Nikto Project

Comprehensive Cyber Resilience:

Dive into the depths of cybersecurity with the Nikto CyberShield Initiative. This project explores vital vulnerability scanning using Nikto, targeting common web exploits like XSS 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

Stealth Protectors

Report

"Silently Secure, Effectively Stealthy."

Meet the Team

Maryna Yaroshenko

Historical & Foundational Cybersecurity Researcher

Maryna spearheaded the foundational phase of our project, guiding us through the historical context and development of cybersecurity tools which laid the groundwork for our deeper dive into Nikto and its capabilities.

Chet Flowers

Technical Researcher for Malicious Use & Payloads

Chet took the reins in Nikto, where he applied his expertise in technical vulnerabilities to demonstrate practical attack scenarios using the Damn Vulnerable Web Application, providing a real-time look at how these exploits can be executed.

Joshua Dyke

Researcher for Securing & Mitigation Strategies

Joshua led Phase III, focusing on analyzing and mitigating the vulnerabilities uncovered in previous phases. His detailed approach to securing systems and his strategic insight were crucial in shaping our prevention tactics.

Jamie Ruth

Operations Support & Project Manager

Jamie wrapped up our project with Phase IV, drawing on his operational and management skills to summarize our findings and encourage continued learning and vigilance in the cybersecurity field.

Problem Statement & Goal

Problem Statement:

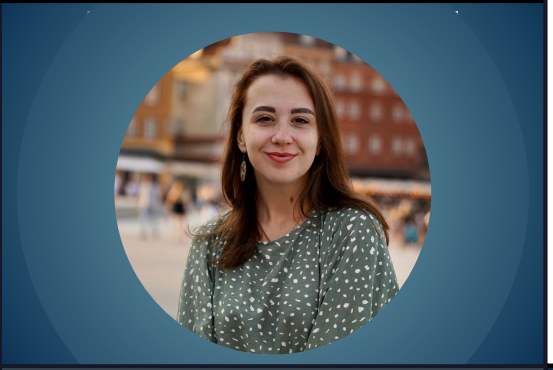
Despite continuous advancements in cybersecurity tools and methodologies, vulnerabilities such as **XSS, SQL Injection, and Command Injection** persistently threaten the integrity and security of web applications. Our Nikto project has uncovered numerous such vulnerabilities, highlighting an ongoing struggle against sophisticated cyber threats in real-world environments.

Goal:

- Enhance **awareness** and understanding of common web vulnerabilities detected by **Nikto**.
- Demonstrate effective mitigation techniques tailored for **vulnerabilities** in *real-world* settings.
- Improve the overall security posture of web applications by showcasing a proactive approach using **Nikto** and other **cybersecurity tools** .

Table of Contents

Maryna Yaroshenko



Historical & Foundational Cybersecurity Researcher

About Me:

I am a cybersecurity professional from Ukraine with a strong background in economic cybernetics, finance, and business management, currently studying at Columbia University and pursuing my CompTIA Security+ certification. My experience includes research in financial risk management and running a small handmade goods business, as well as work in the beauty and service industries. These roles have enhanced my attention to detail, customer service skills, time management, creative problem-solving and adaptability. Committed to continuous learning, I bring a unique perspective and diverse skill set to every challenge I face.

Why I am in Cyber Security:

My interest in cybersecurity developed through my academic studies in economic cybernetics and hands-on experience with real-world security challenges. I'm passionate about protecting digital assets and solving complex security problems. My background in finance, research, and business management has given me a unique perspective on how critical cybersecurity is in our connected world. I'm dedicated to applying my skills and knowledge to build a safer digital space and defend against emerging cyber threats.

Nikto Project: Introduction, Background & Application

1. What is Nikto?
2. The History & Development of Nikto
3. Key Features
4. Usage & Practical Applications
5. Vulnerabilities associated with Nikto
6. Why use Nikto?

What is Nikto?

Nikto is an open-source web server scanner that performs comprehensive tests against web servers for multiple items, including over 6700 potentially dangerous files and scripts, versions on over 1250 servers, and version-specific problems on over 270 servers. It's a valuable tool for identifying potential vulnerabilities and misconfigurations in web applications.

Nikto also checks for server configuration items such as the presence of multiple index files, HTTP server options, and will attempt to identify installed web servers and software.

The History & Development of Nikto

Chris Sullo , a security researcher and developer, created Nikto in 2001.

Purpose

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.

🔑In 2000, he created **CIRT.net** and the **Default Password Database** .

🔑In 2004, Sullo helped create the **Open Source Vulnerability Database (OSVDB)** , a free, community-focused reference tool for all manner of vulnerabilities.

🔑Nikto 1.00 Beta was released on December 27, 2001, with a quick 1.01 bug fix. Although version 2.0, released in November 2007, marked years of improvements, the major version has remained unchanged, with ongoing development continuing on GitHub.

🔑To this day, Sullo has been developing Nikto while working for a variety of companies and pursuing other ventures, including founding the RVAsec cybersecurity conference in 2011.

Key Features of Nikto

1. **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.

2. **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.

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

4. **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.

How Nikto is used! & 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.

Vulnerabilities associated with Nikto

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
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()`).

Vulnerabilities associated with Nikto

Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) is 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 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.

!

Examples of Nikto Detection

- **Reflected XSS** : Injecting payloads like `<script>alert('XSS')</script>` into query parameters.
- **Stored XSS** : Checking for scripts that get stored in databases and executed when the data is rendered on web pages.

Vulnerabilities associated with Nikto

Command Injection

Command Injection is 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).

Why use Nikto?

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

Why use Nikto?

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.

Why use Nikto?

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.

Chet Flowers

Technical Researcher for Malicious Use & Payloads

About Me:

I'm driven by a newfound passion for ethical hacking and the thrill of outsmarting emerging threats. High-pressure situations are my jam—I thrive under stress, and I'm all about trust, loyalty, and constant learning. After winning a pool tournament that landed me in Vegas, I'm gearing up for my next adventure: DEFCON, where I plan to mix my love for cybersecurity with a bit of that Vegas magic.

Why I am in Cyber Security:

I'm in cybersecurity because I'm driven by a deep desire to protect and empower others in our digital age. Inspired by my grandfather's legacy at IBM, my fascination with technology naturally evolved into a passion for securing it. The thrill of outsmarting threats, the satisfaction of safeguarding sensitive data, and the constant learning in this ever-changing field are what keep me committed to this career.

Nikto Project: DVWA & Nikto In Action

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

What is DVWA?

DVWA Setup

- 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'.

What is MariaDB?

Configure and Start Services

- Install and Configure
- Setup
- Initialize and Test

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.

Configure and Start Services

- 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 and Start Services

- 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';
```

Final Setup and Testing

- 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 ‘low’

```
┌──(kali㉿kali)-[/var/www/html/DVWA/config]
└─$ sudo service apache2 start
sudo service mariadb start
```

Setup DVWA
Instructions

About

Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: **/var/www/html/DVWA/config/config.inc.php**

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("**admin // password**") at any stage.

Setup Check

Web Server SERVER_NAME: **localhost**

Operating system: ***nix**

PHP version: **8.2.18**
PHP function display_errors: **Disabled**
PHP function display_startup_errors: **Disabled**
PHP function allow_url_include: **Disabled**
PHP function allow_url_fopen: Enabled
PHP module gd: **Missing - Only an issue if you want to play with captchas**
PHP module mysql: Installed
PHP module pdo_mysql: Installed

Backend database: **MySQL/MariaDB**
Database username: **dvwa**
Database password: ******
Database database: **dvwa**
Database host: **127.0.0.1**
Database port: **3306**

reCAPTCHA key: **Missing**

Writable folder /var/www/html/DVWA/hackable/uploads/: Yes
Writable folder /var/www/html/DVWA/config: Yes

Status in red, *indicate there will be an issue when trying to complete some modules.*

If you see disabled on either *allow_url_fopen* or *allow_url_include*, set the following in your php.ini file and restart Apache.

```
allow_url_fopen = On
allow_url_include = On
```

These are only required for the file inclusion labs so unless you want to play with those, you can ignore them.

Create / Reset Database

XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect

DVWA Security

Impossible ⌄ Submit

Low
Medium
High
Impossible

RUN A NIKTO SCAN

- Initiate the Scan
- Target Specification
- Vulnerability Assessment

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.

RESULTS OF NIKTO SCAN

- Outdated Software
- Injection Points
- Our focus

kali@kali: /var/www/html/DVWA/config

File Actions Edit View Help

```
└─$ nikto -h http://localhost/DVWA/

- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          127.0.0.1
+ Target Hostname:    localhost
+ Target Port:        80
+ Start Time:         2024-08-12 00:42:48 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.62 (Debian)
+ /DVWA/: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US
/docs/Web/HTTP/Headers/X-Frame-Options
+ /DVWA/: The X-Content-Type-Options header is not set. This could allow the user agent to render the content o
f the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/v
ulnerabilities/missing-content-type-header/
+ Root page /DVWA redirects to: login.php
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ OPTIONS: Allowed HTTP Methods: GET, POST, OPTIONS, HEAD .
+ /DVWA/config/: Directory indexing found.
+ /DVWA/config/: Configuration information may be available remotely.
+ /DVWA/tests/: Directory indexing found.
+ /DVWA/tests/: This might be interesting.
+ /DVWA/database/: Directory indexing found.
+ /DVWA/database/: Database directory found.
+ /DVWA/docs/: Directory indexing found.
+ /DVWA/login.php: Admin login page/section found.
+ /DVWA/.git/index: Git Index file may contain directory listing information.
+ /DVWA/.git/HEAD: Git HEAD file found. Full repo details may be present.
+ /DVWA/.git/config: Git config file found. Infos about repo details may be present.
+ /DVWA/.gitignore: .gitignore file found. It is possible to grasp the directory structure.
+ /DVWA/.dockerignore: .dockerignore file found. It may be possible to grasp the directory structure and learn
more about the site.
+ 7850 requests: 0 error(s) and 16 item(s) reported on remote host
+ End Time:           2024-08-12 00:42:52 (GMT-4) (4 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested


      ************************************************************************
```

RESULTS OF NIKTO SCAN

- XSS (Cross-Site Scripting) Potential
- SQL Injection Points
- Command Injection Susceptibility

RESULTS OF NIKTO SCAN

🔑**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

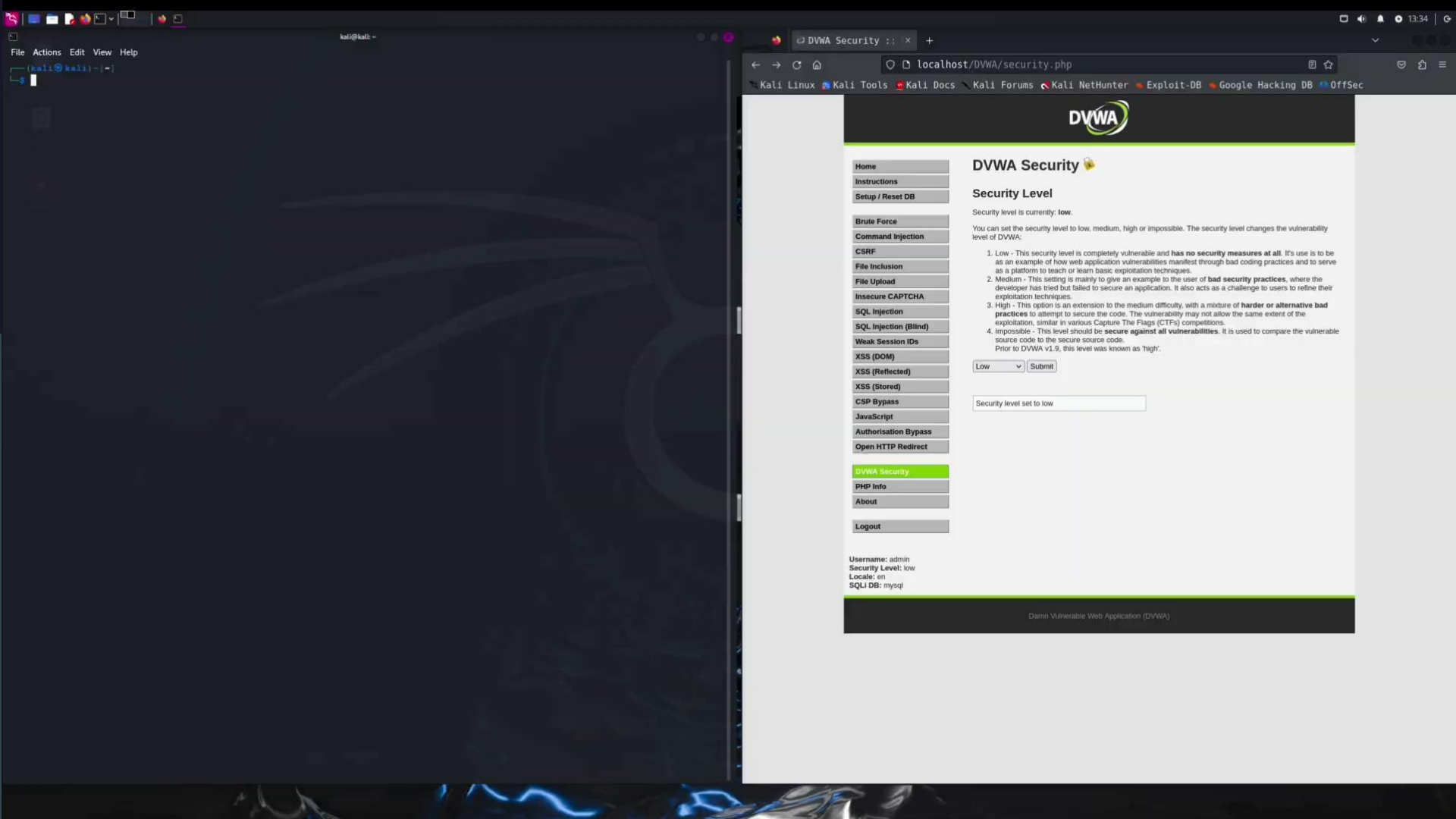
🔑 **Demonstration Focus:**

- 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)

Joshua Dyke

About Me: I'm Joshua Dyke, a dedicated and versatile professional with a dynamic background spanning cybersecurity, sound engineering, and IT support. Currently, I'm deepening my expertise in cybersecurity through a rigorous bootcamp at Columbia University, where I'm honing skills in modern cyber defense strategies, threat intelligence, and network security. My journey in cybersecurity is complemented by hands-on experience with tools like Metasploit, Kali Linux, and Splunk, as well as a solid foundation in network security and database management.

Why I am in Cyber Security:

I am driven by a deep passion for protecting individuals and organizations from cyber threats, combined with a strong affinity for technical challenges. My enthusiasm for cybersecurity stems from my desire to safeguard sensitive information and ensure the integrity of digital systems. I thrive in environments where I can leverage my technical skills to address complex problems and outsmart evolving threats. For me, cybersecurity is not just a career but a commitment to making a tangible impact in a field that constantly pushes the boundaries of technology.

Prevention Researcher for Malicious Use & Payloads

Nikto Project: Analysis & Prevention of Nikto Attacks

1. XSS
2. SQL Injection
3. Command Line Injection

Vectors Nikto Uses to Exploit

XSS Remediation

🔑 Use Content Security Policy (CSP)

- **Implement CSP:** A Content Security Policy helps mitigate XSS by specifying which sources of content are allowed. Configure CSP headers to restrict sources for scripts, styles, and other resources. Example header configuration:

🔑 Input Sanitization

- **Sanitizing an Email** : Cleans up the email address to ensure it's valid.
- **Sanitizing a URL** : Removes unsafe characters to ensure the URL is properly formatted and secure.
- **Sanitizing a String** : Strips out HTML tags and optionally encodes characters to prevent code injection and XSS attacks.

🔑 Limit Permissions

- **Run with Privileges** : Ensure that your web server runs with the minimum privileges necessary to reduce the impact of potential command injection.

Good Security Practices to Prevent Nikto and Tools like it!

SQL Injection

🔑Input Validation and Sanitization

- **Strict Validation:**
  - Validate all user inputs against expected values (e.g., numeric for IDs).
- **Sanitization:**
  - Removal of special characters that can be used for SQL Injection.

🔑Implement Principle of Least Privilege:

Use of Parameterized Queries

- **Prepared Statements:**
  - Use prepared statements to separate SQL code from data inputs, preventing execution of injected code.
- **ORMs (Object-Relational Mappers):**
  - Utilize ORMs to handle database interactions securely and abstract SQL queries.

🔑 Least Privilege Principle

- **Restrict Database Permissions:**
  - Grant the database user only the permissions necessary for application functions. Avoid using administrative privileges.

Are other Operating Systems Different?

Can the be Secured against Nikto

Command Line Injection

🔑**Update and Patch:**

- **Keep Software Updated:** Ensure that both your web server and DVWA are updated to the latest versions to benefit from security patches and improvements.
- **Regular Security Audits:** Perform regular security audits to identify and address potential vulnerabilities.

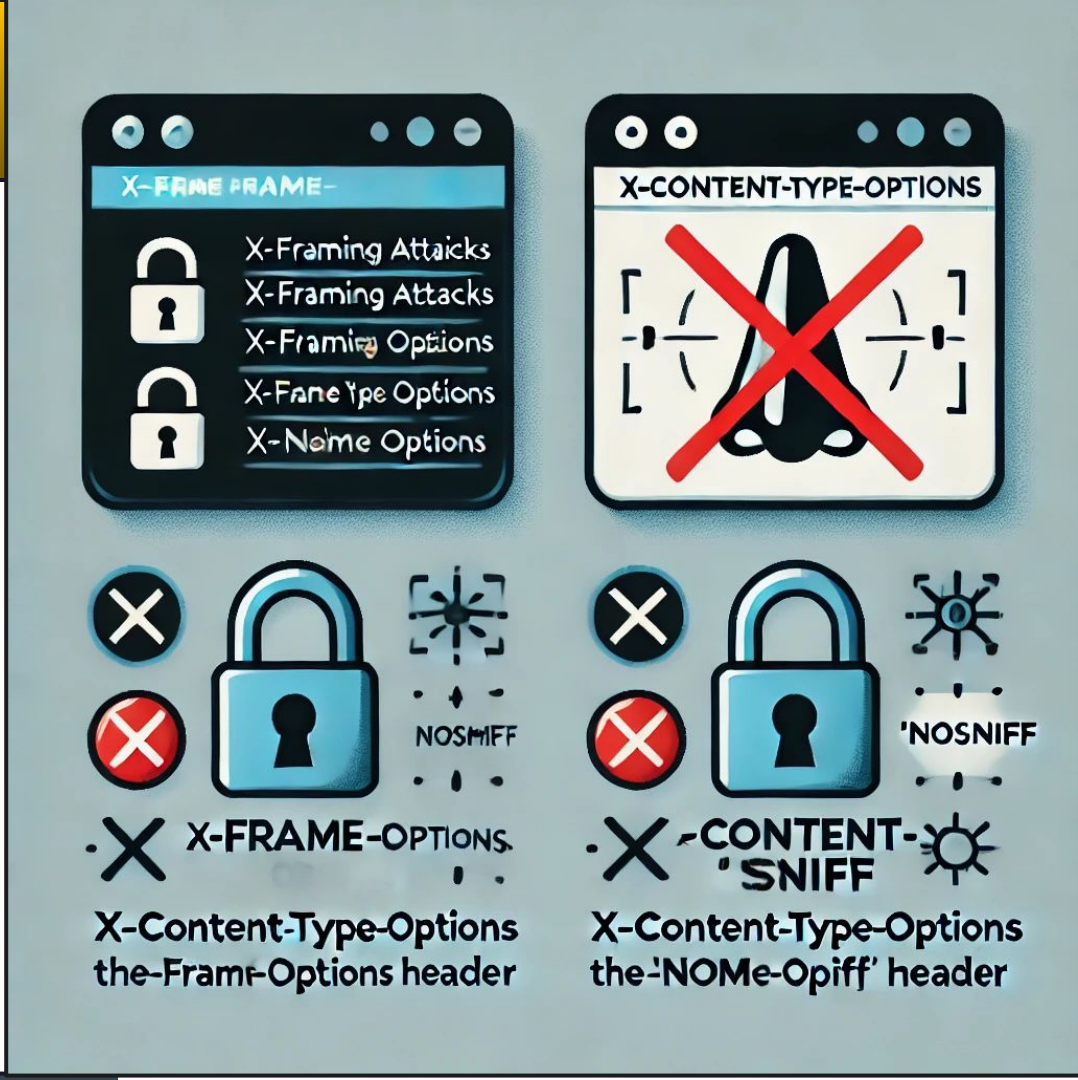
🔑**Limit Permissions**

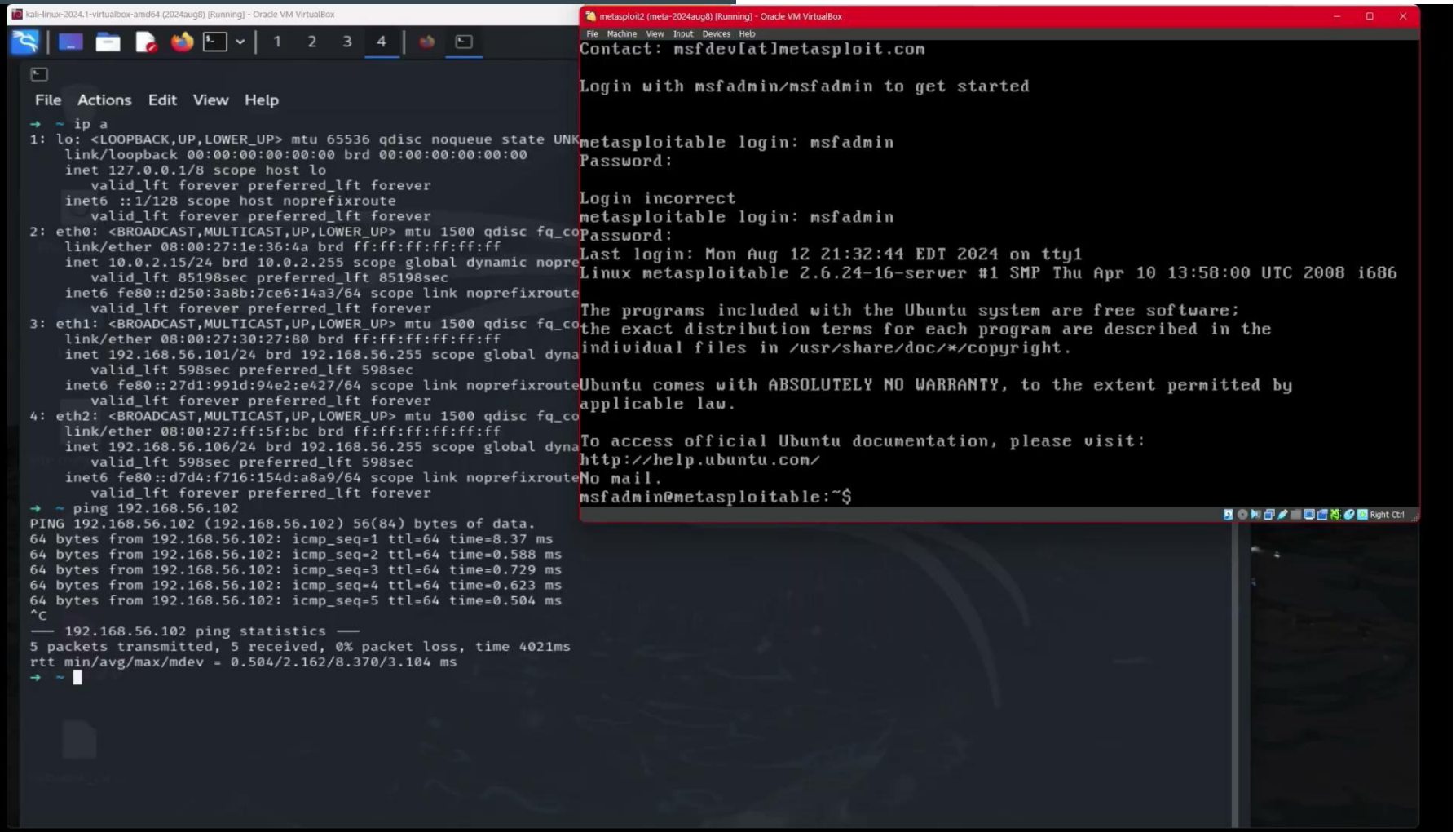
- **Limiting Privileges** : Ensure that your web server runs with the minimum privileges necessary to reduce the impact of potential command injection.

Are other Operating Systems Different? Can the be Secured against Nikto

Securing Headers to Mitigate Attacks

- **X-Frame-Options Header**
  - Prevents malicious framing attacks
  - Acts as a digital lock on your website's windows
- **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




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

Jamie Ruth

Operations Support & Program Manager

About Me:

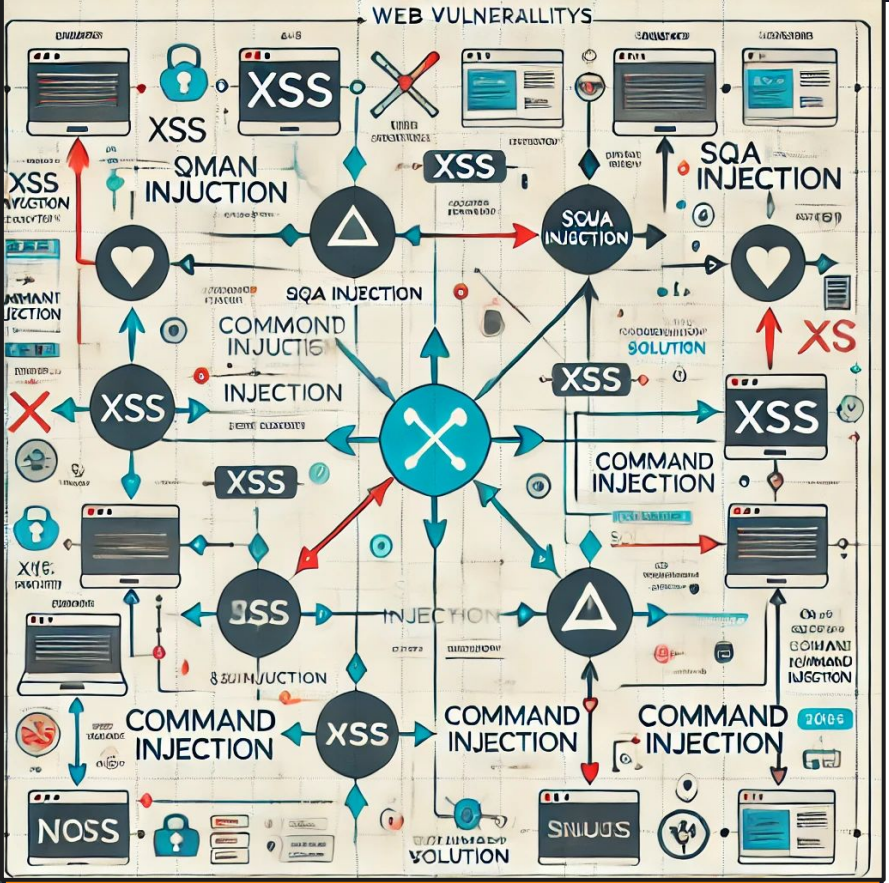
As aspiring cybersecurity professional, my journey from the woodshop to celestial navigation in the Navy, and ultimately earning a math degree, has instilled in me a profound appreciation for problem-solving and strategic thinking. My military background provides a disciplined approach, allowing me to tackle cybersecurity challenges with precision and innovation. I harness my expertise in data visualization with tools like Splunk and leverage generative AI to develop solutions that extend the reach of advanced technologies to individuals and small businesses, making cybersecurity with AI accessible beyond corporate confines. Additionally, my passion for endurance trail running up 14,000 ft peaks mirrors my approach in cybersecurity—persistent, determined, and always pushing the limits.

Why I am in Cyber Security:

I am driven by a desire to make the digital world safer and more accessible. My entry into cybersecurity was inspired by my ability to see patterns and solutions where others see obstacles, cultivated through my mathematical and strategic military experiences. I am passionate about integrating AI to develop innovative cybersecurity solutions that support individuals and small businesses—groups often underserved by the high costs and complexity of traditional security tools. My goal is to democratize cybersecurity, ensuring protection is within everyone's reach, and fostering a secure, inclusive digital future.

Nikto Project: Project Overview & Project Breakdown

1. Nikto Project Overview
    a. Technical Research: Focus on Malicious Use & Payloads
2. Project Breakdown
    a. What We Learned: Key insights from the project
    b. Real World Application: How the findings apply outside the classroom



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.

Mitigation Strategies and Best Practices

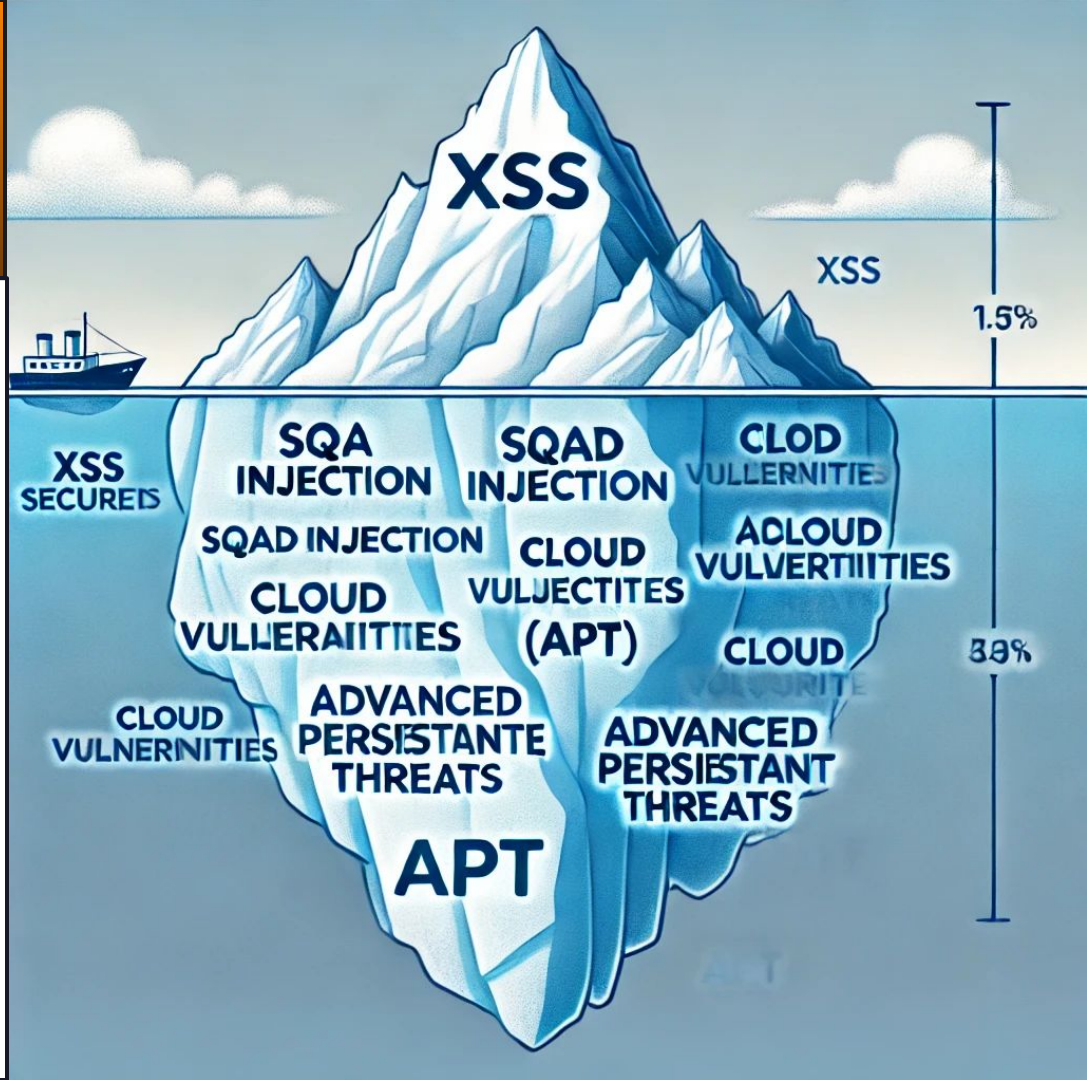
Future Areas of to Cover

API Security

Cloud Vulnerabilities

Advanced Persistent Threats (APT)

Cybersecurity Research



"Embrace your role as a guardian of the digital realm—continuously learn, stay vigilant, and engage with the global cybersecurity community to safeguard our interconnected world."

Continuous Learning

Feedback

Future Topics

Community Forums

Vigilance

Online Courses

Webinars

Questions
&
Answers

Appendix & Resources