

# DVWA Demo

## Penetration Testing Report

### Notes:-

- You can refer to this guide or some other to install DVWA:- [How to install DVWA](#)
- You will need to do some research by yourself on DVWA.
- If you are facing any issues during installation or this report, you can ask your fellow interns in discord for help. Also this is a sample report based on DVWA, you will need to create a similar report for upcoming weekly labs that we will provide you.

## 1. Introduction

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

## 2. Objective

The objective of the assessment was to assess the state of security and uncover vulnerabilities in **Damn Vulnerable Web Application (DVWA)** and provide with a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

## 3. Scope

This section defines the scope and boundaries of the project.

Application Name	Damn Vulnerable Web Application (DVWA)
URL	<a href="http://127.0.0.1/DVWA/">http://127.0.0.1/DVWA/</a> *

### 3.1. Assessment Attribute(s)

Parameter	Value
Starting Vector	External
Target Criticality	Critical
Assessment Nature	Cautious & Calculated
Assessment Conspicuity	Clear
Proof of Concept(s)	Attached wherever possible and applicable.

### 3.2. Risk Calculation and Classification

Following is the risk classification:

Info	Low	Medium	High	Critical
No direct threat to host/ individual user accounts. Sensitive information can be revealed to the adversary.	Vulnerabilities may not have public exploit (code) available or cannot be exploited in the wild. Vulnerability observed may not have a high rate of occurrence. Patch workaround released by vendor.	Vulnerabilities may not have public exploit (code) available or cannot be exploited in the wild. Patch/ workaround not yet released by vendor.	Vulnerabilities which can be exploited publicly, workaround or fix/ patch available by vendor.	Vulnerabilities which can be exploited publicly, workaround or fix/ patch may not be available by vendor.

Table 1: Risk Rating

## Summary

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

<http://127.0.0.1/DVWA/vulnerabilities/exec/>

<http://127.0.0.1/DVWA/vulnerabilities/sqli/>

Following section illustrates **Detailed** Technical information about identified vulnerabilities.

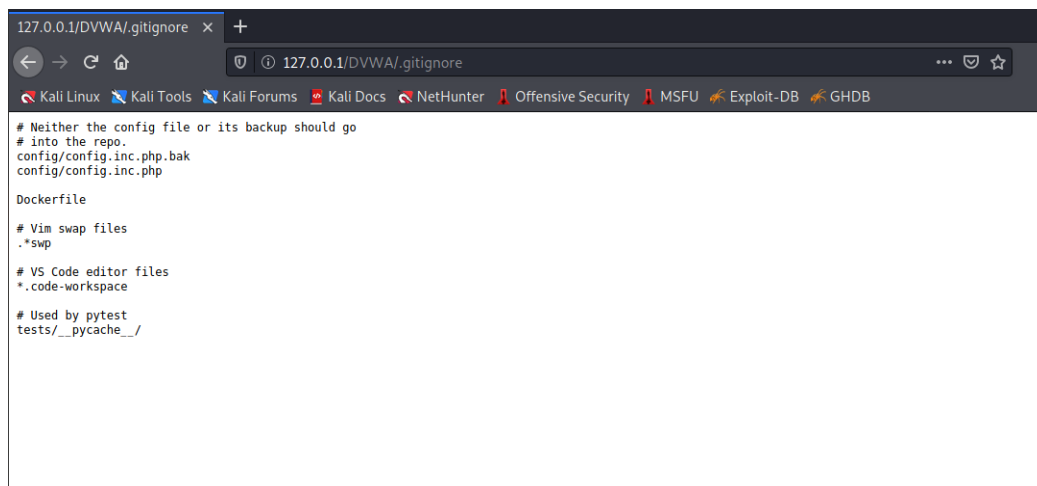
**Total: 3 Vulnerabilities**

High	Medium	Low
1	1	2

## 1.1 Interesting files found

Reference No:	Risk Rating:
files_01	Low <div><div></div></div>
Tools Used:	
Browser	
Vulnerability Description:	
These files/folders usually contain sensitive information which may help attackers to mount further attacks against the server. Manual validation is required	
Vulnerability Identified by / How It Was Discovered	
Manual Analysis	
Vulnerable URLs / IP Address	
<a href="http://127.0.0.1/DVWA/.gitignore">http://127.0.0.1/DVWA/.gitignore</a>	
Implications / Consequences of not Fixing the Issue	
Attackers will often attempt to exploit unpatched flaws or access default accounts, unused pages, unprotected files and directories, etc to gain unauthorized access or knowledge of the system.	
Suggested Countermeasures	
<ul style="list-style-type: none"><li>• We recommend you to analyze if the mentioned files/folders contain any sensitive information.</li><li>• Restrict their access according to the business purposes of the application.</li></ul>	
References	
<a href="#">A6:2017-Security Misconfiguration</a>   <a href="#">OWASP</a>	

### Proof of concept:



```
# Neither the config file or its backup should go
# into the repo.
config/config.inc.php.bak
config/config.inc.php

Dockerfile


# Vim swap files
.*swp

# VS Code editor files
*.code-workspace

# Used by pytest
tests/__pycache__/
```

Fig 1:gitignore file found. It is possible to grasp the directory structure

## 1.2 Interesting files found

Reference No:	Risk Rating:
Config_02	Low 
Tools Used:	
Browser	
Vulnerability Description:	
Security misconfiguration can happen at any level of an application stack, including the network services, platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage. Automated scanners are useful for detecting misconfigurations, use of default accounts or configurations, unnecessary services, legacy options, etc.	
Vulnerability Identified by / How It Was Discovered	
Manual Analysis	
Vulnerable URLs / IP Address	
<a href="http://127.0.0.1/DVWA/config/">http://127.0.0.1/DVWA/config/</a>	
Implications / Consequences of not Fixing the Issue	
Such flaws frequently give attackers unauthorized access to some system data or functionality. Occasionally, such flaws result in a complete system compromise. The business impact depends on the protection needs of the application and data	
Suggested Countermeasures	
It is recommended to secure installation processes should be implemented, including : <ul style="list-style-type: none"><li>• A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. Development, QA, and production environments should all be configured identically, with different credentials used in each environment. This process should be automated to minimize the effort required to set up a new secure environment.</li><li>• A minimal platform without any unnecessary features, components, documentation, and samples. Remove or do not install unused features and frameworks.</li></ul>	
References	
<a href="#">A6:2017-Security Misconfiguration   OWASP</a>	

**Proof of concept:**

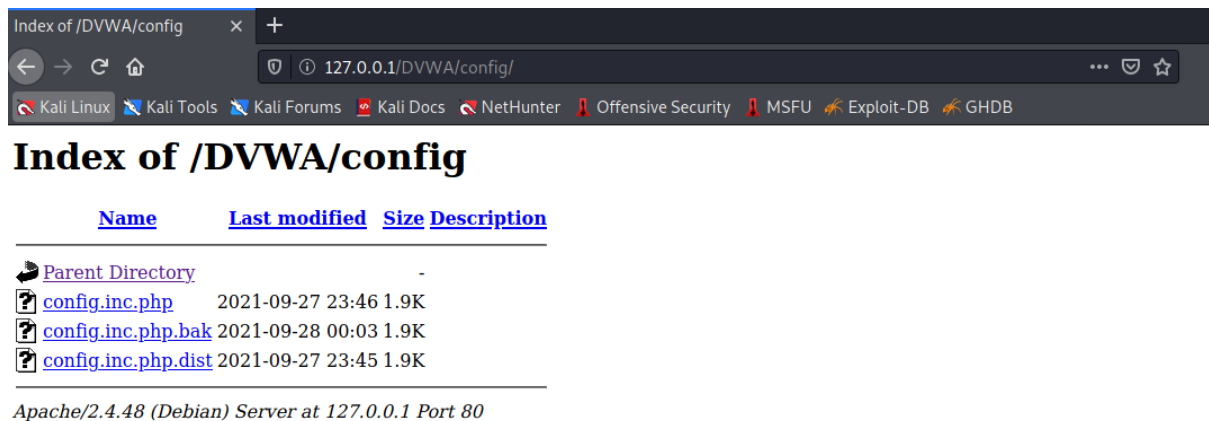


Fig 1: Directory indexing found.

## 2. Insufficient Password Policy .

Reference No:	Risk Rating:
Insufficient_Password_Policy_01	Medium <div></div>
Tools Used:	
Browser	
Vulnerability Description:	
<ul style="list-style-type: none"> <li>Users of the application can set a weak password. These passwords are prone to brute-force attacks.</li> <li>The confidentiality cannot be ensured.</li> </ul>	
Vulnerability Identified by / How It Was Discovered	
Manually	
Vulnerable URLs / IP Address	
<a href="http://127.0.0.1/DVWA/vulnerabilities/brute/">http://127.0.0.1/DVWA/vulnerabilities/brute/</a>	
Implications / Consequences of not Fixing the Issue	
An attacker could easily guess user passwords and gain access to user accounts.	
Suggested Countermeasures	
<ul style="list-style-type: none"> <li>Enforcement of a minimum and maximum length</li> <li>Restrictions against password reuse</li> <li>Restrictions against using common passwords</li> <li>Restrictions against using contextual string in the password (e.g., user id, app name)</li> </ul>	
References	
<a href="#">CWE - CWE-521: Weak Password Requirements (4.5)</a>	

Proof of concept:

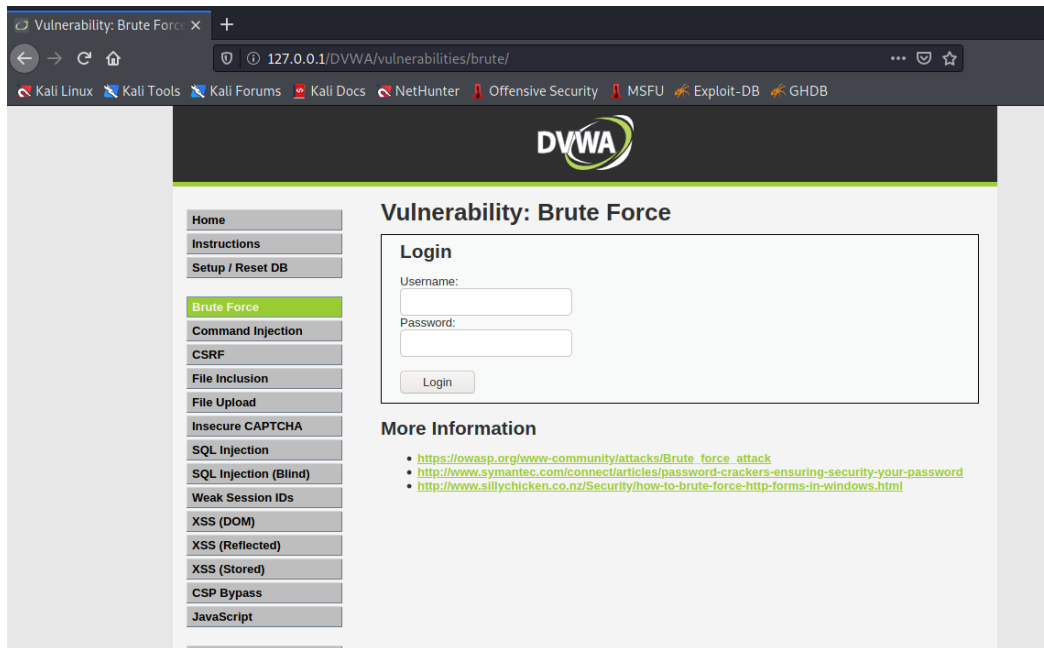


Fig 1: Weak Password Requirements

### 3.SQL Injection

Reference No:	Risk Rating:
SQL_Injection_01	High <div style="width: 100px; height: 10px; background-color: red; display: inline-block;"></div>
Tools Used:	
Browser	
Vulnerability Description:	
The web application is prone to SQL injections. In consequence, all data of the database can be potentially accessed or manipulated by an attacker.	
Vulnerability Identified by / How It Was Discovered	
Manual Analysis	
Vulnerable URLs / IP Address	
<a href="http://127.0.0.1/DVWA/vulnerabilities/sqli/">http://127.0.0.1/DVWA/vulnerabilities/sqli/</a>	
Implications / Consequences of not Fixing the Issue	
The risk exists that an attacker gains unauthorized access to the information from the database of the application. He could extract information such as: application usernames, passwords, client information and other application specific data.	
Suggested Countermeasures	
<ul style="list-style-type: none"> <li>● We recommend implementing a validation mechanism for all the data received from the users.</li> <li>● The best way to protect against SQL Injection is to use prepared statements for every SQL query performed on the database.</li> <li>● Otherwise, the user input can also be sanitized using dedicated methods such as: <code>mysql_real_escape_string</code>.</li> </ul>	
References	
<ul style="list-style-type: none"> <li>● <a href="#">SQL Injection   OWASP</a></li> <li>● <a href="#">CheatSheetSeries/SQL Injection Prevention Cheat Sheet.md at master · OWASP/CheatSheetSeries</a></li> </ul>	

## Proof of concept

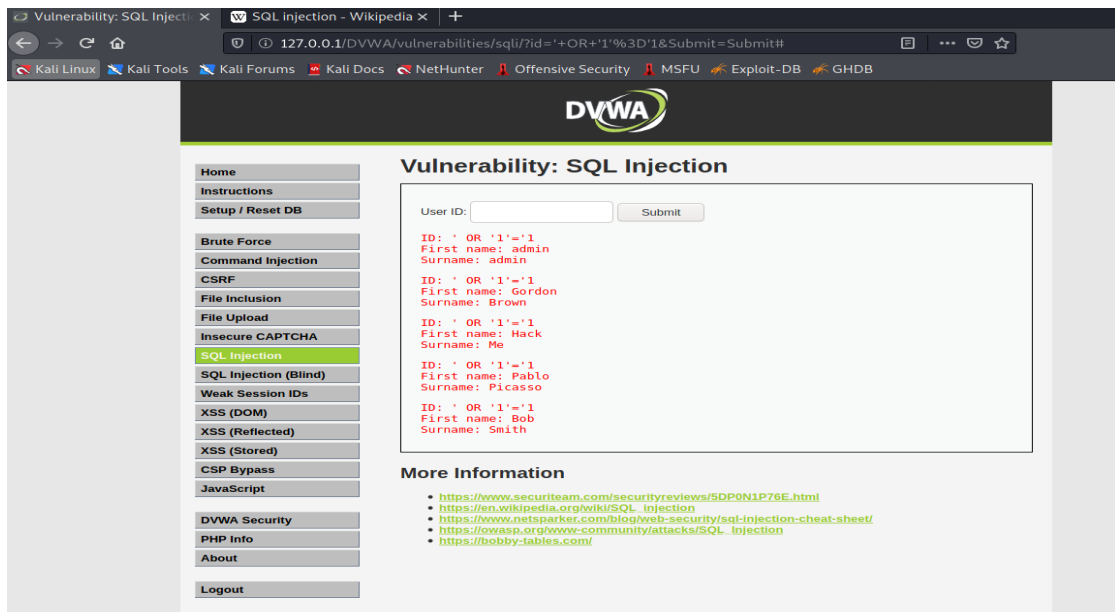


Fig 1: SQL Injection

-----EOF-----