# Week 1
# Penetration Testing Report

## Introduction

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the **DVWA**. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

## 1. Objective

The objective of the assessment was to uncover vulnerabilities in the **DVWA** and provide a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

## 2. Scope

This section defines the scope and boundaries of the project.

| | |
|---|---|
| **Application Name** | **DVWA** |

## 3. Summary

Outlined is a Black Box Application Security assessment for the DVWA.

| High | Medium | Low |
|---|---|---|
| 1 | 1 | 1 |

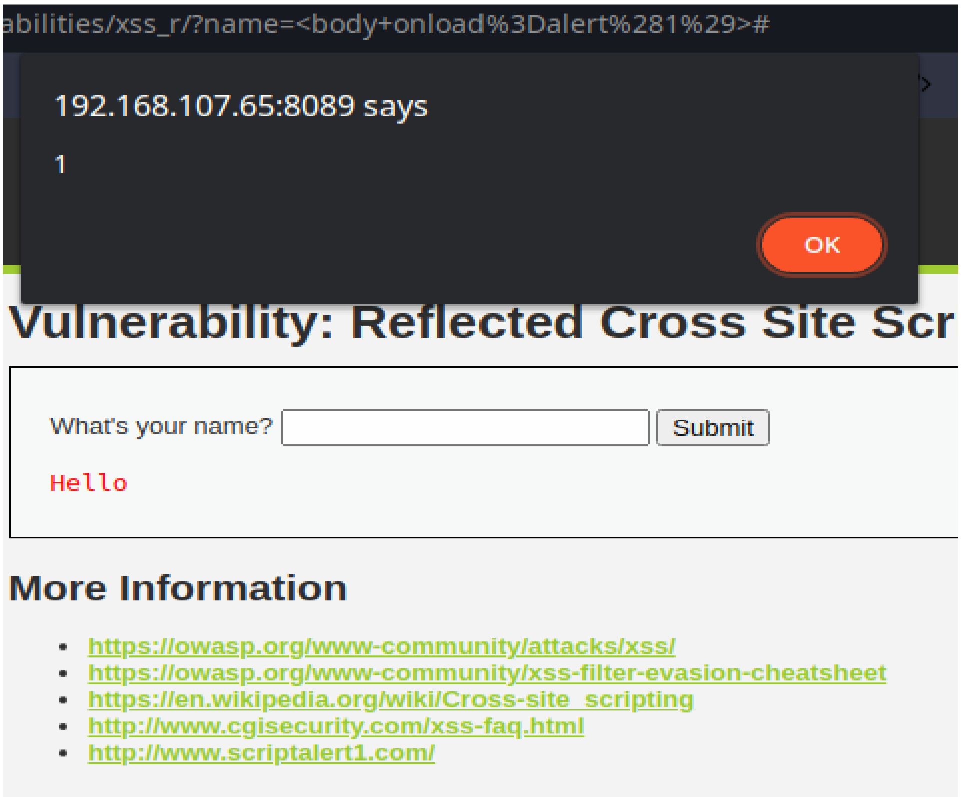
| | | |
|---|---|---|
| High | - | **Number of Sub-labs with hard difficulty level** |
| Medium | - | **Number of Sub-labs with Medium difficulty level** |
| Low | - | **Number of Sub-labs with Easy difficulty level** |

## 1. XSS

| Reference | Risk Rating |
|---|---|
| XSS | **Low** |
| **Tools Used** | |
| Browser | |
| **Vulnerability Description** | |
| The vulnerability is Cross-Site Scripting that allows user to execute JS codes in the input fields | |
| **How It Was Discovered** | |
| Manual Analysis - Pass any JS code in the input field and it will get executed | |
| **Vulnerable URLs** | |
| http://192.168.107.65:8089/vulnerabilities/xss_r/ | |
| **Consequences of not Fixing the Issue** | |
| Impersonating users and stealing credentials. | |
| **Suggested Countermeasures** | |
| Filter input and encode data. | |
| **References** | |
| https://github.com/s0md3v/AwesomeXss/ | |

## Proof of Concept

The proof of the above vulnerability.



abilities/xss_r/?name=<body+onload%3Dalert%281%29>#

192.168.107.65:8089 says

1

Vulnerability: Reflected Cross Site Scr

What's your name?

Hello

More Information

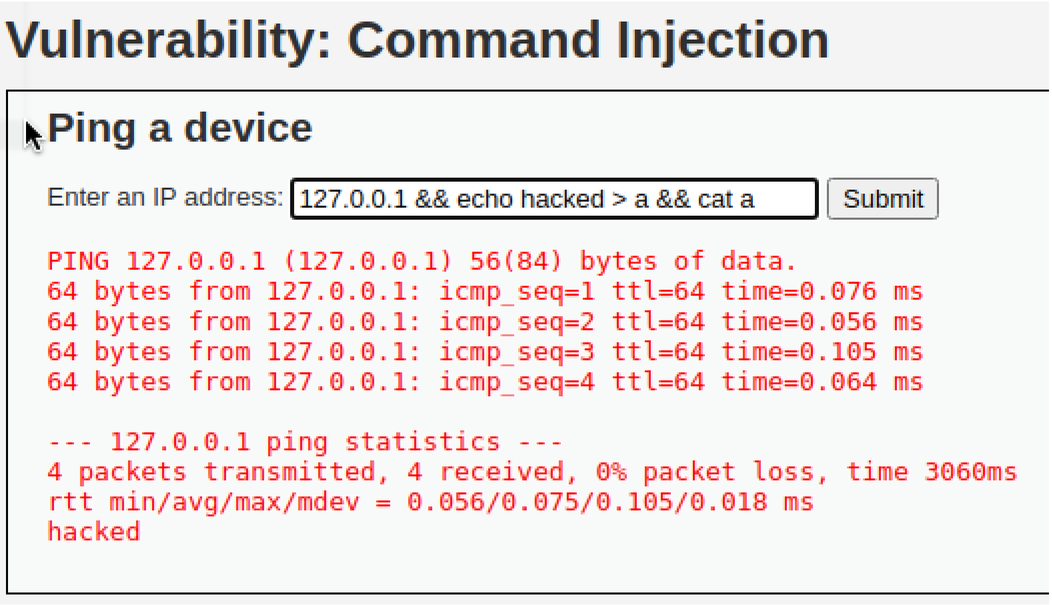
- https://owasp.org/www-community/attacks/xss/
- https://owasp.org/www-community/xss-filter-evasion-cheatsheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

## 2. Command Injection

| Reference | Risk Rating |
|---|---|
| Command Injection | **Medium** |
| **Tools Used** | |
| Browser | |
| **Vulnerability Description** | |
| The vulnerability is command injection that allows an attacker to access server directly passing commands through url or text fields. | |
| **How It Was Discovered** | |
| Manual Analysis - Use & and pass command after ip address. | |
| **Vulnerable URLs** | |
| http://192.168.107.65:8089/vulnerabilities/exec/ | |
| **Consequences of not Fixing the Issue** | |
| Complete web user takeover. | |
| **Suggested Countermeasures** | |
| Input filter | |
| **References** | |
| https://portswigger.net/web-security/os-command-injection | |

## Proof of Concept

The proof of the above vulnerability.



Vulnerability: Command Injection

Ping a device

Enter an IP address: 127.0.0.1 && echo hacked > a && cat a

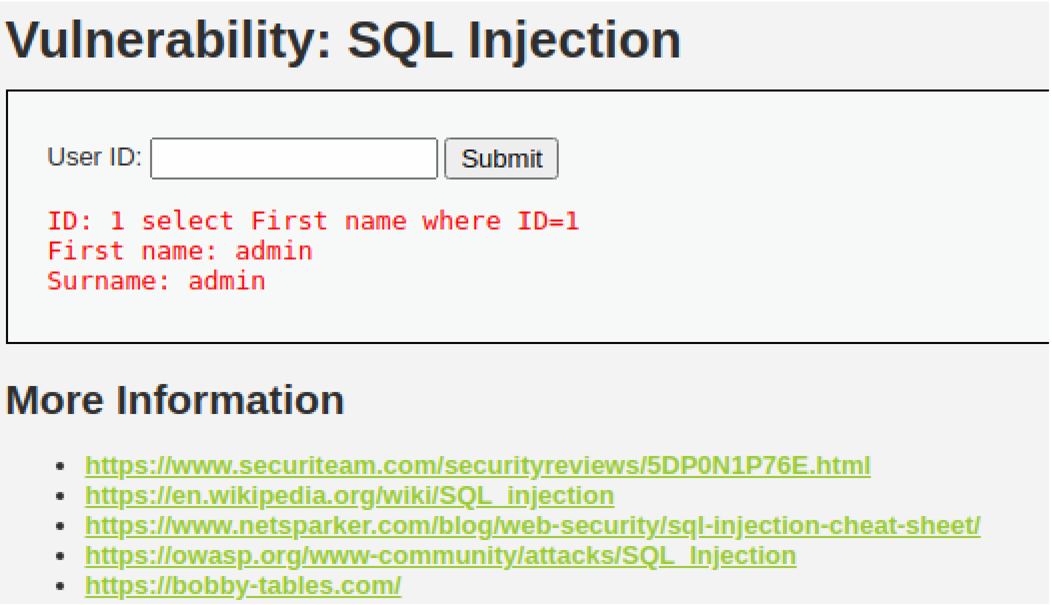
```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.076 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.105 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.064 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.056/0.075/0.105/0.018 ms
hacked
```

## 3. SQL Injection

| Reference | Risk Rating |
|---|---|
| SQL Injection | **High** |
| **Tools Used** | |
| Browser | |
| **Vulnerability Description** | |
| The vulnerability is sql injection which allows an attacker to access a database. | |
| **How It Was Discovered** | |
| Manual Analysis - Pass sql query after an integer id. | |
| **Vulnerable URLs** | |
| http://192.168.107.65:8089/vulnerabilities/sqli/ | |
| **Consequences of not Fixing the Issue** | |
| Stealing creds and sensitive data. | |
| **Suggested Countermeasures** | |
| Implement waf and sanitize input. | |
| **References** | |
| https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/ | |

## Proof of Concept

The proof of the above vulnerability.



Vulnerability: SQL Injection

```
ID: 1 select First name where ID=1
First name: admin
Surname: admin
```

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/