Nikto Exploitation of Web Vulnerabilities

An S.O.S Security Report

Dive into the depths of cybersecurity with this comprehensive report on Nikto, an open-source web server scanner. This project explores vital vulnerability scanning, targeting common web exploits like Command Injection, Cross-Site Scripting, and SQL Injections. We aim to empower participants by demonstrating robust mitigation techniques, ensuring they are equipped to enhance their security posture in an ever-evolving digital landscape. Join us in building a more secure digital future.

Aug 13, 2024

Sink or Swim (S.O.S) Security

"Secure the Tide, Anchor Your Data"

Introduction

My name is Chet Flowers, and I'm presenting this project focused on exploiting web vulnerabilities using Nikto and DVWA.

- Project Foundation: Explored the historical context and development of cybersecurity tools, which set the groundwork for applying Nikto.
- Technical Focus: Demonstrated practical attack scenarios and detailed the exploitation of vulnerabilities like SQL Injection, XSS, and Command Injection.
- Mitigation and Strategy: Analyzed vulnerabilities, developed prevention tactics, and emphasized securing systems through strategic insight.
- Conclusion: Summarized findings and highlighted the importance of continued learning and vigilance in cybersecurity.

Hello, I'm Chet

Operations and Program Leadership

"Secure the Tide, Anchor Your Data"

Problem Statement & Goal

Even as **cybersecurity** evolves with cutting-edge tools and techniques, age-old **vulnerabilities** like **XSS**, **SQL Injection**, and **Command Injection** continue to haunt web applications. Our recent **Nikto** project revealed a troubling number of these issues, underscoring the persistent challenge of defending against increasingly sophisticated **cyber threats** in real-world environments. Despite our best efforts, these **attack vectors** remain stubborn adversaries, reminding us that the battle for **secure systems** is far from over.

- Boost awareness and deepen understanding of the common web vulnerabilities that Nikto flags, from legacy risks to more complex threats.
- Showcase practical mitigation strategies specifically tailored to real-world scenarios, turning knowledge into action.
- Elevate the security posture of web applications by promoting a proactive defense using Nikto and other robust cybersecurity tools.

Table of Contents

My First Role

Historical & Foundational Cybersecurity Researcher

The Role

A “Historical & Foundational Cybersecurity Researcher” is like a digital archaeologist who digs through the past to help shape the future of cybersecurity. They explore the roots of cyber threats, studying legendary hacks, classic vulnerabilities, and the evolution of defense techniques. By understanding how cyberattacks and protection strategies have evolved, they uncover timeless lessons and forgotten tactics that still hold value today. This role involves connecting the dots between yesterday’s exploits and today’s innovations, ensuring that modern cybersecurity is built on a solid, time-tested foundation. Think of them as the guardians of cybersecurity history, using the past to keep the future safe!

Nikto Project: Introduction, Background & Application

- What is Nikto?
- The History & Development of Nikto
- Key Features
- Usage & Practical Applications
- Vulnerabilities Associated with Nikto
- Why use Nikto?

Nikto What is?

Nikto is

- Open-source web server scanner: Nikto is freely available and open-source, making it accessible for security professionals and enthusiasts.
- Comprehensive scanning: Performs tests against web servers for various vulnerabilities and misconfigurations.
- Extensive database: Scans for over 6700 potentially dangerous files and scripts, helping detect common security issues.
- Server version checks: Identifies versions for over 1250 servers and checks for version-specific problems across more than 270 servers.
- Server configuration analysis: Detects the presence of multiple index files and checks for potentially risky HTTP server options.
- Web server and software identification: Attempts to identify installed web servers and associated software.
- Security-focused tool: Useful for identifying potential vulnerabilities, misconfigurations, and outdated software in web applications.
- Regular updates: Frequently updated to include new vulnerabilities and ensure ongoing relevance.
- Widely used in security assessments: Commonly employed by security analysts, penetration testers, and system administrators.

Nikto History & Development

The tool was designed to identify potential vulnerabilities and security issues in web servers. It was developed to be an easy-to-use tool for web security assessments, providing comprehensive scanning capabilities.

- **2001:** Chris Sullo created **Nikto**, a popular web vulnerability scanner.
- **2000:** Launched **CIRT.net** and the **Default Password Database**.
- **2004:** Co-founded the **Open Source Vulnerability Database (OSVDB)**.
- **2001:** Released **Nikto 1.00 Beta**; ongoing updates since then.
- **2007:** Released **Nikto 2.0** with significant improvements.
- Continues to develop Nikto and co-founded **RVAsec cybersecurity conference** in 2011.

Nikto Key Features

Key Features

Vulnerability Detection:

- **SQL Injection**: Scans for SQL injection vulnerabilities.
- **Cross-Site Scripting (XSS)**: Detects XSS vulnerabilities that allow script injection.
- **Command Injection**: Identifies flaws where arbitrary commands can be executed.

Comprehensive Testing:

- **Server Misconfigurations**: Checks for common server misconfigurations.
- **Default Files and Scripts**: Looks for potentially risky default files and scripts.
- **Outdated Software**: Identifies outdated software versions with known vulnerabilities.

Plugins and Customization:

- **Custom Plugins**: Supports custom plugins for additional tests.
- **Integration**: Can be integrated with other security tools.

Reporting and Logging:

- **Detailed Reports**: Provides detailed reports in various formats.
- **Activity Logs**: Logs all scan activities for audits and analysis.

Supporting Components:

- **Pre-Defined Payloads and Tests**: Nikto offers a variety of predefined payloads and tests to detect vulnerabilities like SQL Injection, XSS, and Command Injection.
- **Extensive Database**: It has a regularly updated database of known vulnerabilities for various web server types and configurations.

Nikto Uses & Practical Applications

Usage:

- **Command Line Tool**: Nikto is primarily a command-line tool, making it suitable for use in scripts and automated testing environments.
- **Cross-Platform**: It is available on multiple platforms, including Windows, Linux, and macOS.

Practical Applications:

Penetration Testing	Compliance Testing	Security Audits
Nikto is widely used by penetration testers to identify potential entry points and vulnerabilities in web applications.	Organizations use Nikto to ensure their web applications meet security compliance requirements.	IT security teams utilize Nikto to perform regular security audits of their web infrastructure.

Example Command:

```
nikto -h http://example.com
```

This command runs a basic scan on the specified target.

Nikto & SQL Injection (SQLi)

SQL Injection

SQL Injection (SQLi) is a code injection technique that exploits a vulnerability in a web application's software by manipulating SQL queries. This occurs when user input is not properly sanitized, allowing an attacker to interfere with the queries an application makes to its database.

How Nikto Identifies SQL Injection (SQLi)

Nikto identifies SQL Injection vulnerabilities by sending payloads designed to trigger SQL errors or unusual behavior in the database. These payloads are crafted to exploit common SQL Injection points such as form fields, query parameters, and cookies.

Examples of Nikto Detection

- **Error-based SQL Injection**: Nikto attempts to induce SQL errors by inputting special characters and SQL syntax (e.g., ' OR 1=1 --).
- **Union-based SQL Injection**: By attempting to union additional SQL queries (e.g., UNION SELECT null, null, user(), database()).

Nikto & Cross Site Scripting (XSS)

Cross-Site Scripting (XSS)

A security vulnerability that allows an attacker to inject malicious scripts into content from otherwise trusted websites. These scripts can then execute in the context of the user's browser, leading to session hijacking, defacement, or the spread of malware.

How Nikto Identifies Cross-Site Scripting (XSS)

Nikto scans for XSS vulnerabilities by injecting common XSS payloads into input fields and URL parameters. If the injected script is reflected back in the response, it indicates a potential XSS vulnerability.

This script will display a pop-up alert with the message “XSS Attack!” when executed. **<script>alert('XSS Attack!');</script>**

Reflected XSS:

- The script is injected into a URL or form input field and immediately executed when the page is rendered.

Stored XSS:

- The script is stored in a database and executed every time the affected page is loaded by users.

!

Nikto & Command Injection

Command Injection

A type of vulnerability where an attacker can execute arbitrary commands on the host operating system via a vulnerable application. This typically occurs when user input is passed directly to a system shell without proper validation or sanitization.

How Nikto Identifies Command Injection

Nikto identifies Command Injection vulnerabilities by sending input designed to break out of the expected input context and execute system commands. These payloads often include characters and syntax specific to command-line interfaces (e.g., ; ls -la).

Examples of Nikto Detection

- **Basic Command Injection**: Sending payloads like '; cat' & '/etc/passwd' to see if the contents of the file are returned.
- **Complex Command Injection**: Using logical operators and command chaining (e.g., | whoami).

!

Nikto Tools

COMPETITIVE TOOLS FOR NIKTO

Tool	Strengths	Key Features	Consideration	Included in Kali Linux
OWASP ZAP	Popular open-source tool for finding vulnerabilities in web applications.	Extensive GUI support and automation features, offering broader capabilities than Nikto.	Focuses more on web applications rather than just web servers.	Yes.
Burp Suite	Comprehensive web vulnerability scanner used by professionals.	Includes features for manual testing and advanced scanning, making it highly versatile.	More complex and expensive, often used by those with advanced knowledge.	Yes (Community Edition); Pro version requires separate installation
Acunetix	Commercial tool known for robust scanning capabilities.	Detects over 6,500 vulnerabilities, with detailed reporting and CI/CD pipeline integration.	High price tag makes it less accessible for users on a budget.	No
Nessus	Primarily known for network vulnerability scanning but also includes web application scanning.	Offers detailed reporting and compliance checks.	Requires a license, as it is not open-source.	No

Nikto
vs.
OWASP ZAP

OWASP ZAP vs. NIKTO

Comparison Aspect	OWASP ZAP	Nikto
Primary Focus	Web application security scanner, focusing on finding vulnerabilities like SQL Injection and XSS.	Choose when the primary goal is to secure web applications.
Features	Offers automated crawling, spidering, and proxying, with both active and passive scanning capabilities.	Specializes in server-specific scans, focusing on server integrity and misconfigurations.
Versatility	More versatile with broader features, making it ideal for comprehensive web application testing.	Better suited for quick identification of server-side vulnerabilities.
Best Use Case	Ideal for deep, comprehensive scanning of web applications, finding complex vulnerabilities.	Ideal for quick scans focused on server security, especially useful for legacy systems.
When to Choose	Choose when the primary goal is to secure web applications.	Choose when the focus is on securing the web server rather than the application itself.

The Choice Is Made

Why choose Nikto?

- **Open-Source and Free:** Nikto is fully open-source, meaning it's free to use, modify, and improve without any licensing fees.
- **Quick and Efficient:** Designed for straightforward, fast scanning of web servers, Nikto is perfect for users needing rapid assessments without the complexity of advanced tools like OWASP ZAP.
- **Extensive Vulnerability Database:** With a robust database of known vulnerabilities, misconfigurations, and default settings, Nikto effectively identifies common issues across various web servers.
- **Strong Community Support:** As an open-source tool, Nikto benefits from a global community that regularly contributes updates, ensuring it remains up-to-date with the latest security threats.

My Second Role

Technical Researcher for Malicious Use & Payloads

The Role

A “Technical Researcher for Malicious Use & Payloads” is like a cyber-detective specializing in the dark arts of malware and exploit delivery. Their mission? Dive deep into the tactics, techniques, and payloads used by bad actors to understand how cyberattacks are engineered and deployed. They analyze everything from sneaky phishing schemes to sophisticated payloads like ransomware, reverse-engineering code to discover the methods behind malicious intent. Armed with this knowledge, they help build defenses and teach others how to spot and block emerging threats. In short, they think like attackers to outsmart them, staying one step ahead in the cat-and-mouse game of cybersecurity!

Nikto Project: DVWA & Nikto in Action

- A Guide: Setup and Utilization of DVWA
- The Nikto Scan: Findings and Analysis
- Demonstration: Command Injection

DVWA Setup On Kali

- Intro
- Setup
- Configuration

DVWA Setup on Kali Linux

Introduction to DVWA:

- Damn Vulnerable Web Application (DVWA) is an intentionally insecure web application.
- Designed to help security professionals and developers practice and understand common web vulnerabilities.
- Supports testing of techniques like SQL injection, XSS, and command injection.
- Provides a controlled environment for safe vulnerability exploration.
- Widely used for educational purposes to understand the risks and impacts of security flaws.

Setup Requirements:

- **Install Required Tools:** Set up Apache, MariaDB, and PHP on Kali Linux.
- **Clone DVWA:** Download and place DVWA in the web server directory.

Configuration and Access:

- **Configure Database:** Create a dvwa database and user in MariaDB, and update DVWA's config file.
- **Start Services:** Launch Apache and MariaDB.
- **Access DVWA:** Initialize the database and set the security level to 'Low'.

Setting Up DVWA:

Configuring MariaDB and Initializing the Database

What is MariaDB?

- **MariaDB Overview**: Open-source RDBMS, forked from MySQL, known for high performance, scalability, and advanced security features.
- **Key Features**: Provides a robust and reliable solution for managing databases in web applications and data-intensive environments.

Install and Configure Services:

- Install Apache, MariaDB, and PHP on Kali Linux.
- Clone DVWA from GitHub into the web server directory

Database Setup:

- Execute SQL commands to create the dvwa database and a user with full privileges.
- Update DVWA's **config.inc.php** file with the correct database credentials.

Initialize and Test:

- Navigate to DVWA's setup page in your browser and click "Create / Reset Database".”
- Set the security level to 'Low' and begin testing vulnerabilities.

What is MariaDB?

Configure & Start

- Install and Configure
- Setup
- Initialize and Test

Configure & Start (cont.)

- Install
- Start
- Status
- Database

```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get update
sudo apt-get install apache2 mariadb-server php libapache2-mod-php php-mysql

┌──(kali㉿kali)-[~]
└─$ sudo service mariadb start


┌──(kali㉿kali)-[~]
└─$ sudo service mariadb status

● mariadb.service - MariaDB 11.4.2 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; disabled; preset: disabled)
     Active: active (running) since Mon 2024-08-12 00:28:00 EDT; 8s ago
┌──(kali㉿kali)-[~]
└─$ sudo mysql -u root

Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 43
Server version: 11.4.2-MariaDB-4 Debian n/a

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Support MariaDB developers by giving a star at https://github.com/MariaDB/server
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE dvwa;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> CREATE USER 'dvwa'@'localhost' IDENTIFIED BY 'p@ssw0rd';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON dvwa.* TO 'dvwa'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> exit;
Bye
```

Configure & Start (cont.)

- Clone
- Permissions
- Configuration

```
└─$ sudo git clone https://github.com/digininja/DVWA.git

Cloning into 'DVWA'...
remote: Enumerating objects: 4593, done.
remote: Counting objects: 100% (143/143), done.
remote: Compressing objects: 100% (104/104), done.
remote: Total 4593 (delta 60), reused 95 (delta 38), pack-reused 4450
Receiving objects: 100% (4593/4593), 2.31 MiB | 17.81 MiB/s, done.
Resolving deltas: 100% (2171/2171), done.
```

```
┌──(kali㉿kali)-[/var/www/html]
└─$ sudo chown -R www-data:www-data /var/www/html/DVWA/
sudo chmod -R 755 /var/www/html/DVWA/
```

```
┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ sudo cp config.inc.php.dist config.inc.php
```

```
┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ ls
config.inc.php  config.inc.php.dist
```

```
┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ sudo nano /var/www/html/DVWA/config/config.inc.php
```

```
$_DVWA = array();
$_DVWA[ 'db_server' ]   = getenv('DB_SERVER') ?: '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ]     = 'dvwa';
$_DVWA[ 'db_password' ] = 'p@ssw0rd';
$_DVWA[ 'db_port']      = '3306';
```

Setup & Test

- Browser
- Database
- Credentials

Final Setup and Testing

Access DVWA in the Browser:

- **Launch DVWA:** Open your web browser and navigate to http://localhost/DVWA/ to access the DVWA web interface.
- **Setup Initialization:** Upon first access, DVWA will prompt you to complete the setup by initializing the database.

Create/Reset the Database:

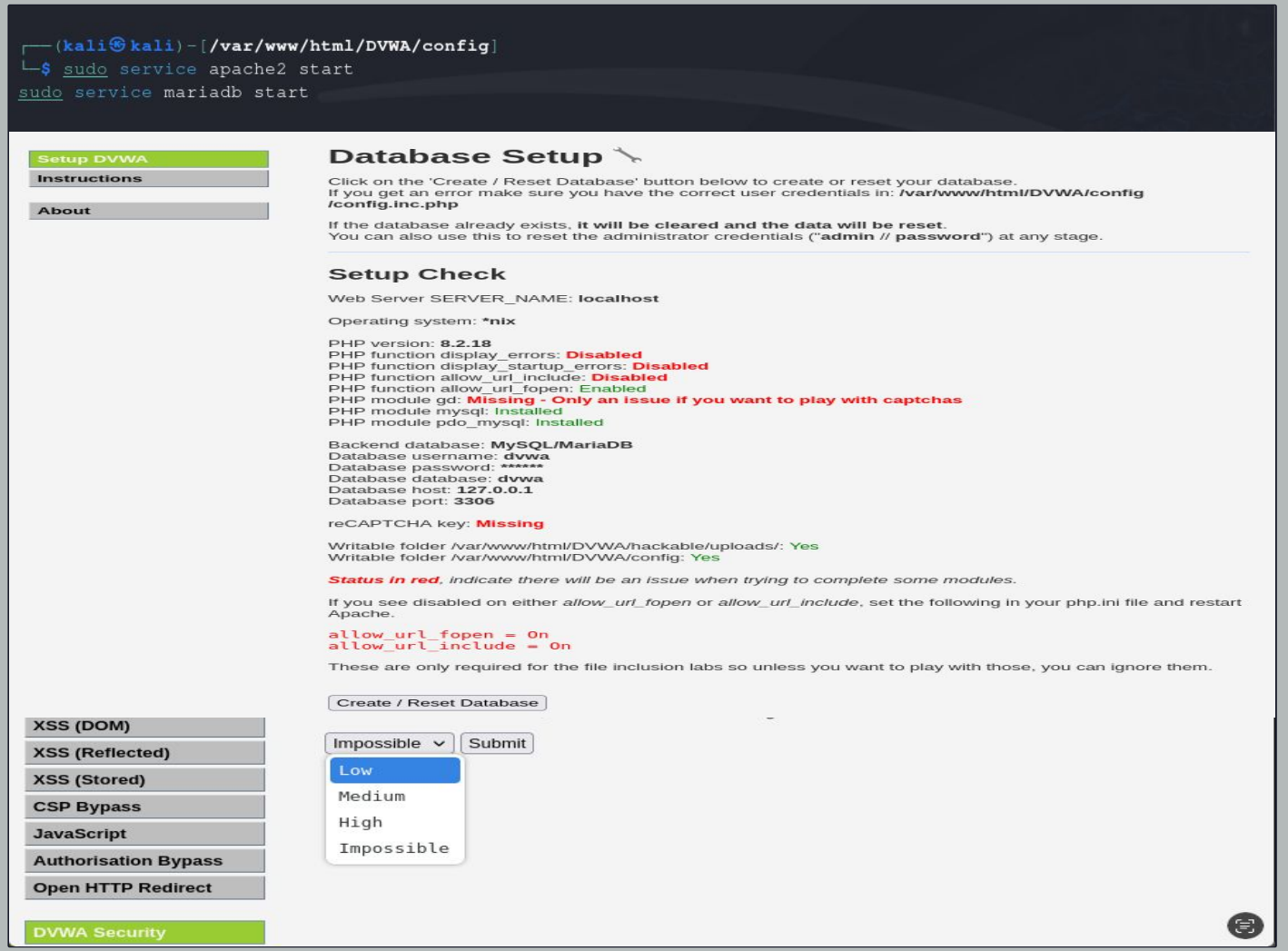
- **Database Setup:** Click the "Create / Reset Database" button on the DVWA setup page, which will create the necessary tables and populate them with default data in the dvwa database.
- **Confirmation:** After the database is successfully created or reset, a confirmation message will appear, indicating the DVWA environment is ready for use.

Log in with Default Credentials:

- **Default Authentication:** Use the default credentials admin for the username and password for the password to log in.
- **Access DVWA Dashboard:** Upon successful login, you will be directed to the DVWA dashboard, where you can navigate to different sections to test various vulnerabilities.

Final Setup and Testing

- Set security to ‘low’



Run a Nikto Scan

- **Initiate the Scan:** Use the command **nikto -h http://localhost/DVWA/** to start scanning the DVWA web application hosted locally on your Apache server.
- **Target Specification:** The **-h** flag specifies the target host or URL. In this case, '**http://localhost/DVWA/**' points Nikto to scan the DVWA instance running on your local machine.
- **Vulnerability Assessment:** Nikto will enumerate potential vulnerabilities, including misconfigurations, outdated software versions, and common security issues.
- **Output Review:** The scan results will be displayed directly in the terminal, showing identified vulnerabilities and possible security weaknesses in the web application.

Nikto Scan Results

- Initiate the Scan
- Target Specification
- Vulnerability Assessment

Nikto Scan Results

- Outdated Software
- Injection Points
- Our focus

Nikto Scan Results & Review

Results & Review

XSS (Cross-Site Scripting) Potential:

- Nikto flagged multiple input fields where user input was not properly sanitized, indicating a potential vulnerability to XSS attacks.

SQL Injection Points:

- Nikto detected areas where SQL queries were directly interacting with user input, suggesting that these inputs were vulnerable to SQL injection attacks.

Command Injection Susceptibility:

- The scan highlighted forms that potentially allowed direct command execution based on user input, making them susceptible to command injection.

Output Review:

- The Nikto scan uncovered critical vulnerabilities in DVWA, including XSS, SQL Injection points, and outdated software, highlighting significant security risks.

Live Demo

Exploiting Command Injection in DVWA

Let’s go!

Exploiting Command Injection in DVWA

A Practical Demonstration of How Attackers Can Compromise Web Servers

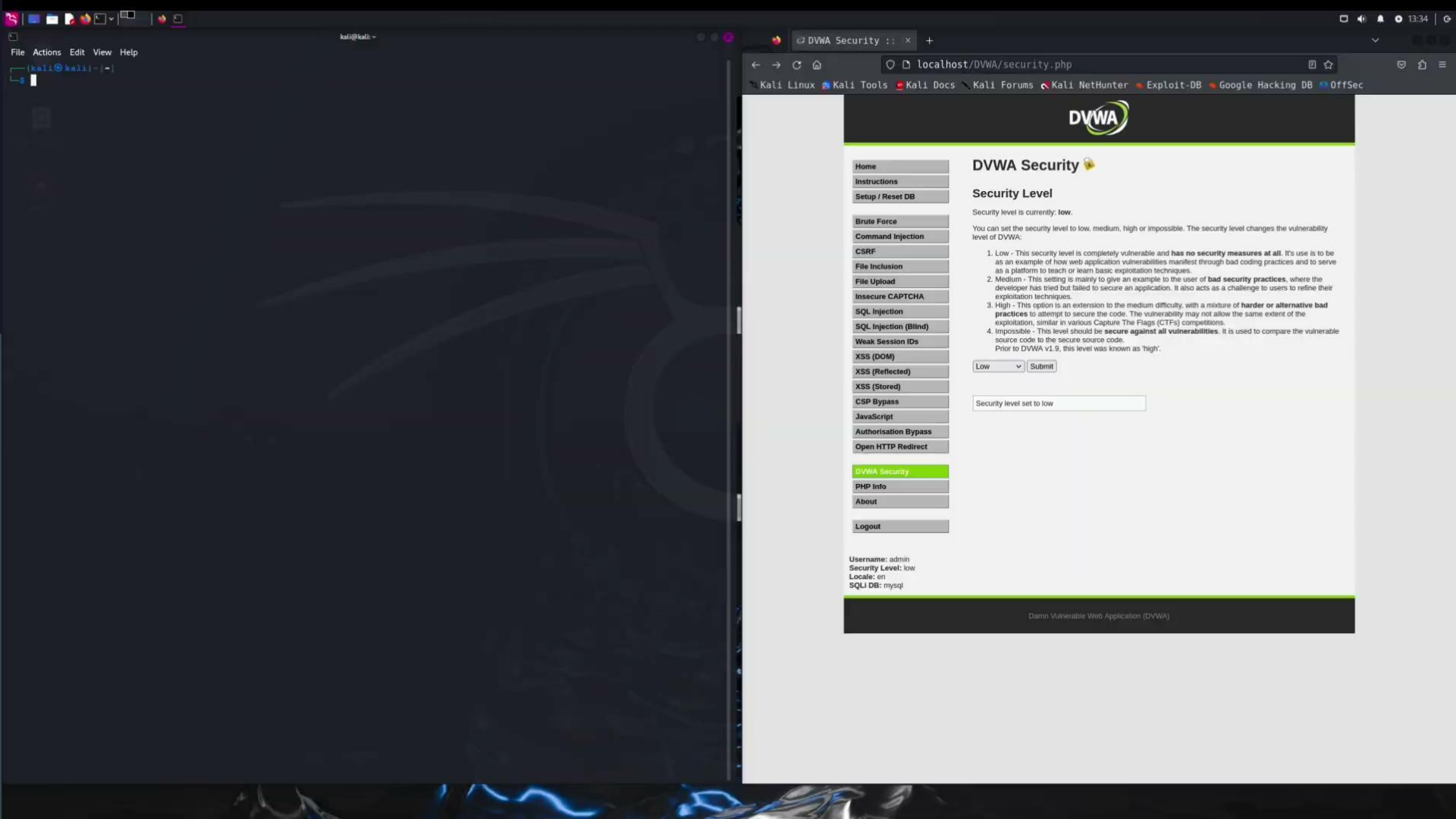
Demonstration Focus:

- Exploiting a Command Injection vulnerability.
- Executing arbitrary commands on a server.

Key Takeaway:

- Emphasizes the critical need for securing web applications.

In this demonstration, we'll show you how simple commands can lead to big consequences, driving home the importance of securing your web applications against these kinds of threats.


kali@kali: ~
File Actions Edit View Help
DVWA Security ::
localhost/DVWA/security.php
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
DVWA Security
PHP Info
About
Logout
DVWA Security
Security Level
Security level is currently: low.
You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:
1. Low - This security level is completely vulnerable and has no security measures at all. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of bad security practices, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of harder or alternative bad practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.
Low
Submit
Security level set to low
Username: admin
Security Level: low
Locale: en
SQLi DB: mysql
Damn Vulnerable Web Application (DVWA)

My Third Role

Analysis & Prevention Specialist for Nikto Attacks

The Role

An “Analysis & Prevention Specialist for Nikto Attacks” is a proactive defender focused on securing web servers against vulnerabilities identified by Nikto scans. They analyze scan results to uncover misconfigurations, outdated software, and common exploits, then implement solutions like server updates, secure configurations, and WAFs. By blending detection with prevention, they ensure potential risks are addressed before they become breaches, acting as key architects in building resilient web defenses. Their expertise helps create a proactive security culture, minimizing exposure to evolving threats.

Nikto Project: Analysis & Prevention of Nikto Attacks

- XSS
- SQL Injection
- Command Line Injection

Cross Site Scripting (XSS) Remediation

XSS Remediation

Content Security Policy (CSP)

- Implement CSP: A Content Security Policy helps mitigate XSS by specifying trusted sources for content. Configure CSP headers to control where scripts, styles, and other resources can be loaded from. Example header configuration:

  Content-Security-Policy: default-src 'self'; script-src 'self' https://trusted-source.com; object-src 'none';

Input Sanitization

- Sanitizing an Email: Clean and validate email inputs to ensure they are correctly formatted and safe.
- Sanitizing a URL: Remove or encode unsafe characters to ensure the URL is properly formatted and secure.
- Sanitizing a String: Strip out HTML tags and encode special characters to prevent code injection and XSS attacks.

Limit Permissions

- Run with Least Privilege: Ensure the web server operates with minimal privileges necessary to reduce the impact of potential command injections or other attacks.

SQL Injection (SQLi) Remediation

SQLi Remediation

Input Validation and Sanitization

- Strict Validation: Ensure all user inputs match expected data types and formats (e.g., numeric for IDs).
- Sanitization: Remove or properly encode special characters that could be used for SQL injection.

Implement Parameterized Queries

- Prepared Statements: Use prepared statements to separate SQL code from user inputs, preventing malicious code execution.
- ORMs (Object-Relational Mappers): Leverage ORMs to securely manage database interactions and abstract away raw SQL queries.

Least Privilege Principle

- Restrict Database Permissions: Grant the database user only the minimum permissions required for application functionality. Avoid using accounts with administrative privileges.

Command Injection (CI) Remediation

CI Remediation

Update and Patch

- Regularly update all software (OS, web server, apps) to ensure security patches are applied.
- Perform security audits to detect outdated software, misconfigurations, and vulnerabilities.

Input Validation and Sanitization**

- Enforce strict validation rules for user inputs (e.g., allow only expected formats).
- Sanitize inputs to remove or escape potentially dangerous characters.

Use Safe APIs and Avoid Shell Commands

- Avoid direct shell commands; use built-in functions or secure libraries instead.
- Parameterize inputs if using shell commands, separating user inputs from command logic.

Command Injection Remediation (cont.)

CI Remediation (cont.)

Limit Permissions

- Run servers and applications with the least privilege necessary (avoid root access).
- Isolate sensitive operations to restricted user accounts.

Application-Level Controls

- Disable unnecessary features such as(e.g., file uploads, command execution).

Implement logging and monitoring for suspicious command activity.

- This concise list fits key remediation strategies into one slide while covering all critical aspects.

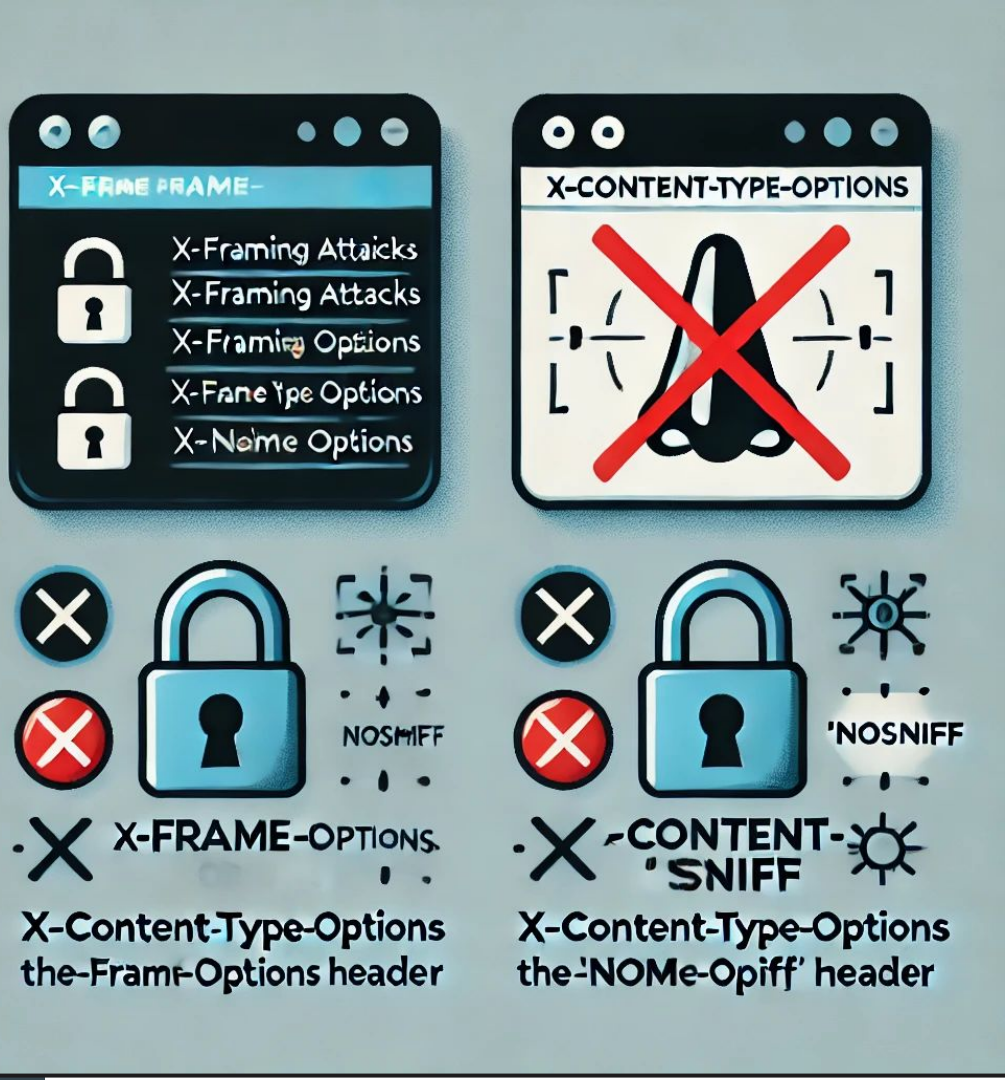
Securing Headers to Mitigate Attacks

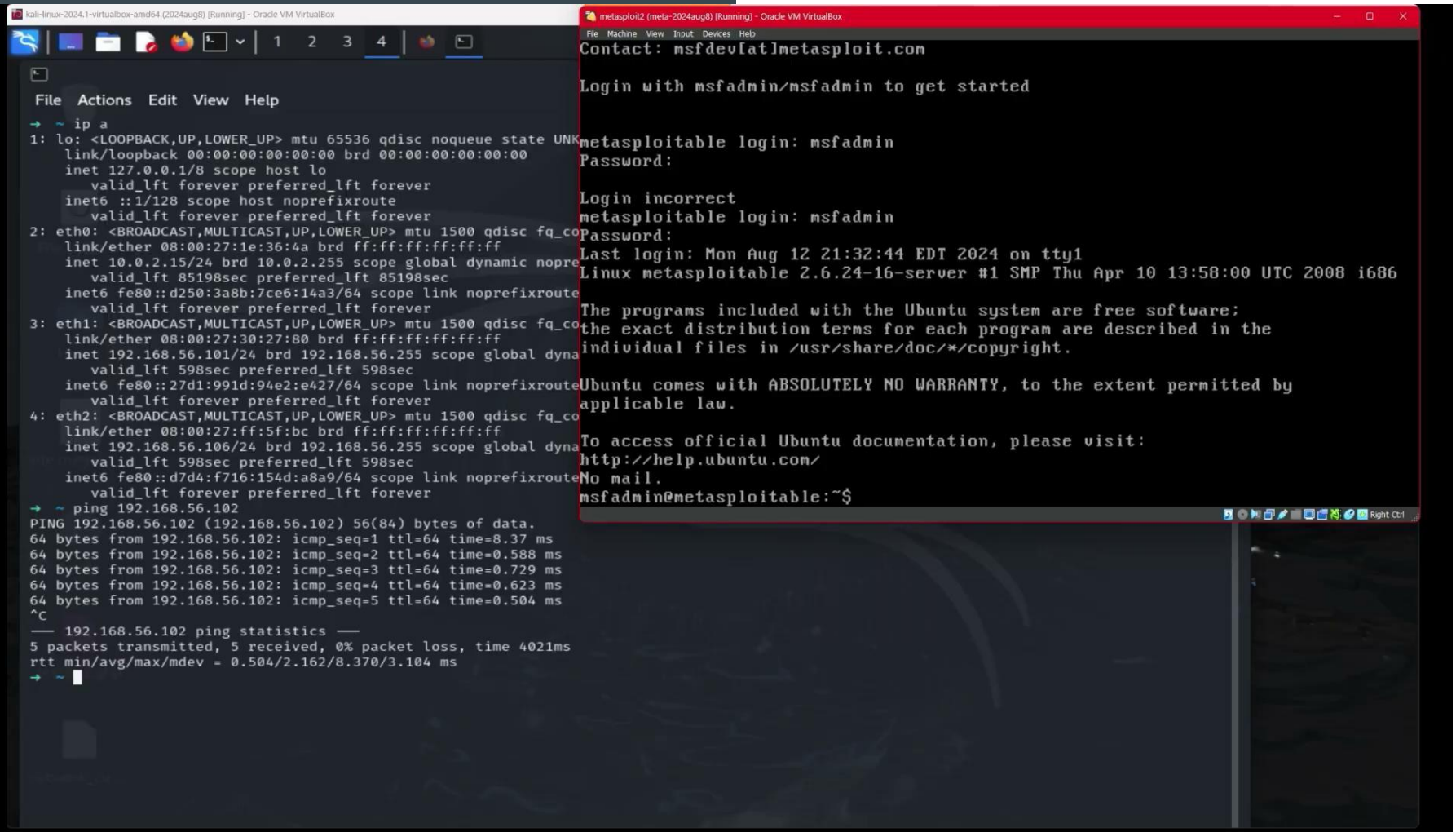
X-Frame-Options Header

- Prevents malicious framing attacks (like clickjacking).
- Acts as a digital lock on your website's "windows" by controlling who can embed your content.

X-Content-Type-Options Header

- Set to `nosniff` to prevent browsers from executing disguised harmful scripts.
- Ensures browsers don't "take candy from strangers" by validating content types correctly.




kali-linux-2024.1-virtualbox-amd64 (2024aug8) [Running] - Oracle VM VirtualBox
1 2 3 4
File Actions Edit View Help
➜ ~ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNK
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
inet 127.0.0.1/8 scope host lo
valid_lft forever preferred_lft forever
inet6 ::1/128 scope host noprefixroute
valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_co
link/ether 08:00:27:1e:36:4a brd ff:ff:ff:ff:ff:ff
inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic nopre
valid_lft 85198sec preferred_lft 85198sec
inet6 fe80::d250:3a8b:7ce6:14a3/64 scope link noprefixroute
valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_co
link/ether 08:00:27:30:27:80 brd ff:ff:ff:ff:ff:ff
inet 192.168.56.101/24 brd 192.168.56.255 scope global dyna
valid_lft 598sec preferred_lft 598sec
inet6 fe80::27d1:991d:94e2:e427/64 scope link noprefixroute
valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_co
link/ether 08:00:27:ff:5f:bc brd ff:ff:ff:ff:ff:ff
inet 192.168.56.106/24 brd 192.168.56.255 scope global dyna
valid_lft 598sec preferred_lft 598sec
inet6 fe80::d7d4:f716:154d:a8a9/64 scope link noprefixroute
valid_lft forever preferred_lft forever
➜ ~ ping 192.168.56.102
PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data.
64 bytes from 192.168.56.102: icmp_seq=1 ttl=64 time=8.37 ms
64 bytes from 192.168.56.102: icmp_seq=2 ttl=64 time=0.588 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=64 time=0.729 ms
64 bytes from 192.168.56.102: icmp_seq=4 ttl=64 time=0.623 ms
64 bytes from 192.168.56.102: icmp_seq=5 ttl=64 time=0.504 ms
^C
— 192.168.56.102 ping statistics —
5 packets transmitted, 5 received, 0% packet loss, time 4021ms
rtt min/avg/max/mdev = 0.504/2.162/8.370/3.104 ms
➜ ~
metasploit2 (meta-2024aug8) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Contact: msfdev[at]metasploit.com
Login with msfadmin/msfadmin to get started
metasploitable login: msfadmin
Password:
Login incorrect
metasploitable login: msfadmin
Password:
Last login: Mon Aug 12 21:32:44 EDT 2024 on tty1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$
Right Ctrl

The Last Role

Operations Support & Program Manager

The Role

As an "**Operations Support & Program Manager**," I bridge the gap between strategic planning and daily execution. I'm responsible for overseeing critical processes, ensuring smooth operations while managing cross-functional programs that align with our organizational goals. My role involves optimizing workflows, coordinating teams, and tracking key metrics to meet objectives on time and within budget. I also take a proactive approach to problem-solving, identifying risks early and driving continuous improvement initiatives. I see myself as the backbone of efficiency, ensuring projects and operations stay aligned and on track for success.

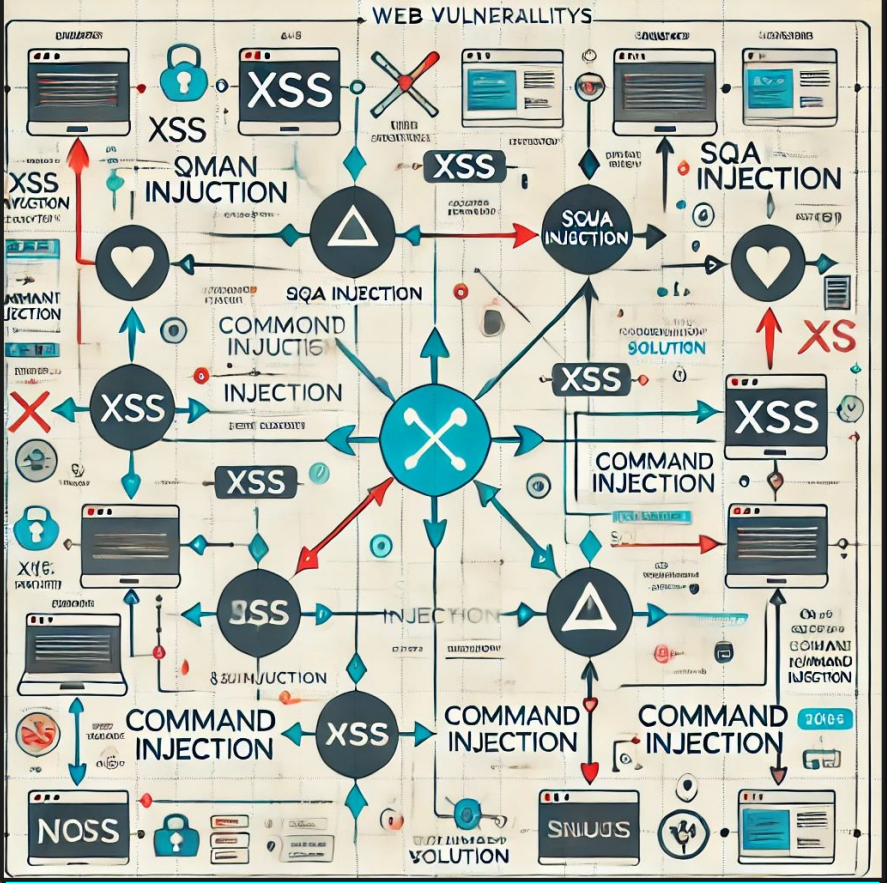
Nikto Project: Project Overview & Breakdown

Nikto Project Overview

- Technical Research: Focus on Malicious Use & Payloads

Project Breakdown

- What We Learned: Key insights from the project Real World Application: How the findings apply in the real world



Key Takeaways

Overview of Discovered Vulnerabilities

- Recap the main vulnerabilities discovered during our Nikto scans, including XSS, SQL Injection, and Command Injection.

Securing Web Applications

- Highlight the importance of implementing security measures to protect against these vulnerabilities.

Effective Remediation Techniques

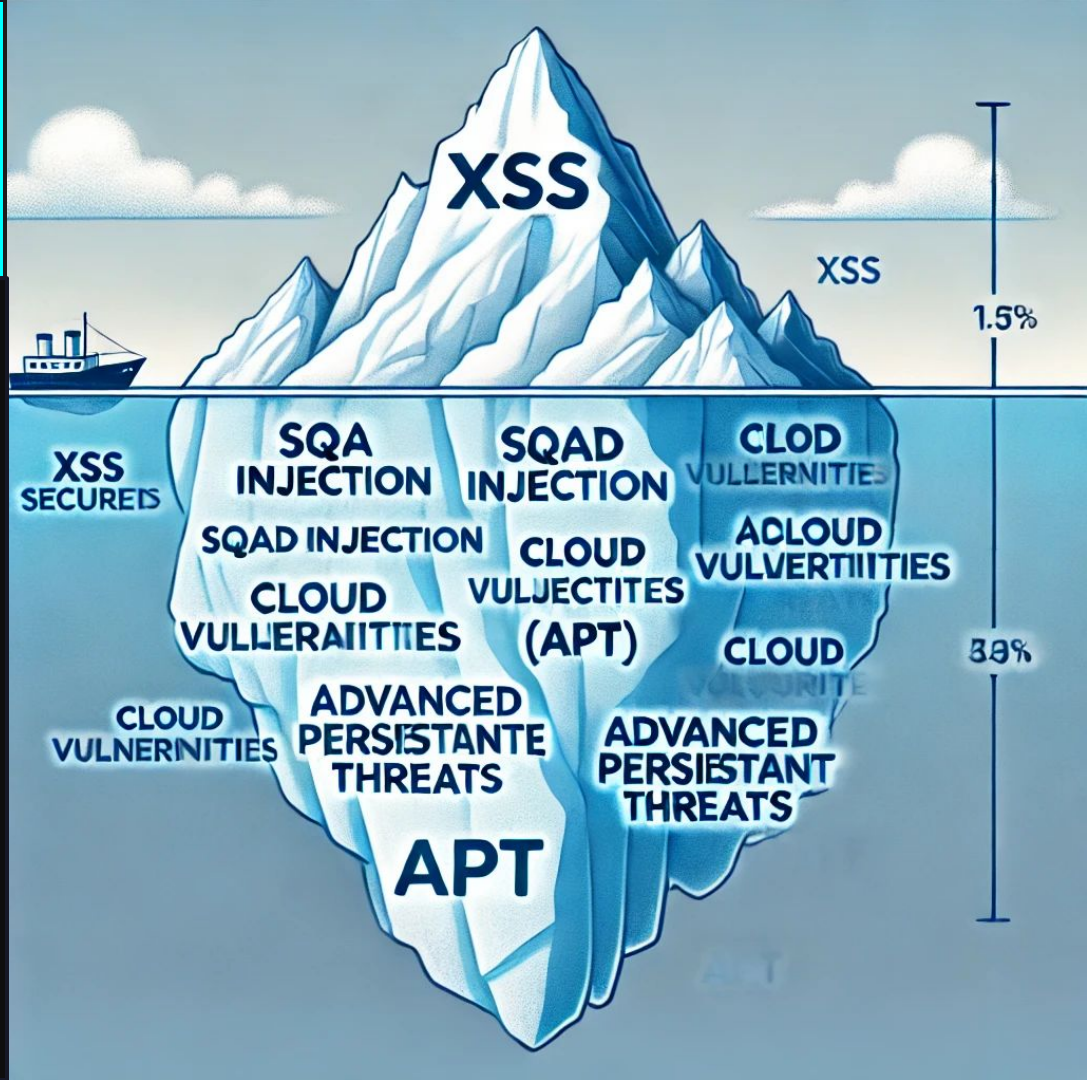
- Discuss the specific remediation techniques used to secure the web applications, emphasizing the practical application of these strategies in a real-world context.

Continuous Vigilance

- Stress the necessity of ongoing security practices and regular updates to safeguard web environments.

Future Areas of to Cover

- API Security
- Cloud Vulnerabilities
- Advanced Persistent Threats (APT)
- Cybersecurity Research



“Your choice is simple: Sink or Swim. With Sink or Swim Security, we anchor your defenses to help you stay ahead in a sea of cyber threats.”

Resilient

Vigilant

Proactive

Secure

Reliable

Fortified

Adaptive

Q & A ?

Appendix & Resources