**A**

***PROJECT REPORT***

*On*

**DAMN Vulnerability Web Application Project**

*Submitted in partial fulfilment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**

****

Session: - Jan-June 2023

Submitted by

Utkarsh Mathur(19ETCCS075)

Luthisht Joshi(19ETCCS035)

8th Semester & Computer Science Engineering

Under Guidance of

Nitish Agrawal

Dharmendra Parmar

Cyber Security Manager

Skill Development Institute

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY – 2023**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY – 2023**

**A**

***PROJECT REPORT***

*On*

DAMN Vulnerability Web Application

*Submitted in partial fulfilment of the requirements for the degree of*

**BACHELOR OF TECHNOLOGY**



**Session: - Jan-June 2023**

Submitted by

Utkarsh Mathur (19ETCCS075)

Luthisht Joshi (19ETCCS035)

8TH Semester & Computer Science Engineering

Under Guidance of

Nitish Agrawal

Dharmendra Parmar

Cyber Security Manager

Skill Development Institute

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY - 2023**



Department of Computer Science and Engineering

Techno India NJR Institute of Technology, Udaipur-313001

**Certificate**

This is to certify that project work titled **DAMN VULNERABILITY Web Application** by **UTKARSH MATHUR** was successfully carried out in the Department of Computer Science and Engineering, TINJRIT and the report is approved for submission in the partial fulfillment of the requirements for award of degree of Bachelor of Technology in Computer Science and Engineering.

Nitish Agrawal Dr. Rimpy Bishnoi

Cyber Security Manager Head of Department

CS & Skill Development Institute ,Odisha Dept. of CSE TINJRIT, Udaipur

Date...................... Date......................



Department of Computer Science and Engineering

Techno India NJR Institute of Technology, Udaipur-313001

**Certificate**

This is to certify that project work titled **DAMN Vulnerability Web Application** by **Luthisht Joshi** was successfully carried out in the Department of Computer Science and Engineering, TINJRIT and the report is approved for submission in the partial fulfillment of the requirements for award of degree of Bachelor of Technology in Computer Science and Engineering.

Nitish Agrawal Dr. Rimpy Bishnoi

Cyber Security Manager Head of Department

CS & Skill Development Institute ,Odisha Dept. of CSE TINJRIT, Udaipur

Date...................... Date......................



Department of Computer Science and Engineering Techno India NJR Institute of Technology, Udaipur-313001

**Examiner Certificate**

This is to certify that the following student

**Utkarsh Mathur**

of final year B.Tech. (Computer Science and Engineering), was examined for the project work titled

***DAMN Vulnerability Web Application”***

during the academic year 2022 – 2023 at Techno India NJR Institute of Technology, Udaipur

Remarks

Date:

Signature Signature

(Internal Examiner) (External Examiner)

Name :-…………………. Name:-……………………..

Designation:-…………… Designation:-……………….

Department:-……………. Department:-………………..

Organization:-…………… Organization:-………………



Department of Computer Science and Engineering Techno India NJR Institute of Technology, Udaipur-313001

**Examiner Certificate**

This is to certify that the following student

**Luthisht Joshi**

of final year B.Tech(Computer Science and Engineering), was examined for the project work titled

***DAMN Vulnerability Web Application”***

during the academic year 2022 – 2023 at Techno India NJR Institute of Technology, Udaipur

Remarks

Date:

Signature Signature

(Internal Examiner) (External Examiner)

Name :-…………………. Name:-……………………..

Designation:-…………… Designation:-……………….

Department:-……………. Department:-………………..

Organization:-…………… Organization:-………………

**Preface**

The purpose of this preface is to provide an introduction and overview of the project report on the Damn Vulnerable Web Application (DVWA) Web Application Project. This preface aims to set the context for the report and explain the motivations, objectives, and scope of the project. It also highlights the key stakeholders and their roles in the project's development and implementation. The DVWA Web Application Project was undertaken with the goal of creating a deliberately vulnerable web application that could serve as a practical learning tool for understanding common web application vulnerabilities and security best practices. This project aimed to provide a hands-on platform for developers, security professionals, and enthusiasts to explore and gain practical experience in identifying and mitigating web application vulnerabilities. The increasing number of web application vulnerabilities and the critical impact they can have on businesses and individuals necessitate a proactive approach to security. By creating a vulnerable web application, we sought to raise awareness about common security flaws and encourage secure coding practices. Through this project, we aimed to empower individuals to enhance their knowledge and skills in web application security.

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY – 2023**

**ACKNOWLEDGEMENT**

We take this opportunity to record our sincere thanks to all who helped us to successfully complete this work. Firstly, We are grateful to our **supervisor Nitish Agrawal and Dharmendra Parmar** for his invaluable guidance and constant encouragement, support and most importantly for giving us the opportunity to carry out this work.

We would like to express our deepest sense of gratitude and humble regards to our

**Head of Department Diwakar Sharma** for giving invariable encouragement in our endeavors and providing necessary facility for the same. Also a sincere thanks to all faculty members of Skill Development Institute , Bhubneshwar Odisha for their help in the project directly or indirectly.

Finally, We would like to thank my friends for their support and discussions that have proved very valuable for us. We are indebted to our parents for providing constant support, love and encouragement. We thank them for the sacrifices they made so that we could grow up in a learning environment. They have always stood by us in everything we have done, providing constant support, encouragement and love

**UTKARSH MATHUR[19ETCCS075]**

**LUTHISHT JOSHI[19ETCCS035]**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**TECHNO INDIA NJR INSTITUTE OF TECHNOLOGY, UDAIPUR-313001**

**MAY - 2023**

**Contents**

Abstract …………………………………………………………………………………. i

Contents …………………………………………………………………………………. ii-iii

List of Figures …………………………………………………………………………. iv-v

List of Tables ………………………………………………………………………….. vi

Chapter 1: Executive Summary ………………………………………………. 12

* 1. Background ……………………………………………………. 12
  2. Application Health…………………………………………. 12
  3. Immediate Actionable Recommendation………. 12
  4. Scan Details………………………………………………….. 13
  5. Threat Distribution………………………………………. 13

Chapter 2: DVWA Introduction………………………………………….. 14

2.1 What is DVWA …………………………………………… 14

2.2 Why we use DVWA……………………………………… 14

Chapter 3: SQL Injection…………………………………………………… 15

3.1 What is SQL Injection………………………………. 15

3.2 Prove of Concept……………………………………. 15

3.3 Impact…………………………………………………… 17

3.4 Mitigations……………………………………………. 17-18

Chapter 4: Cross Site Scripting …………………………………… 19

4.1 What is Cross Site Scripting ……………….. 19

4.2 Types of Cross Site Scripting………………. 19

4.3 Reflected Cross Site Scripting ………………………… 20

4.4 Observation………………………………………………….. 20

4.5 Steps……………………………………………………………. 20

4.6 Impact…………………………………………………………. 22

4.7 Mitigation…………………………………………………… 23

4.8 Stored Cross Site Scripting…………………………. 24

4.9 Steps………………………………………………………… 24

4.10 Proof of Concept…………………………………….. 25

4.11 Mitigation……………………………………………… 25

Chapter 5: INFORMATION LEAKAGE THROUGH HTTP RESPONSE HEADERS

5.1 Observation…………………………………………………….. 26

5.2 Steps ……………………………………………………………… 27

5.3 Impact…………………………………………………………… 27

**List of Tables**

Table 1-1 Scanning Details ……………………………………………. 13

Table 1-2 Threat Indication Distribution……………………….. 13

Table 3-1 SQL Injection DB Information…………………………. 15

Table 4-1 Cross Site Scripting[Reflected]………………………. 20

Table 5-1 HTTP Response Header…………………………………. 26

**Chapter 1: Executive Summary**

* 1. **BACKGROUND**

On March 23 2023, we have to perform a security assessment of their Damn Vulnerable Web Application (DVWA) in an eﬀort to ensure the security of their customer's personal information, which is processed and stored by the DVWA application.

**APPLICATION HEALTH**



HIGH



Vulnerabilities in DVWA Web Application

**1.2 OBSERVATIONS**

During the course of this engagement we observed several areas of concern that we believe could pose a significant risk to the security of the application and should be addressed in timely manner. Exploiting these vulnerabilities an attacker can retrieve any data from the database which includes sensitive customer data or take over other user's account.

### **1.3 IMMEDIATE ACTIONABLE RECOMMENDATIONS**

* Validate all user inputs based on a whitelisting approach.
* Perform output encoding of all user supplied inputs which are reacted back in HTML response.
* Add CAPTCHA to the login page after three failed login attempts.

**1.4 SCAN DETAILS**

|  |  |
| --- | --- |
| Start Date | 23 March 2023 |
| End Date | 27 March 2023 |
| Scan Time | 5 Days |
| Server Technology | PHP |
| URL | https://hack.me/dvwa-107.html |
| Credentials | User : Admin Role: Administrator |
| Scope | Black-Box |

Table 1-1 Scanning Details

**1.5 Threat Distribution**

|  |  |  |
| --- | --- | --- |
| **Severity Level** | **Color Indicator** | **CVSS Category** |
| High | Red | 7.00-10.00 |
| Medium | Orange | 4.00-6.99 |
| Low | Green | 0.01-3.99 |

Table 1-2 Threat Indication Distribution

**Chapter 2:DVWA[DAMN VULNERABLE WEB APPLICATION]**

* 1. **What is DVWA?**

DVWA is a PHP/MySQL web application, whose main goal is to be an aid for security professionals to test their skills and tools in a legal environment. We have tried to make the deployment of the DVWA as simple as possible and have built a feature add-on that can be easily applied to the edge NEXUS ALB-X load balancer.



Figure-1

* 1. **Why do we use DVWA?**

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 to aid both students & teachers to learn about web application security in a controlled classroom environment.

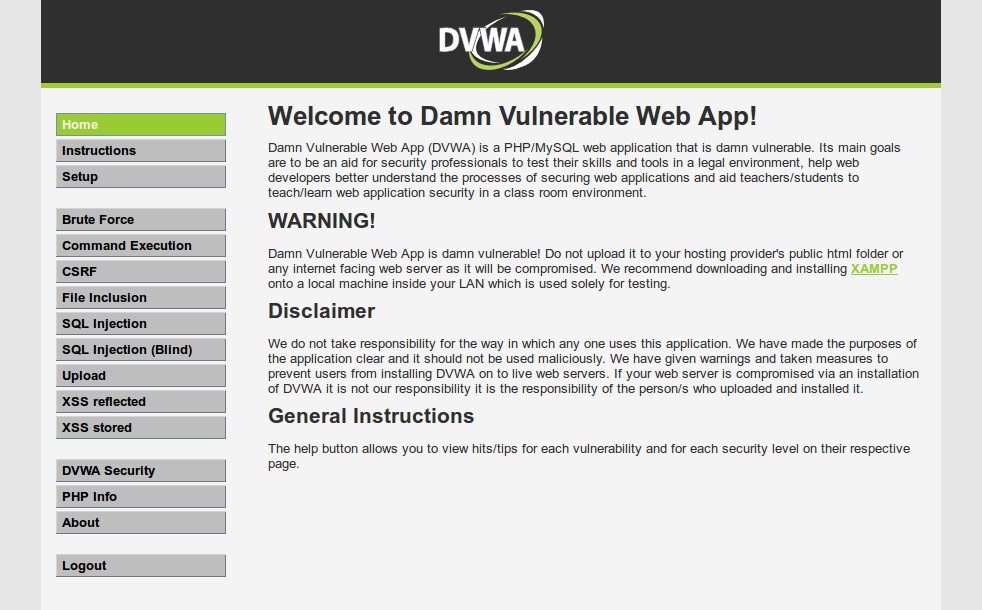


Figure-2

**Chapter 3 :SQL Injection**

**3.1 What is SQL Injection**

SQL Injection is a web application vulnerability that allows an attacker to manipulate a web application's database queries by inserting malicious SQL code into user-supplied input. It occurs when the application fails to properly validate or sanitize user input before using it in an SQL query.

* Following sample data is retrieved from the database using the above mentioned approach

Database Information

|  |  |
| --- | --- |
| Database Name | dvwa |
| Version of Database | 5.1.65-communtiy-log |
| Current User | dvwaUser@localhost |

## **Table 3-1 SQL Injection db Information**

## **3.2 Prove of concept**

* Entered a user id “2” to test the functionality of the application page.
* **Result**: page displayed user “Gordon Brown’s” user ID, first and surname.
* Entered the payload “2’ OR 1=1 — -” to test the presence of an SQL injection vulnerability
* **Result**: the page displayed all user data available on the application as shown below.

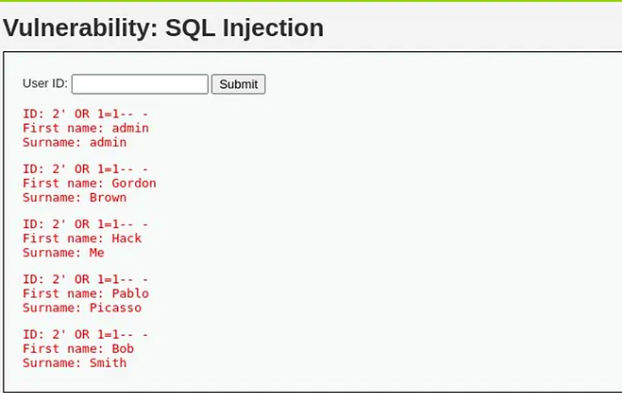


Figure-3

* Through the SQL injection vulnerability, lack of rate limiting of attempts on the server and un-sanitized user-input, I was able to obtain all the usernames and hashed passwords for users held in the “dvwa” database as shown below.

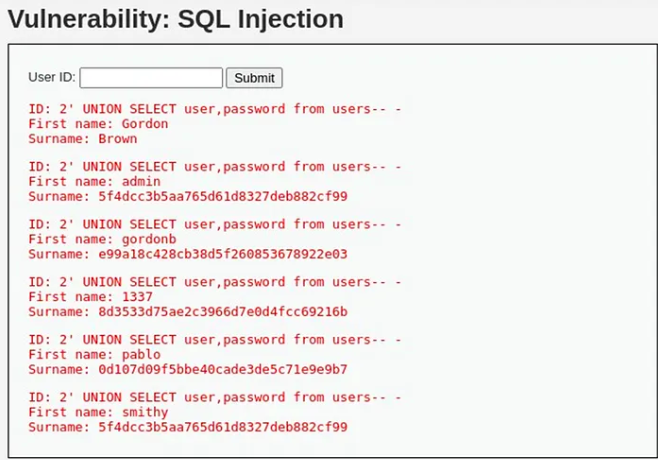


Figure-4

**3.3 Impact**

An attacker can dump entire data from the database that is available to the privilege of current database user. User credentials dumped can further be misused to gain unauthorized access to other user's account. A user only privilege account can be used to conduct this attack in order to gain admin privilege access.

## **3.4 Mitigations**

* **Use parameterized queries**

Rather than having user-supplied input enter directly into the query, utilize “pre-prepared” queries that limit the possibilities of entry of harmful characters or queries. This only works where clauses such as WHERE, INSERT or UPDATE are present. For queries involving table or column names, utilize the second mitigation measure detailed below.

Note: that for a parameterized query to be effective in preventing SQL injection, the string that is used in the query must always be a hard-coded constant, and must never contain any variable data from any origin.

* **Sanitize user-supplied input**

Quite similarly to the command injection vulnerability identified earlier, implement strong user-supplied input validation using methods such as using a whitelist of acceptable characters (input) that the application will accept or that the input contains only alphanumeric characters, no other syntax or whitespace.

**Chapter 4 :Cross Site Scripting**

**4.1 What is Cross Site Scripting ?**

Cross-Site Scripting (XSS) is a web application vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. It occurs when an application fails to properly validate or sanitize user-generated input before displaying it on a webpage.

Here's an explanation of how Cross-Site Scripting works:

User Input: An application accepts user input, such as data entered into a form, user-generated content, or URL parameters.

* **Unsanitized Input:** The application fails to properly validate or sanitize the user input before including it in the HTML response sent to other users.
* **Malicious Input:** An attacker intentionally provides input that includes HTML or JavaScript code as part of the user input.
* **Script Execution:** The malicious code is included in the HTML response and executed by the victim's browser, as it appears to be legitimate code originating from the trusted application.
* **Unauthorized Actions:** The executed script can perform various malicious actions, such as stealing sensitive user information (e.g., login credentials, cookies), modifying the page content, redirecting users to malicious websites, or even launching further attacks against other users.

**4.2 There are different types of XSS attacks:**

* **Stored XSS:** The malicious script is permanently stored on the server and delivered to users whenever they access a specific page or view content that includes the injected script.
* **Reflected XSS:** The malicious script is embedded in a URL or other input, which is then reflected back to the user in the response from the server. The user inadvertently executes the script by clicking on a crafted link or visiting a compromised website.
* **DOM-based XSS:** The vulnerability occurs when the client-side JavaScript modifies the Document Object Model (DOM) based on untrusted data. The attacker can manipulate the DOM and inject malicious code that is executed by the victim's browser.

**4.3 Reflected Cross Site Scripting**

|  |  |
| --- | --- |
| Relative Risk | Medium |
| Vulnerability Class | User Input Handling |
| CVSS | 6.9 |
| URL | https://hack.me/dvwa-107.htmlvulnerabilities/XSS |
| Parameter | name |

Table 4-1 Cross Site Scripting[Reflected]

**4.4 Observation**

The DVWA web application for mobile does not perform output encoding of special characters to prevent Cross Site Scripting vulnerabilities. In one instance user supplied input containing special characters such as <, >, ', /, etc. is echoed back in HTML response without any output encoding performed. This allows an attacker to input malicious JavaScript which can steal victim's cookie, redirect them to other malicious website etc.

**4.5 Steps**

* Navigate to http://hack.me/ vulnerabilities/xss\_r.

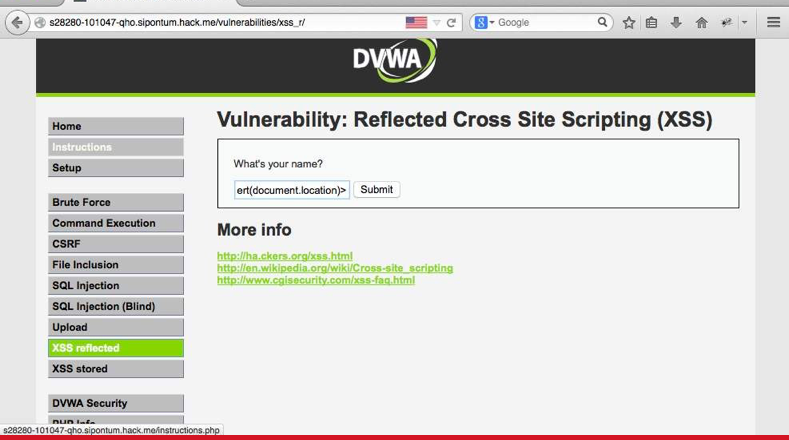


Figure-5

* Navigate to <http://hack.me/vulnerabilities/xss_r>

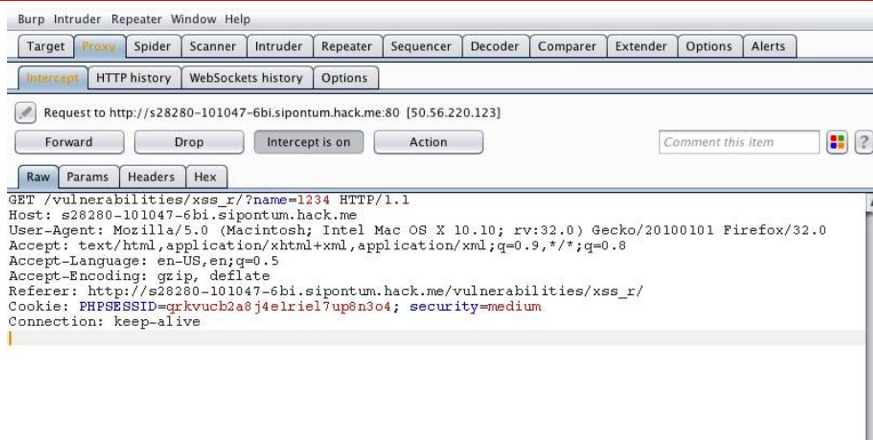


Figure-6

* You will notice a error message.

<iframe onload="alert(document.location)">

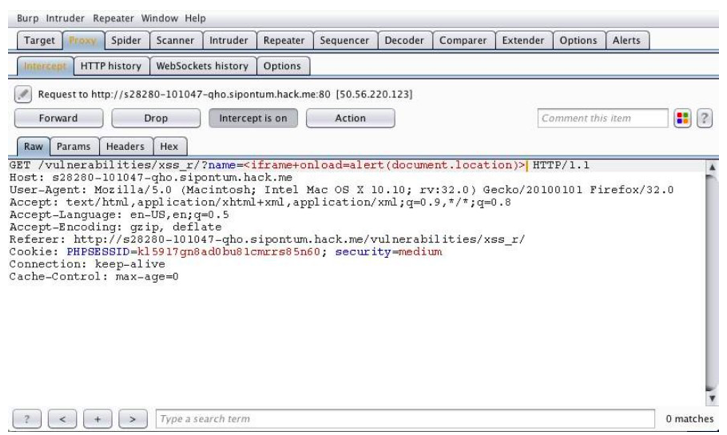


Figure-7

* A dialog box with current URL value will be shown as a result of our payload.



Figure-8

**4.6 Impact**

Reflected Cross Site Scripting is relatively complex to exploit as the malicious payload has to be send as a part of URL and user should be tricked to visit that URL. However, it has the same impact as that of a persistent XSS. In DVWA application, XSS can be used to hijack victim's session and thereby gaining complete access to his/her user account. Additionally, it can be used to redirect victim to a malicious website which may contain browser exploits or a phishing page.

**4.7 Mitigation**

* **Input Validation and Sanitization:** Implement strict input validation and sanitization on the server-side to ensure that user-supplied input is validated against expected formats and sanitized to remove or encode any potentially malicious code. Reject or sanitize any input that does not conform to the expected format.
* **Output Encoding:** Apply proper output encoding to user-generated content or dynamic data before displaying it in web pages. Use output encoding techniques specific to the context where the data is being rendered, such as HTML encoding, URL encoding, or JavaScript encoding. This prevents the execution of embedded scripts.
* **Context-Aware Output Escaping:** Understand the context in which the user input will be rendered and apply the appropriate output escaping technique for that context. Different contexts, such as HTML attributes, JavaScript code, or CSS styles, require specific escaping methods to neutralize the potential injection of malicious code.
* **Content Security Policy (CSP):** Implement a Content Security Policy that defines the types of content that can be loaded and executed on a webpage. Utilize CSP directives, such as "script-

**4.8 Stored XSS**

* Stored XSS arises when an application receives data from an untrusted source and includes that data within its later HTTP responses in an unsafe way.
* The data in question might be submitted to the application via HTTP requests; for example, comments on a blog post, user nicknames in a chat room, or contact details on a customer order. In other cases, the data might arrive from other untrusted sources.

**4.9 Steps:**

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

**4.10 Proof of Concept**

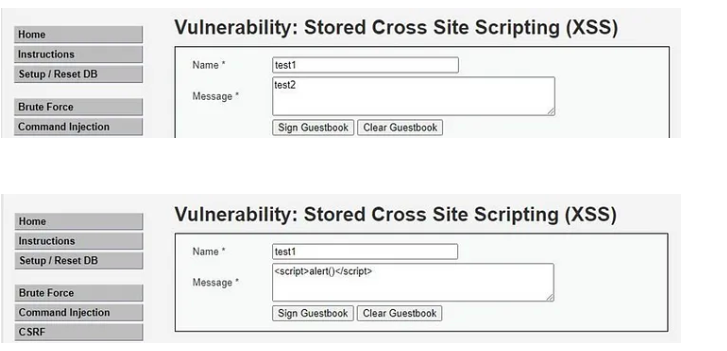


Figure-9

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

## **Mitigation**

* Filter input on arrival
* Encode data on output
* Use appropriate response headers
* Content Security policy.

**Chapter 5: INFORMATION LEAKAGE THROUGH HTTP RESPONSE HEADERS**

|  |  |
| --- | --- |
| Relative Risk | Low |
| Vulnerability Class | Http Security |
| CVSS | 1.0 |
| URL | https://hack.me/ |
| Parameter | Not Applicable |

**Table 5-1[HTTP Header Response Table]**

**5.1 Observation**

The web server hosting DVWA application is misconfigured due to which application server version is exposed to end users.

**5.2 Steps**

* Request any web page of the application and observe the response headers through BURP suite as shown below.

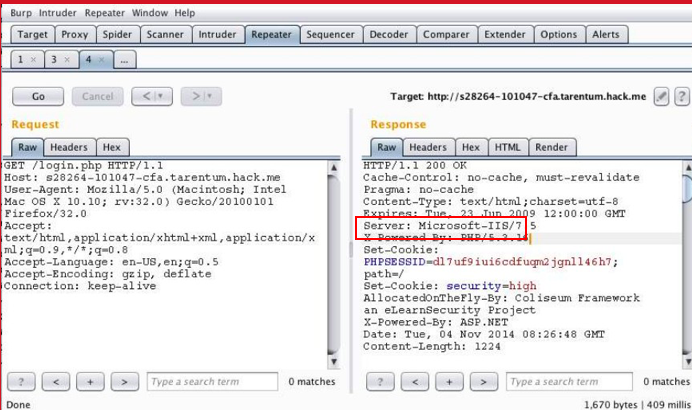


Figure-10

**5.3 Impact**

Attacker can know the version of the PHP running on the web server by the header and can search for the known vulnerabilities of PHP 5.4.23 for further exploitation. There is no direct impact to business with this vulnerability but falls under security best practices.