basic scan is simple to initiate, a deep understanding of its flags and options transforms Nikto from a blunt instrument into a precision tool for surgical reconnaissance. This chapter serves as a definitive reference for mastering Nikto's command line, enabling users to tailor every aspect of the scan, from target selection and output formatting to advanced evasion and performance tuning.

3.1 Basic Scanning: Targeting and Port Selection

The most fundamental operation in Nikto is defining the target. This is accomplished with the -h (or -host) flag, which is the only mandatory option for a scan.

- **Targeting a Host:** The target can be specified in several ways, offering flexibility for different scenarios.[31]
    - **By Hostname:** nikto -h example.com
    - **By IP Address:** nikto -h 192.168.1.1
    - **From a File:** For scanning multiple targets, a simple text file can be provided, with one host per line. nikto -h targets.txt.[17]
- **Specifying Ports:** By default, Nikto assumes the target web server is running on TCP port 80 (the standard for HTTP).[17] The
  -p (or -port) option provides granular control over which ports to scan.
    - **Single Port:** To scan a service on a non-standard port, such as a development server on 8080: nikto -h example.com -p 8080.[31]
    - **Comma-Separated List:** To scan multiple specific ports on a single host: nikto -h example.com -p 80,443,8080.[12]
    - **Port Range:** To scan a continuous range of ports: nikto -h example.com -p 80-90.[12]

3.2 Handling Secure Servers: SSL/TLS Options

When scanning a website that uses HTTPS, the -ssl flag is essential. This option instructs Nikto to wrap its HTTP requests in an SSL/TLS layer, allowing it to communicate with secure servers.[12]

A critical aspect of using the -ssl flag is its impact on performance. By default, Nikto will attempt to determine automatically whether a given port is serving HTTP or HTTPS content. It does this by first trying one protocol, and if that fails or times out, trying the other. This automatic detection can be slow, especially if a firewall is silently dropping packets or if the server is slow to respond to the incorrect protocol request.[17] When a user knows that a port (such as the standard 443) is for HTTPS, explicitly using the
-ssl flag bypasses this slow auto-detection process. The command nikto -h secure.example.com -p 443 -ssl will immediately initiate a TLS handshake, dramatically speeding up the scan by eliminating the initial, failing HTTP request. This is a crucial practical

technique for efficient scanning.

3.3 Controlling the Output: Formatting and Saving Results

The raw output that Nikto prints to the terminal is useful for real-time analysis, but for documentation, reporting, and integration with other tools, saving the results to a file is necessary.

- **Saving to a File:** The -o (or -output) flag is used to specify an output file. For example, nikto -h example.com -o scan_results.txt will save the scan output to a file named scan_results.txt.[31]
- **Specifying Format:** The -Format flag provides control over the structure of the saved file. This is vital for generating reports suitable for different audiences or for programmatic parsing. Supported formats include [20]:
    - txt: Plain text (the default).
    - htm: A user-friendly HTML report.
    - csv: Comma-Separated Values, ideal for importing into spreadsheets.
    - xml: Extensible Markup Language, for structured data interchange.
    - nbe: Nessus NBE format, for integration with the Nessus vulnerability scanner.

A common use case is generating a human-readable report for stakeholders: nikto -h example.com -o report.html -Format htm.[31]

3.4 Advanced Operations: Proxies, Authentication, and Evasion

For more complex testing scenarios, Nikto provides a suite of advanced options.

- **Proxy Support:** Nikto can route its traffic through an HTTP proxy using the -useproxy option.[32] The proxy's address and port are typically defined in the nikto.conf file. This is indispensable when testing from within a restricted corporate network or when using an interception proxy like Burp Suite or OWASP ZAP to monitor and manipulate Nikto's requests.
- **Authentication:** For websites that require authentication, the -id flag can supply credentials. The format is userid:password for Basic authentication or userid:password:realm for NTLM authentication.[17]
- **IDS Evasion (-evasion):** Nikto includes a set of techniques aimed at obfuscating its scan traffic to evade simple, signature-based Intrusion Detection Systems (IDS). The -evasion flag is followed by one or more numeric codes corresponding to a specific technique [34]:
    - 1: Random URI encoding (non-UTF8).
    - 2: Directory self-reference (e.g., /./).
    - 3: Premature URL ending.
    - 4: Prepending a long random string to the request.

- 5: Appending fake parameters to requests.
- 6: Using a TAB character as a request spacer instead of a space.
- 7: Randomizing the case of letters in the URL.
- 8: Using the Windows directory separator (\) instead of (/).

While these options exist, their practical utility in modern environments is limited. Nikto's fundamental scanning methodology relies on sending a high volume of requests for files that are unlikely to exist, generating a storm of 404 Not Found errors in server logs.[4] The -evasion techniques modify the appearance of individual requests but do nothing to reduce the overall volume. Consequently, while a simple IDS looking for the default "Nikto" User-Agent string might be fooled, any log analysis tool, SIEM, or human administrator will immediately spot the anomalous spike in error traffic from a single source IP. The core "noisiness" of Nikto's approach remains its most identifiable characteristic, rendering these evasion features largely ineffective against modern security monitoring.

3.5 The Command-Line Flag Compendium (Table)

To serve as a quick and practical reference, the following table consolidates the most essential Nikto command-line options into a structured format.

Category	Flag & Parameters	Description	Example Usage
Targeting	-h, -host <target>	Specifies the target host, IP, or a file with a list of hosts.	nikto -h example.com
	-p, -port <ports>	Specifies the port(s) to scan (single, list, or range). Default: 80.	nikto -h example.com -p 80,443
Output	-o, -output <file>	Saves the scan output to the specified file.	nikto -h example.com -o report.txt
	-Format <format>	Sets the output format (txt, csv, htm, xml, nbe).	nikto -h example.com -o r.html -Format htm
SSL/TLS	-ssl	Forces SSL mode on the specified port(s) for faster scanning.	nikto -h secure.com -p 443 -ssl
Tuning	-Tuning <type(s)>	Focuses the scan on specific vulnerability types (0-9, a-c).	nikto -h example.com -Tuning 4,9
Performance	-timeout <seconds>	Sets the timeout for each request. Default: 10.	nikto -timeout 5
	-maxtime <duration>	Sets the maximum	nikto -maxtime 30m

		scan time per host (e.g., 1h, 60m).	
Functionality	-Plugins <list>	Specifies which plugins to run.	nikto -Plugins apache_expect_xss
	-update	Updates Nikto's databases and plugins (legacy, see Chapter 6).	nikto -update
	-useproxy	Uses the HTTP proxy defined in the configuration file.	nikto -h example.com -useproxy
	-no404	Disables 404 checking to reduce requests, but may increase false positives.	nikto -h embedded.device -no404

Chapter 4: Decoding the Output - From Raw Data to Actionable Intelligence

Running a Nikto scan is only the first step; the true skill lies in interpreting the resulting data. A Nikto report is a dense stream of information that, to the untrained eye, can appear cryptic or overwhelming. This chapter provides a detailed guide to decoding this output, transforming it from raw data into the actionable intelligence needed to build a meaningful security assessment and prioritize remediation efforts.

4.1 Anatomy of a Nikto Report

A standard Nikto scan output follows a consistent structure. It begins with a header section that provides metadata about the scan, confirming the target and parameters used.[31] This includes:

- **Target IP:** The resolved IP address of the host being scanned.
- **Target Hostname:** The hostname provided by the user.
- **Target Port:** The TCP port on which the scan was performed.
- **Start Time:** A timestamp indicating when the scan commenced.

Following this header, the main body of the report lists the findings. Each finding is presented on a new line, typically prefixed with a + symbol to indicate a positive result.[36] A single line of output contains the core information about a potential vulnerability. For instance, a finding like + The anti-clickjacking X-Frame-Options header is not present. [36] is a self-contained piece of intelligence. It identifies a specific weakness (a missing security header), names the header (

X-Frame-Options), and hints at the associated attack vector (clickjacking). The report concludes with a summary, including the end time of the scan and the total number of requests made.

4.2 Common Findings and Their Significance

Understanding the most common types of findings is crucial for effective analysis.

- **Server Banner Disclosure:** One of the first pieces of information Nikto reports is the server banner, such as Server: Apache/2.4.41 (Ubuntu).[36] This reveals the web server software, its version, and often the underlying operating system. The immediate action for an analyst is to research this specific version in public vulnerability databases like the CVE (Common Vulnerabilities and Exposures) list to determine if there are any known, exploitable vulnerabilities.[8]
- **Missing Security Headers:** Nikto is particularly adept at identifying the absence of critical HTTP security headers. These headers are instructions that the server sends to the client's browser to enable built-in security features. Common findings include:
    - X-Frame-Options: When missing, the site is vulnerable to "clickjacking," where an attacker can embed the victim's site in an invisible <iframe> on a malicious page to trick users into performing unintended actions.[21]
    - X-XSS-Protection: This header can enable the browser's built-in filter against some forms of Cross-Site Scripting (XSS) attacks.[36]
    - Strict-Transport-Security (HSTS): Its absence on an SSL-enabled site means browsers are not instructed to enforce HTTPS-only connections, potentially allowing for protocol downgrade attacks.[38]

      These are often considered "low-hanging fruit" as they can typically be fixed with simple configuration changes in the web server (e.g., Apache or Nginx).36
- **Sensitive Files and Directories:** Nikto's database is filled with checks for default, backup, and administrative files and directories that should not be publicly accessible.
    - Findings like /admin/, /phpmyadmin/, or /wp-admin/ indicate the presence of administrative interfaces that are prime targets for brute-force login attempts.[36]
    - A browsable directory like /icons/ might seem harmless, but it can reveal information about the server's structure and technology stack.[36]
    - The discovery of robots.txt is always noteworthy. While its purpose is to guide search engine crawlers, it often contains a list of paths the administrator explicitly wishes to hide, inadvertently providing a roadmap of potentially interesting locations to an attacker.[8]
- **Outdated Software:** This is one of Nikto's most critical findings. The report will often explicitly state The web server version is outdated.[8] This is a high-priority alert, as outdated software is a leading cause of security breaches due to the availability of public exploits for its known vulnerabilities.

4.3 The OSVDB Mystery: Handling Deprecated IDs

A unique characteristic of Nikto's output is its frequent reference to OSVDB identifiers, such as OSVDB-3092.[36] These IDs correspond to entries in the Open Sourced Vulnerability Database, a community-driven project that was once a valuable resource. The problem for modern users is that the OSVDB was officially shut down in April 2016, and its website is no longer active.[39]

This situation requires the analyst to move beyond passively consuming the report and engage in active investigation. An OSVDB ID should not be dismissed as irrelevant simply because its source is defunct. The ID serves as a historical pointer to a vulnerability that may still be present and exploitable. The analyst's task is to uncover the details of that original vulnerability. This process transforms the Nikto report from a list of conclusions into a set of investigative leads. The following steps can be taken:

1. **Search for Mappings:** The first step is to search for publicly available mappings that correlate OSVDB IDs to more current identifiers, such as CVEs.[39]
2. **Consult Web Archives:** If a direct mapping cannot be found, the Internet Archive's Wayback Machine can be used. By searching for archived snapshots of the osvdb.org website from before 2016, it is often possible to view the original vulnerability entry, which contains a detailed description of the issue.[39]
3. **Contextual Search:** The description provided by Nikto alongside the OSVDB ID (e.g., /admin/: This might be an administrative login page.) can be used as a search query in modern search engines to find information about the general class of vulnerability.

This investigative process is a crucial, non-obvious step in the modern workflow of using Nikto. It elevates the role of the security professional from a mere tool operator to an analyst who must use open-source intelligence (OSINT) techniques to contextualize historical data and assess its present-day relevance.

4.4 Prioritizing Findings: From Noise to Action Plan

Nikto is intentionally aggressive and can generate a significant amount of output, including informational items and potential false positives.[40] A key skill is the ability to triage these findings and create a prioritized action plan. A general framework for prioritization is as follows:

- **High Priority:**
    - Vulnerabilities with a direct path to compromise, such as Command Execution (-Tuning 8), SQL Injection (-Tuning 9), or Remote File Inclusion.
    - Outdated software for which a public, weaponized exploit is known to exist.
    - Exposed configuration files containing plaintext credentials.[8]
- **Medium Priority:**

    - Missing critical security headers (X-Frame-Options, Strict-Transport-Security).
    - Exposed administrative login panels (/admin/, /phpmyadmin/).
    - Information disclosure that reveals internal paths, usernames, or sensitive configuration details.
- **Low Priority / Informational:**
    - Server banner disclosure (useful for an attacker, but not a vulnerability in itself).
    - Browsable directories containing non-sensitive content (e.g., /icons/).
    - The presence of robots.txt.

This prioritization framework, adapted from security best practices [36], allows teams to focus their limited resources on mitigating the most severe risks first, ensuring that critical vulnerabilities are addressed before less impactful issues.

Chapter 5: Advanced Reconnaissance - Extending Nikto with Plugins and Tuning

While Nikto's default scan is powerful, its true potential as a reconnaissance tool is realized through customization. By leveraging its scan tuning options and extensible plugin architecture, a security professional can move beyond generic assessments to perform highly targeted and efficient tests. This chapter explores the advanced features that allow users to focus Nikto's power, extend its capabilities, and uncover vulnerabilities that a standard scan might miss.

5.1 Scan Tuning: Focusing Your Efforts

One of Nikto's most potent features for tailoring a scan is the -Tuning option.[5] This command-line flag allows the user to control which specific classes of tests are executed, enabling faster, more focused assessments. This is particularly useful when a tester wants to investigate a specific type of vulnerability or when scanning a sensitive system where minimizing the number of requests is desirable.

The -Tuning flag accepts a string of numeric or alphabetic codes, each corresponding to a category of vulnerability check. Multiple codes can be combined to run several test types simultaneously. The available tuning options are comprehensive [33]:

- 0: **File Upload** - Checks for exploits that allow a file to be uploaded to the server.
- 1: **Interesting File / Seen in logs** - Looks for suspicious files or attack patterns observed in server logs.
- 2: **Misconfiguration / Default File** - Identifies default files, documentation, or resources that are improperly configured.
- 3: **Information Disclosure** - Seeks resources that reveal sensitive information, such as file paths or account names.

- 4: **Injection (XSS/Script/HTML)** - Tests for various injection vulnerabilities, including Cross-Site Scripting.
- 5: **Remote File Retrieval - Inside Web Root** - Checks for vulnerabilities that allow unauthorized access to files within the web root directory.
- 6: **Denial of Service** - Identifies resources that could be used to launch a DoS attack (Note: Nikto does not perform actual DoS attacks).
- 7: **Remote File Retrieval - Server Wide** - Tests for vulnerabilities that allow access to files anywhere on the server's file system.
- 8: **Command Execution / Remote Shell** - Looks for vulnerabilities that permit the execution of system commands.
- 9: **SQL Injection** - Checks for any type of SQL injection vulnerability.
- a: **Authentication Bypass** - Tests for flaws that allow access to protected resources.
- b: **Software Identification** - Focuses on checks that positively identify installed software.
- c: **Remote Source Inclusion** - Checks for vulnerabilities that allow the inclusion of remote source code.
- x: **Reverse Tuning** - This powerful modifier inverts the logic. When x is used, Nikto will *exclude* the specified test types and run all others.

Practical Use Case: Imagine a penetration tester is tasked with assessing a web application specifically for its susceptibility to data exfiltration and command execution. Instead of running a full, time-consuming scan, they can use tuning to focus their efforts: nikto -h example.com -Tuning 7,8,9. This command instructs Nikto to run only the tests for server-wide file retrieval, command execution, and SQL injection, providing a highly relevant and efficient assessment.

5.2 The Power of Plugins: Extending Nikto's Reach

Nikto is built on a modular and extensible architecture. Its core functionality is augmented by a suite of plugins, which are essentially self-contained Perl scripts located in the /plugins/ directory of the Nikto installation.[44] This pluggable design allows Nikto to be easily updated with new checks and enables users to extend its capabilities to suit specific needs.

- **Managing Plugins:** A user can list all available plugins by running Nikto with the -list-plugins flag.[5] To execute a scan using only one or more specific plugins, the -Plugins flag is used, followed by a comma-separated list of plugin names.[31]
- **Example Plugins:** The plugin system allows for highly specialized checks that go beyond Nikto's general-purpose scans. For example:
    - An **SSL scanner plugin** can perform in-depth checks on a server's SSL/TLS configuration, looking for weak ciphers or expired certificates.[44]
    - A **WordPress vulnerability plugin** can be used to target a WordPress site specifically, detecting outdated themes, vulnerable plugins, and other common CMS-specific issues. A command like nikto -h myblog.com -Plugins

wordpress_vuln might yield specialized output such as + WordPress Plugin Detected: old-plugin (Version: 1.0).[44]
    - A **header check plugin** can focus solely on analyzing the HTTP security headers returned by the server.[44]
- **Writing Custom Plugins:** For advanced users, the ability to write custom plugins is Nikto's most powerful feature. While a detailed tutorial on Perl scripting is beyond the scope of this guide, the Nikto documentation provides an overview of its plugin API.[45] By creating their own
.pl files and placing them in the plugins directory, developers and security researchers can add new, proprietary, or highly specialized checks, making Nikto an infinitely extensible platform for vulnerability discovery.[31]

5.3 Mutation Techniques: Guessing the Unknown

In older versions of Nikto, the -mutate option was a key feature for discovering non-obvious content. This option enables various "mutation" tests that combine dictionaries and known paths to guess additional file and directory names.[17] The different mutation levels included:

- 1: Test all files with all directories from the database.
- 3: Enumerate potential Apache usernames (e.g., /~user).
- 5: Attempt to brute-force sub-domain names.

It is important to understand that in modern versions of Nikto, these -mutate options have been largely deprecated and superseded by more granular controls within the plugin system.[46] For example, what was once
-mutate 3 is now handled by the apacheusers plugin. While the flags may still function for backward compatibility, the modern and preferred method for this type of testing is to use the -Plugins flag to call the specific plugin that performs the desired function. This provides a more modular and transparent approach to advanced scanning.

Chapter 6: Tool Maintenance - Keeping Nikto Sharp

In the fast-paced world of cybersecurity, a vulnerability scanner is only as good as its last update. New vulnerabilities are discovered daily, and server software is constantly evolving. To remain effective, a tool like Nikto must be kept current. This chapter addresses the critical process of updating Nikto, clarifying the modern procedure and resolving confusion that may arise from legacy documentation.

6.1 The Old Way vs. The New Way: -update vs. git pull

For many years, the standard method for updating Nikto's vulnerability databases and plugins was the built-in -update command-line flag. Numerous tutorials and older documentation sources still reference this command: nikto -update.[31] While this method served its purpose, it has been officially deprecated for modern installations of the tool.
The official Nikto GitHub wiki explicitly states that as of version 2.1.6 and later, the git version control system is the sole recommended method for both installing and updating Nikto.[47] This reflects a broader shift in open-source software distribution towards more robust and reliable version control systems. Using an in-app updater like
-update can sometimes lead to partial or failed updates, whereas git ensures that the entire codebase and all associated files are brought to a consistent, stable state.
For any user who has installed Nikto by cloning the repository from GitHub, the correct and only procedure required to update the entire tool—including the core Perl scripts, plugins, and all vulnerability databases (db_tests, etc.)—is to use the git pull command.
The process is simple:

1. Open a terminal.
2. Navigate to the root directory of your local Nikto installation (the one containing the .git folder).
3. Execute the command:
   Bash
   git pull

This command will contact the official GitHub repository, download any changes that have been made since the last update, and merge them into the local installation.[34] The Nikto maintainers ensure that the
master branch remains stable, so this command can be run confidently to get the latest stable version of the tool and all its components.

6.2 Verifying Your Version

After performing an update, it is a good practice to verify that the new version has been successfully applied. Nikto provides a simple command-line flag for this purpose: nikto -Version.
Running this command will display the current version of the Nikto scanner itself, as well as the version numbers of its core databases and plugins.[7] This provides a quick and easy way to confirm that the
git pull operation was successful and that the scanner is now operating with the latest available checks and features.

Chapter 7: Nikto in Context - A Comparative Analysis

No single tool can address every aspect of a comprehensive security assessment. Understanding where a tool excels and where its limitations lie is the hallmark of an effective security professional. Nikto is a powerful instrument, but its true value is realized when it is used as part of a larger, well-orchestrated toolkit. This chapter provides a critical comparative analysis, positioning Nikto alongside other essential security tools to help users understand its unique role, its strengths and weaknesses, and when to deploy it for maximum impact.

7.1 Nikto vs. Nmap (and the Nmap Scripting Engine)

A common point of confusion for newcomers is the distinction between Nikto and Nmap, two foundational tools in network reconnaissance.

- **Distinct Roles:** At their core, the two tools serve fundamentally different purposes. Nmap (Network Mapper) is a *network discovery and port scanner*. Its primary function is to map a network, identify which hosts are online, and determine which TCP/UDP ports are open on those hosts.[21] Nikto, in contrast, is a dedicated *web server scanner*. It operates at a higher level, performing deep-dive analysis on a specific web service (e.g., on port 80 or 443) that Nmap has already identified.[10]
- **The NSE Overlap:** The lines begin to blur with the introduction of the Nmap Scripting Engine (NSE). The NSE is a powerful feature that allows Nmap to run scripts to perform a wide variety of automated tasks, including vulnerability detection.[10] Scripts like http-enum can perform checks for common web directories and files, mimicking some of Nikto's functionality.[48] Some NSE scripts are even capable of leveraging Nikto's own vulnerability database to enhance their checks.[48]
- **Strengths and Weaknesses:**
    - **Nikto:** Its strength lies in its singular focus and the depth of its specialized database, which contains over 7,000 checks specifically for web server vulnerabilities, misconfigurations, and outdated software.[9] It is fast and exceptionally easy to use for this specific purpose. Its primary weaknesses are its narrow focus (it only scans web services) and its notoriously "noisy" nature, which makes it easy to detect.[10]
    - **Nmap (with NSE):** Its key strength is its incredible versatility. Using the NSE, Nmap can audit a vast array of services, not just HTTP. It can perform brute-force attacks against FTP, enumerate users via SNMP, and check for vulnerabilities in SSH configurations.[21] Its weakness, in this comparison, is that its web scanning capabilities, while useful, are not as comprehensive or specialized as those of a dedicated tool like Nikto.[21]
- **Conclusion: A Complementary Workflow:** Nikto and Nmap are not competitors; they are partners in a logical workflow. A professional penetration test almost always begins with Nmap to build a map of the target environment. Once Nmap identifies a web server running on a host, Nikto is then deployed to perform an in-depth vulnerability assessment of that specific service. The workflow is sequential: Nmap finds the doors,

and Nikto checks if they are unlocked.[22]

7.2 Nikto vs. DAST Tools (OWASP ZAP, Burp Suite)

Nikto is often categorized alongside tools like OWASP ZAP (Zed Attack Proxy) and Burp Suite, but they belong to different classes of security tools.

- **Different Classes of Tool:** Nikto is a vulnerability *scanner*. It operates like a checklist, running through its predefined database of known issues and reporting what it finds.[49] OWASP ZAP and Burp Suite are comprehensive *Dynamic Application Security Testing (DAST) frameworks*.[51] They provide an interactive environment for deep analysis of a web application's behavior and logic.
- **Key Differentiators:**
    - **Interactivity and Proxying:** The most significant difference is that ZAP and Burp Suite are designed to function as man-in-the-middle (MitM) proxies. A tester configures their browser to route traffic through the tool, which can then intercept, inspect, modify, and replay every HTTP request and response.[52] This interactive capability is essential for testing complex application workflows, session management, and business logic flaws. Nikto is non-interactive and cannot perform these functions.
    - **Discovery Method:** Nikto's discovery is based on its static database of known files and directories. ZAP and Burp employ "spiders" or "crawlers" that dynamically navigate the target application, following links to discover its unique structure and uncover pages and API endpoints that would not be found in any standard list.[52]
    - **Vulnerability Focus:** Nikto excels at finding server-level issues: misconfigurations, default files, and outdated software versions.[22] ZAP and Burp are purpose-built to find complex, application-specific vulnerabilities like SQL Injection (SQLi), Cross-Site Scripting (XSS), and Command Injection. They do this through active "fuzzing," where they systematically send malformed and malicious payloads to every input field to observe how the application responds.[49] While Nikto has some checks for these vulnerability classes (e.g., via `-Tuning 4,9`), this is not its primary strength compared to a dedicated DAST framework.

The relationship between these tools is not one of competition, but of sequence within a typical penetration testing pipeline. A tester's workflow often progresses from broad reconnaissance to focused analysis. Nikto is deployed in the early reconnaissance phase to quickly identify server-level weaknesses and known vulnerabilities.[40] The findings from this scan then provide direct leads for the next phase. For example, if Nikto reports `+ /phpmyadmin/: phpMyAdmin installation detected.` [36], the tester has found a high-value target of opportunity. They would then configure Burp Suite or OWASP ZAP to proxy their connection to that specific phpMyAdmin instance. From there, they would perform in-depth,

interactive testing: attempting to brute-force credentials, looking for XSS vulnerabilities in the login page's parameters, or testing for SQL injection flaws—all tasks for which DAST frameworks are specifically designed.[52] Nikto finds the promising targets; ZAP and Burp perform the deep-dive analysis.

7.3 Comparative Analysis of Vulnerability Assessment Tools (Table)

To provide a clear, at-a-glance summary of where Nikto fits within the broader ecosystem of vulnerability assessment tools, the following table compares it against its common counterparts.

Tool	Primary Purpose	Key Strength	Key Limitation	Ideal Use Case
Nikto	Web server vulnerability scanning	Comprehensive database of known vulnerabilities, misconfigurations, and outdated software.[9] Fast and easy to use for this purpose.	Not stealthy (very noisy).[18] Not designed for complex application logic testing. Can have false positives.[40]	Quick, initial assessment of a web server's security hygiene. Finding "low-hanging fruit".[21]
Nmap	Network discovery and port scanning	The industry standard for network mapping, service identification, and OS fingerprinting. Highly extensible via NSE.[21]	Web vulnerability scanning (via NSE) is less comprehensive than dedicated tools like Nikto.[21]	The first step in any network assessment: discovering live hosts and open ports to identify potential targets.[22]
OWASP ZAP	Dynamic Application Security Testing (DAST) framework	Powerful, free, and open-source. Excellent for interactive testing, fuzzing, and finding complex web app vulnerabilities (SQLi, XSS).[49]	Steeper learning curve than Nikto. Can be slower due to comprehensive crawling and active scanning.[50]	In-depth, manual, and automated testing of a web application's functionality and business logic.[52]

Chapter 8: The Ethical Scanner - Legal and

Responsible Use

The power of a tool like Nikto comes with significant responsibility. By its very nature, it is an intrusive tool that probes for weaknesses in computer systems. Its use is governed by a strict set of legal and ethical principles that every security professional must understand and adhere to. Wielding this tool without proper authorization and a strong ethical framework is not only unprofessional but can also lead to severe legal consequences. This chapter provides the essential guardrails for the legal and responsible use of Nikto.

8.1 The Golden Rule: Authorization is Mandatory

The single most important principle governing the use of any vulnerability scanner is **authorization**. A security professional must **never** scan a system, network, or application that they do not own or for which they have not received explicit, prior, and preferably written permission to test.[7]
Scanning a third-party system without consent can be interpreted as a hostile act and may constitute a criminal offense under various jurisdictions. In the United States, such activity could be prosecuted under the Computer Fraud and Abuse Act (CFAA). In the United Kingdom, it could fall under the Computer Misuse Act (CMA).[56] These laws often carry severe penalties, including substantial fines and imprisonment. It is critical to understand that scanning with "good intentions"—for example, to find vulnerabilities and report them to the owner—is not a valid legal defense. The act of unauthorized probing itself is often what is prohibited.[56]

8.2 Principles of Responsible Disclosure

When a vulnerability is discovered during an *authorized* scan, the process of reporting it is just as important as the discovery itself. The industry standard for this process is known as "responsible disclosure" or "coordinated vulnerability disclosure." This model is designed to ensure that vulnerabilities are fixed without putting the public at risk. Its core principles are:

- **Prompt and Private Reporting:** As soon as a vulnerability is confirmed, it should be reported privately to the affected organization. This should be done through their designated security channel, which could be a specific email address (e.g., security@example.com), a formal bug bounty program platform, or a Vulnerability Disclosure Program (VDP) portal.[59]
- **Provide Detailed, Actionable Information:** The vulnerability report must contain sufficient detail to allow the organization's security and development teams to understand, reproduce, and ultimately fix the issue. A good report includes the specific URL or IP address, a clear description of the vulnerability, step-by-step instructions to

reproduce it (with screenshots or proof-of-concept scripts where helpful), and an assessment of its potential impact.[61]

- **Allow a Reasonable Time for Remediation:** A cornerstone of responsible disclosure is patience. The researcher must give the organization a reasonable timeframe to develop, test, and deploy a patch for the vulnerability before any public disclosure is made. A period of 90 days is a widely accepted industry standard.[60]

This model stands in stark contrast to "full disclosure," where a vulnerability is publicized immediately upon discovery. Full disclosure is highly controversial because it provides details of the flaw to malicious actors before a fix is available, often increasing rather than decreasing risk for the public.[59] It is typically considered a measure of last resort.

8.3 Understanding Vulnerability Disclosure Policies (VDPs)

To formalize the process of receiving vulnerability reports, many organizations and government agencies now publish a Vulnerability Disclosure Policy (VDP). These documents provide clear rules of engagement for security researchers wishing to test their systems. Analyzing the VDPs from entities like the U.S. Department of Health and Human Services (HHS) and the Department of Commerce reveals several common components [60]:

- **Scope:** The VDP will clearly define which systems, domains, and applications are in-scope for testing and which are explicitly out-of-scope (e.g., third-party vendor systems).[60]
- **Rules of Engagement:** These are the ground rules for testing. They specify what actions are permitted (e.g., identifying and documenting vulnerabilities) and what actions are strictly forbidden. Prohibited actions almost always include data exfiltration, denial of service (DoS) attacks, modification or destruction of data, and social engineering.[62]
- **Safe Harbor:** This is a critical clause. A safe harbor statement provides a legal assurance from the organization that security researchers who conduct their activities in good faith and strictly adhere to the VDP will not face legal action from that organization.[60]
- **Reporting Mechanism:** The VDP will specify the official and sole channel through which vulnerability reports should be submitted, such as a dedicated web portal.[60]

Before conducting any scan, a researcher must thoroughly read and agree to the target's VDP, if one exists.

8.4 Minimizing Impact and Maintaining Professionalism

Ethical scanning extends beyond just having permission. Even during an authorized test, a professional must act to minimize any potential disruption to the target system. This includes:

- **Targeting Scans Tightly:** Avoid broad, unnecessary scans. If the goal is to test a web

server, scan only the relevant port (e.g., -p 443), rather than all 65,536 ports on the host.[57]

- **Managing Scan Speed:** Use options like -Pause to introduce a delay between requests, which can reduce the load on the target server, especially if it is a low-powered or embedded device.[35]
- **Timing Scans Appropriately:** Whenever possible, conduct intrusive scans during off-peak hours or within a maintenance window agreed upon with the system owner to avoid impacting business operations.
- **Meticulous Documentation:** All aspects of the engagement should be documented, including the written authorization, the scope of the test, the tools and methods used, and all findings. This documentation is crucial for demonstrating compliance, professionalism, and ethical intent.[56]

Chapter 9: Nikto in Action - Real-World Scenarios and Case Studies

Theory and command-line references are essential, but the true measure of a tool is its application in the real world. This chapter synthesizes the knowledge from the preceding sections into a practical, narrative-driven case study. By walking through a realistic penetration testing scenario, we can see how Nikto's findings are not just a list of vulnerabilities, but a series of clues that guide an analyst from initial reconnaissance to a significant security breakthrough.

9.1 Case Study: Assessing the Damn Vulnerable Web Application (DVWA)

For this demonstration, we will use a scenario based on scanning the Damn Vulnerable Web Application (DVWA), a web application intentionally designed with security flaws for training purposes. This allows us to observe how Nikto performs against a target with known, discoverable vulnerabilities.[8]

Step 1: Initial Scan

The engagement begins with a standard Nikto scan against the IP address of the server hosting DVWA. The command is simple and direct:

Bash

perl nikto.pl -h <DVWA_IP>

Step 2: Interpreting Initial Findings

The scan output begins to populate the screen. The analyst examines each line, building a mental model of the server's security posture. Early findings include [8]:

- + Server: Apache/2.2.8 (Ubuntu): The server banner is immediately noted. An analyst would cross-reference this specific version against CVE databases to check for public exploits.
- + robots.txt found: The robots.txt file is present. While its content in this case (Disallow: /) is not immediately revealing, it is always flagged for manual review.
- + The web server version is outdated: Nikto explicitly flags the Apache version as obsolete. This is a high-priority finding, indicating the server is likely missing years of security patches.
- + TRACE method is allowed: The HTTP TRACE method, used for debugging, is enabled. This can be abused in some circumstances to perform Cross-Site Tracing (XST) attacks to steal cookie information.

At this stage, the findings point to poor server hygiene but have not yet revealed a direct path to compromise.

Step 3: Following the Scent - Investigating a Key Vulnerability

As the scan continues, a more promising finding appears:
+ /dvwa/config/: Directory indexing is enabled. 8
This is a critical lead. Directory indexing means the web server is configured to display a list of all files and folders within that directory if a default index page (like index.html) is not present. The /config/ directory, by its name, is highly likely to contain sensitive configuration files. An experienced tester would immediately pivot from the automated scan to manual investigation of this directory.

Step 4: The Breakthrough

The analyst navigates to http://<DVWA_IP>/dvwa/config/ in a web browser. The server responds with a list of files, one of which is config.inc.php. This file almost certainly contains the database connection settings for the application. While clicking the file might execute it, the analyst recalls a common developer practice: leaving backup files in the same directory. By appending a tilde (~) to the filename—a common convention for backup files on Linux systems—the analyst requests http://<DVWA_IP>/dvwa/config/config.inc.php~.

The server, instead of executing the PHP code, serves the raw source code of the backup file as plain text. Within this file, the analyst finds the application's database credentials in cleartext [8]:

- **Database:** dvwa
- **Database User:** root
- **Database Password:** (blank/null)

Step 5: Impact Assessment and Next Steps

The discovery of valid database credentials represents a critical breach. The analyst has moved from external reconnaissance to having direct access to the application's data store. In a professional penetration test report, this finding would be rated as "Critical" or "High" severity.

The next steps in the engagement are now clear:

1. Use the discovered credentials to connect to the MySQL database.
2. Enumerate the database structure, dump sensitive user data, and look for opportunities to leverage database access to gain further control.
3. Attempt to use the database access to write a file to the web server's file system, potentially achieving remote code execution.

This case study perfectly illustrates Nikto's role. It did not exploit the system itself, but its discovery of the directory indexing vulnerability was the crucial "laser pointer" that guided the human tester directly to the flaw that led to the compromise.[5]

9.2 Integrating Nikto into a Broader Workflow

This scenario highlights the standard workflow where Nikto is a key component. The process involves:

1. **Reconnaissance:** Using a tool like Nikto (or Nmap followed by Nikto) to perform broad, automated scanning to identify potential weaknesses.[5]
2. **Analysis:** A human analyst reviews the scanner's output, filtering out noise and identifying high-value leads.
3. **Manual Verification and Exploitation:** The analyst uses the leads from the scan to perform targeted manual testing or to configure an exploitation tool (like Metasploit) to take advantage of the discovered vulnerability.[20]

Nikto excels at the first step, making the subsequent steps more efficient and effective.

9.3 Final Thoughts: The Enduring Value of a Classic Tool

The landscape of cybersecurity is ever-changing, with new tools and techniques emerging

constantly. Yet, for over two decades, Nikto has maintained its place as an essential utility. Its value does not lie in its complexity or its ability to bypass modern defenses. Instead, its strength is its simplicity, its speed, and its exhaustive, community-curated knowledge of the fundamental mistakes that leave web servers vulnerable.

For quickly and effectively identifying server misconfigurations, outdated software, and the dangerous remnants of default installations, Nikto is unparalleled. It serves as a rapid-fire audit, a baseline check for security hygiene, and a powerful reconnaissance tool that can uncover the first thread that, when pulled, can unravel an entire system's security. In the hands of a knowledgeable and ethical professional, Nikto remains an indispensable sentinel, standing guard and pointing the way toward a more secure digital world.[21]

Works cited

1. www.cisa.gov, accessed June 18, 2025, https://www.cisa.gov/resources-tools/services/nikto#:~:text=Nikto%20is%20an%20open%20source,versions%20of%20web%20server%20software.
2. Nikto - CISA, accessed June 18, 2025, https://www.cisa.gov/resources-tools/services/nikto
3. Nikto (vulnerability scanner) - Wikipedia, accessed June 18, 2025, https://en.wikipedia.org/wiki/Nikto_(vulnerability_scanner)
4. What is Nikto and it's usages ? | GeeksforGeeks, accessed June 18, 2025, https://www.geeksforgeeks.org/what-is-nikto-and-its-usages/
5. How to Scan for Vulnerabilities on Any Website Using Nikto :: Null Byte, accessed June 18, 2025, https://null-byte.wonderhowto.com/how-to/scan-for-vulnerabilities-any-website-using-nikto-0151729/
6. 1.2.3.3. Nikto | Security Guide | Red Hat Enterprise Linux | 6, accessed June 18, 2025, https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/6/html/security_guide/sect-security_guide-evaluating_the_tools-nikto
7. Web Vulnerability Scanning with Nikto - Infosec Train, accessed June 18, 2025, https://www.infosectrain.com/blog/web-vulnerability-scanning-with-nikto/
8. Scanning Web Servers With Nikto – Penetration Testing Lab, accessed June 18, 2025, https://pentestlab.blog/2012/12/06/scanning-web-servers-with-nikto/
9. Nikto 2.5 - CIRT.net, accessed June 18, 2025, https://www.cirt.net/Nikto2
10. Nikto vs Nmap - Which to use and how to take the headache out, accessed June 18, 2025, https://www.hackingloops.com/nikto-and-nmap/
11. History & Trivia · sullo/nikto Wiki · GitHub, accessed June 18, 2025, https://github.com/sullo/nikto/wiki/History-&-Trivia
12. Nikto Web Scanner Guide and Examples - Sohvaxus, accessed June 18, 2025, https://sohvaxus.github.io/content/nikto.html
13. Who is Nikto? MW2 And MW3 Lore - YouTube, accessed June 18, 2025, https://www.youtube.com/shorts/SSshm4rdSdU
14. Nikto Species Biology, Society, and History - YouTube, accessed June 18, 2025, https://www.youtube.com/watch?v=i5If-m8-K0U

15. Nikto - ::Star Wars Combine::, accessed June 18, 2025, https://www.swcombine.com/rules/?Races&ID=77
16. The Story of Nikto ~ Who is Nikto codm? - YouTube, accessed June 18, 2025, https://www.youtube.com/watch?v=GsEDLyfE5II
17. nikto_usage.txt - GitHub, accessed June 18, 2025, https://github.com/lattera/nikto/blob/master/nikto-1.x/nikto-1.36/docs/nikto_usage.txt
18. Nikto Tutorial - Installation to Effective Targeting | HackerTarget.com, accessed June 18, 2025, https://hackertarget.com/nikto-tutorial/
19. nikto_usage.txt - GitHub, accessed June 18, 2025, https://github.com/lattera/nikto/blob/master/nikto-1.x/nikto-1.10/docs/nikto_usage.txt
20. Web Server Scanning With Nikto | Tutorials - Armur AI, accessed June 18, 2025, https://armur.ai/tutorials/nikto/nikto/what_is_nikto/
21. Vulnerability Assessment with Nikto: Reliable Results Using Only FOSS Tools - DevDigest, accessed June 18, 2025, https://www.samgalope.dev/2024/10/21/comparative-analysis-of-nikto-and-other-vulnerability-scanners-for-vulnerability-assessment/
22. Tool Wars | Comparing Nmap, Nessus, and Nikto | Recon Tools for Ethical Hackers, accessed June 18, 2025, https://www.webasha.com/blog/tool-wars-comparing-nmap-nessus-and-nikto-recon-tools-for-ethical-hackers
23. Why Nikto Web Vulnerability Scanner is Easily Detected - With Wireshark and NMAP, accessed June 18, 2025, https://www.youtube.com/watch?v=g76F3ds4gUc
24. Installing Nikto on Windows - Mad Irish, accessed June 18, 2025, https://www.madirish.net/185
25. nikto | Kali Linux Tools, accessed June 18, 2025, https://www.kali.org/tools/nikto/
26. Installing and updating Nikto on Linux - Mastering Linux Security and Hardening [Book] - O'Reilly Media, accessed June 18, 2025, https://www.oreilly.com/library/view/mastering-linux-security/9781788620307/2a39b39c-b87e-479c-a2df-2590cac136a9.xhtml
27. How to Install and Use Nikto on Debian 10 – Step-by-Step (2025), accessed June 18, 2025, https://eldernode.com/tutorials/nikto-on-debian-10/
28. Hacking: Using a Macbook and Nikto to Scan your Local Network, accessed June 18, 2025, https://andrewbaker.ninja/2023/01/25/hacking-using-my-macbook-and-nikto-to-scan-your-local-network/
29. nikto - Homebrew Formulae, accessed June 18, 2025, https://formulae.brew.sh/formula/nikto
30. Home · sullo/nikto Wiki - GitHub, accessed June 18, 2025, https://github.com/sullo/nikto/wiki
31. Nikto - Hackviser, accessed June 18, 2025, https://hackviser.com/tactics/tools/nikto
32. nikto - andrewjkerr/security-cheatsheets - GitHub, accessed June 18, 2025, https://github.com/andrewjkerr/security-cheatsheets/blob/master/nikto

33. Nikto - secureCodeBox, accessed June 18, 2025, https://www.securecodebox.io/docs/scanners/nikto
34. Nikto Cheat Sheet - Commands & Examples - HighOn.Coffee, accessed June 18, 2025, https://highon.coffee/blog/nikto-cheat-sheet/
35. nikto - Scan web server for known vulnerabilities - Ubuntu Manpage, accessed June 18, 2025, https://manpages.ubuntu.com/manpages/focal/man1/nikto.1.html
36. Nikto + Terminal Mastery: Understanding Your Website Security Scan Results - DevDigest, accessed June 18, 2025, https://www.samgalope.dev/2024/10/21/understanding-niktos-scan-output-for-website-security-scan/
37. Scan Web Servers in Nikto - LabEx, accessed June 18, 2025, https://labex.io/tutorials/nmap-scan-web-servers-in-nikto-549948
38. The Nikto scanner and Microsoft IIS., accessed June 18, 2025, https://learn.microsoft.com/en-us/answers/questions/576424/the-nikto-scanner-and-microsoft-iis
39. Nikto: locating vulnerability details from OSVDB IDs : r/oscp - Reddit, accessed June 18, 2025, https://www.reddit.com/r/oscp/comments/dn1yl4/nikto_locating_vulnerability_details_from_osvdb/
40. nikto's results differ from other vulnerability scanners , is it more or less accurate?, accessed June 18, 2025, https://security.stackexchange.com/questions/185138/niktos-results-differ-from-other-vulnerability-scanners-is-it-more-or-less-ac
41. Vulnerability scanning tools - Security - Hak5 Forums, accessed June 18, 2025, https://forums.hak5.org/topic/40905-vulnerability-scanning-tools/
42. nikto - secureCodeBox - Artifact Hub, accessed June 18, 2025, https://artifacthub.io/packages/helm/securecodebox/nikto/2.9.1
43. Nikto | secureCodeBox, accessed June 18, 2025, https://www.securecodebox.io/docs/scanners/nikto/
44. Security Testing in the Terminal: How to Hack Smarter with Nikto ..., accessed June 18, 2025, https://www.samgalope.dev/2024/10/21/enhancing-nikto-with-plugins-for-comprehensive-security-testing/
45. Writing Plug-ins for the Nikto Vulnerability Scanner - Tutorial - Vskills, accessed June 18, 2025, https://www.vskills.in/certification/tutorial/writing-plug-ins-for-the-nikto-vulnerability-scanner/
46. Selecting Plugins · sullo/nikto Wiki - GitHub, accessed June 18, 2025, https://github.com/sullo/nikto/wiki/Selecting-Plugins
47. Updating · sullo/nikto Wiki · GitHub, accessed June 18, 2025, https://github.com/sullo/nikto/wiki/Updating
48. Nmap and 12 useful NSE scripts - research.securitum.com, accessed June 18, 2025, https://research.securitum.com/nmap-and-12-useful-nse-scripts/
49. 6 Best DAST Tools You Should Know in 2024 - Bright Security, accessed June 18, 2025, https://www.brightsec.com/blog/dast-tools/

50. Top 7 Open Source Vulnerability Scanning Tools in 2025, accessed June 18, 2025, https://research.aimultiple.com/open-source-vulnerability-scanning-tools/
51. Top 8 penetration testing tools | Snyk, accessed June 18, 2025, https://snyk.io/blog/top-8-penetration-testing-tools/
52. Top 10 Web Security Tools Every Developer Should Know, accessed June 18, 2025, https://blog.pixelfreestudio.com/top-10-web-security-tools-every-developer-should-know/
53. edwinmelero/Nikto-and-Burp-Suite: Analyzing Output from Web Application Assessment Tools - GitHub, accessed June 18, 2025, https://github.com/edwinmelero/Nikto-and-Burp-Suite
54. Currently, I am studying the OWASP ZAP tool. What are the pros and cons of using this tool?, accessed June 18, 2025, https://www.quora.com/Currently-I-am-studying-the-OWASP-ZAP-tool-What-are-the-pros-and-cons-of-using-this-tool
55. A Beginner's Guide to Nikto: Web Server Scanning Made Simple - DevDigest, accessed June 18, 2025, https://www.samgalope.dev/a-beginners-guide-to-ethical-hacking-with-termux/introduction-to-nikto-a-comprehensive-guide-to-web-server-vulnerability-scanning/
56. Ethical Considerations and Legal Aspects of Network Scanning with ..., accessed June 18, 2025, https://www.siberoloji.com/ethical-considerations-and-legal-aspects-of-network-scanning-with-nmap/
57. Legal Issues | Nmap Network Scanning, accessed June 18, 2025, https://nmap.org/book/legal-issues.html
58. The Ethics and Legality of Port Scanning - GIAC Certifications, accessed June 18, 2025, https://www.giac.org/paper/gsec/1237/ethics-legality-port-scanning/102383
59. What is Responsible Disclosure? - Bugcrowd, accessed June 18, 2025, https://www.bugcrowd.com/resources/guide/what-is-responsible-disclosure/
60. Vulnerability Disclosure Policy - U.S. Department of Commerce, accessed June 18, 2025, https://www.commerce.gov/vulnerability-disclosure-policy
61. Vulnerability Disclosure - OWASP Cheat Sheet Series, accessed June 18, 2025, https://cheatsheetseries.owasp.org/cheatsheets/Vulnerability_Disclosure_Cheat_Sheet.html
62. Vulnerability Disclosure Policy - HHS.gov, accessed June 18, 2025, https://www.hhs.gov/vulnerability-disclosure-policy/index.html
63. Vulnerability Disclosure Policy - Secret Service, accessed June 18, 2025, https://www.secretservice.gov/Vulnerability_Disclosure_Policy
64. Hack Web Servers using Nikto and WhatWeb: Web Scanning Unleashed - YouTube, accessed June 18, 2025, https://www.youtube.com/watch?v=YxZOFQNC4QY
65. vulnerabilities scan : r/HowToHack - Reddit, accessed June 18, 2025, https://www.reddit.com/r/HowToHack/comments/1e4yw4d/vulnerabilities_scan/