MINOR PROJECT

Find out the vulnerabilities and prepare a report intentionally vulnerable web application.

Table of contents

- Description of the vulnerability
- The URL in which the vulnerability has been found.
- The parameters which are vulnerable
- Payload used to trigger the vulnerability
- Observation slides containing step by step information to replicate the exploit.
- Business impact of the vulnerability, explaining in detail how the vulnerability can cause damage to organization
- Remediations/Countermeasures/
- Recommendations on how to fix the vulnerability.
- Reputed references for the vulnerabilities

Vulnerability Assessment

Summary Test Report

Description of the vulnerability

SQLi Injection

SQLi injection is a type of Cyber Attack that targets the security of a web applications database by injecting malicious SQL code into user-input fields. The goal is to manipulate the Applications database queries, potentially gaining Unauthorized access to sensitive information or performing Unintended actions.

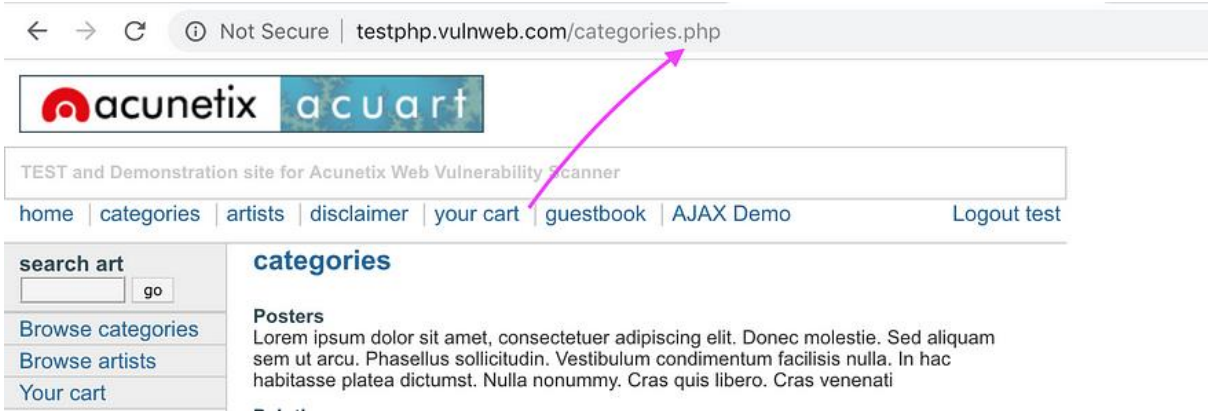
In a typical scenario, a web application collects user Input through forms or URL parameters and uses this input to construct SQL queries for database interactions. If the application doesn't properly validate or sanitize user input, attackers can insert SQL commands into these input fields,

tricking the application into executing unintended database queries.

To prevent SQL injection, web developers should use Parameterized queries or prepared statements, which ensure that user input is treated as data rather than executable code. Regular security audits and input validation practices are crucial to identify and mitigate potential vulnerabilities in web applications.

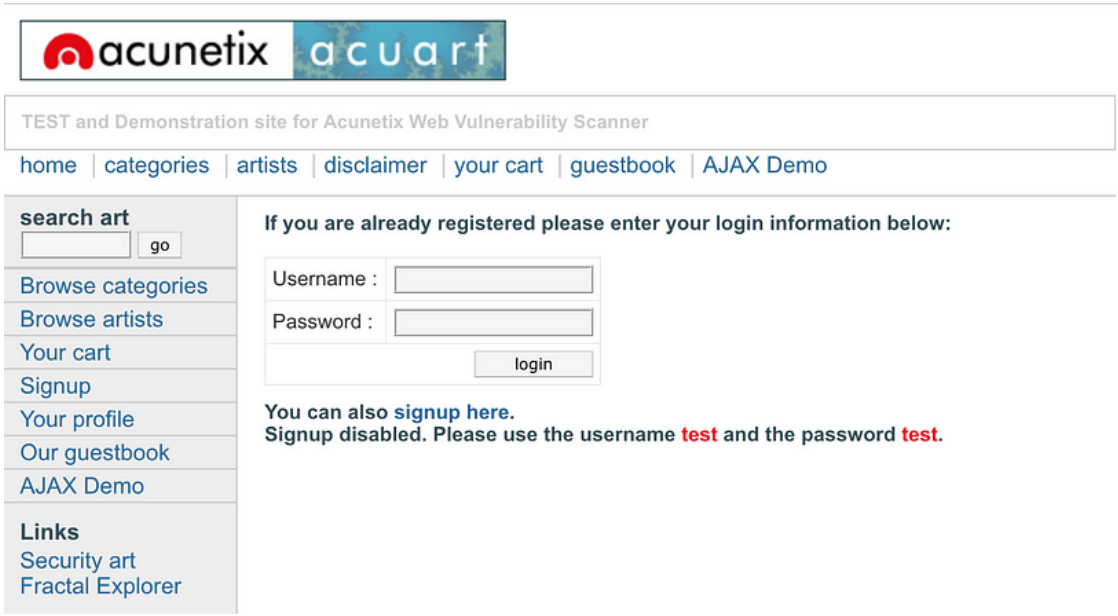
The URL in which the vulnerability has been found

First of all lets start by gathering data or ‘Reconnaissance’ navigating through the site notice the .php extension in the URL. When sites use PHP they are going to be using the LAMP stack. The LAMP stack means: Linus as the operating System, Apache as the web server, MySQL as the database, and PHP as the programming language.



Note: the .php extension as you browse the site

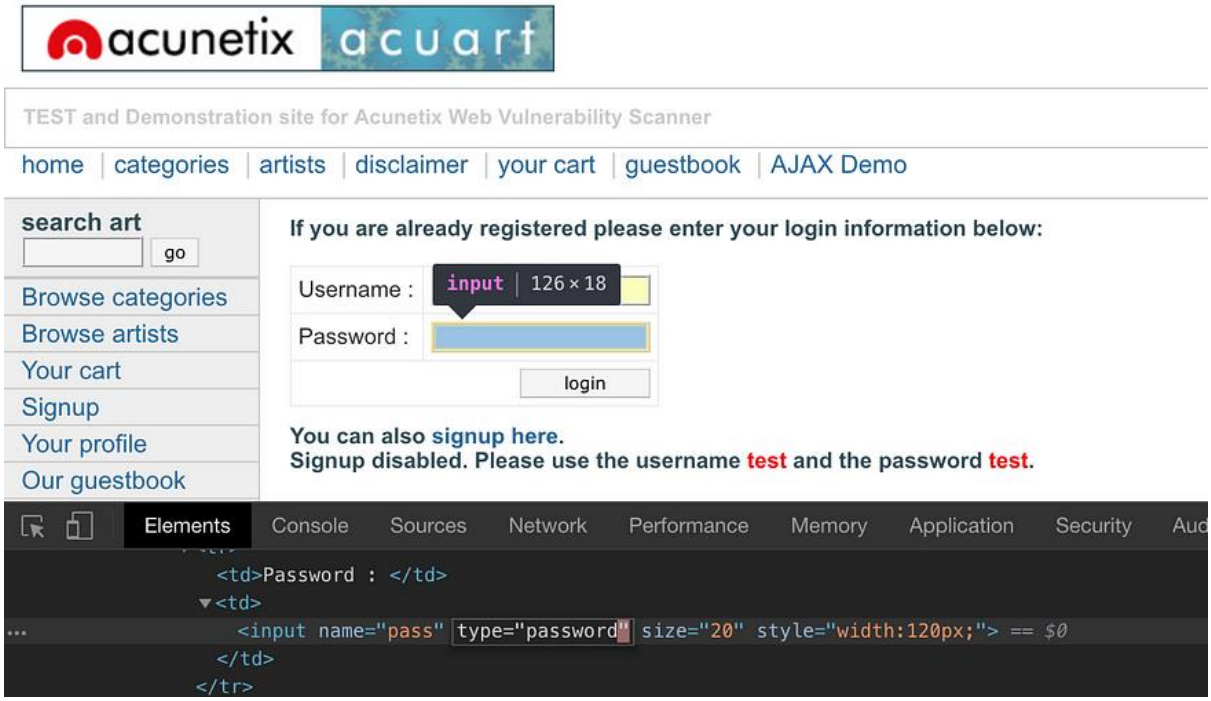
SQL Injection (SQLi) on the login page to bypass the password



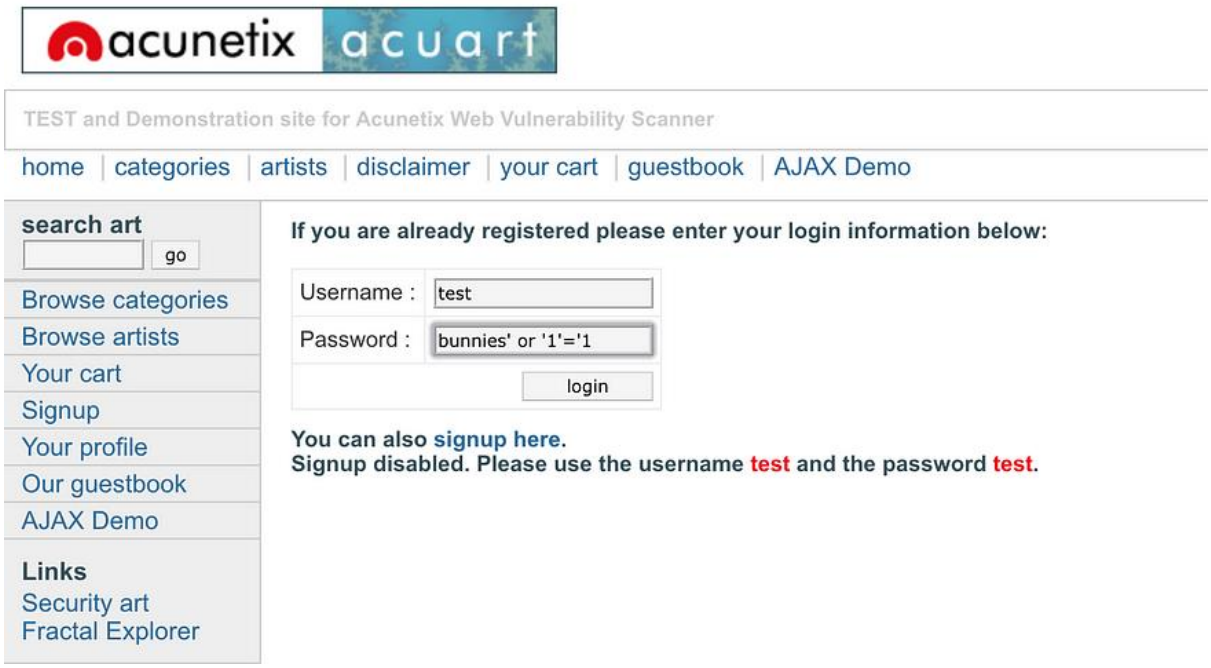
By default the username and password are both test!

Using the chrome developer tools we can inspect the Password inout element by hovering over it, right click then

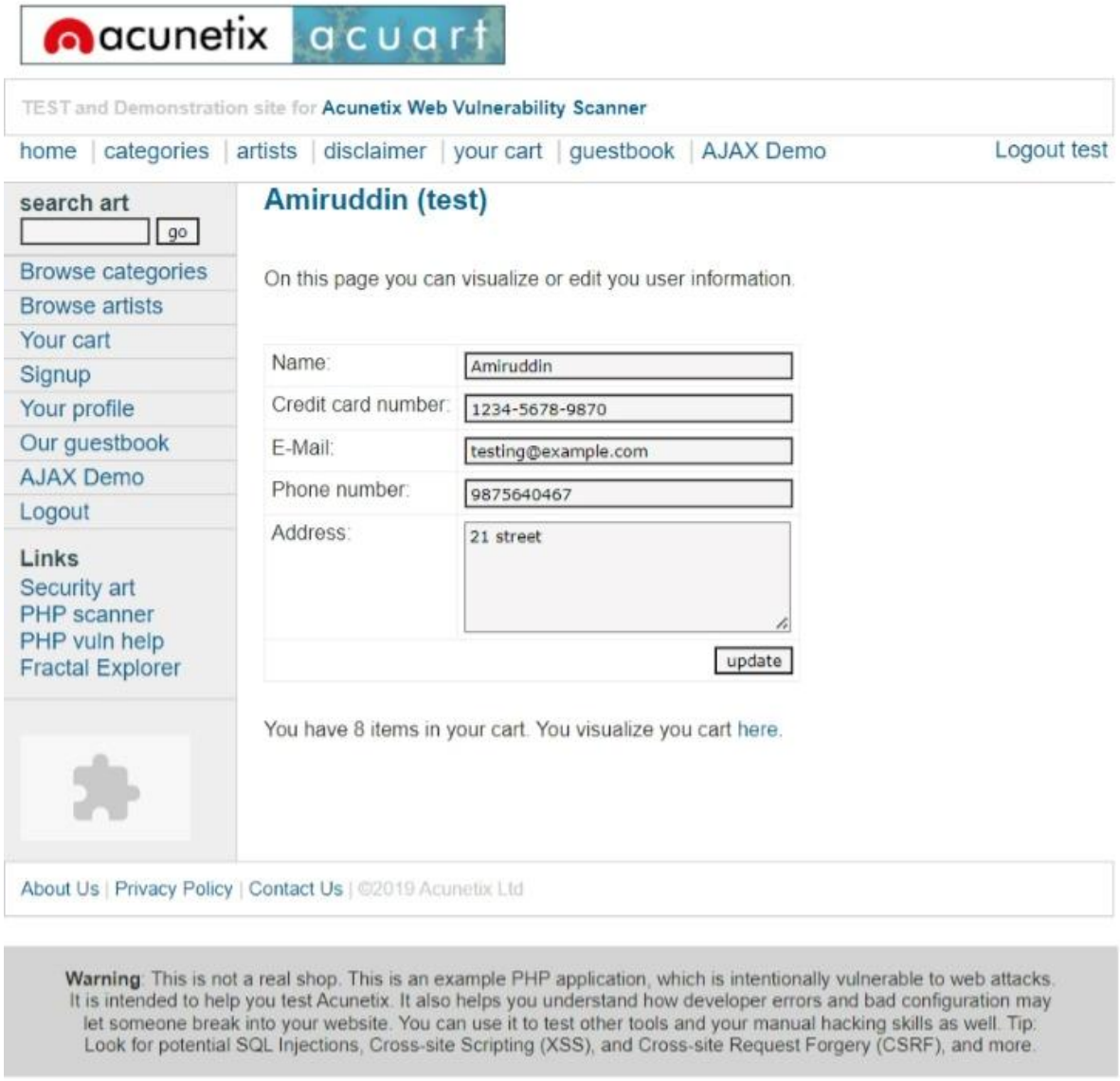
select inspect. Change the HTML value from text to password. Next if you remember SQL stands for Structured Query Language used to communicate with databases.



Change the type=”password” to type=”text” in order to see the text when you type



Instead of typing the original username: test and password: test we can use an SQLi to by pass the password.



Testing for SQLi in URL parameters

Based tips and advice the SQL injections which will be more straight forward to find will be available via the URL.

Manually navigate the page and find a place where the site is listing, displaying results, search for items. Because it will most likely mean the site has to talk to its database to retrieve the results! Which gives us the attackers an opening

to manipulate the search parameters!

For example the site we are testing:

http://testphp.vulnweb.com/listproducts.php?cat=2

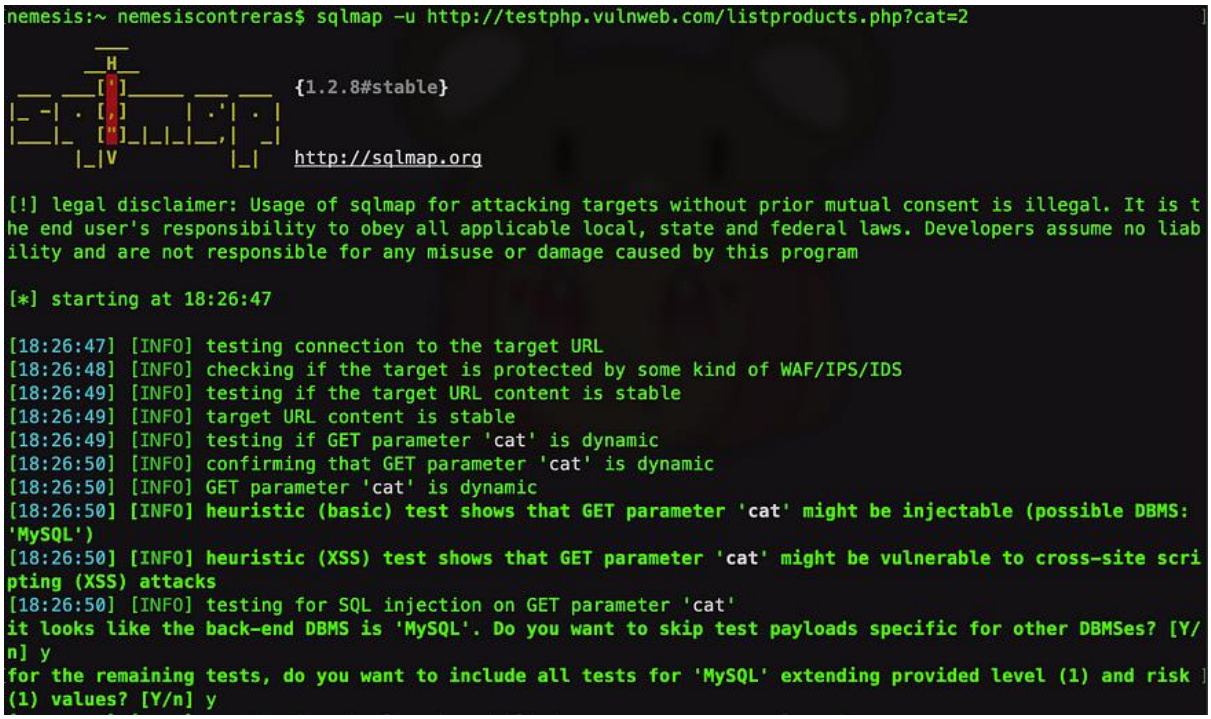
There parameter cat=2 is seems like an awesome opening to check if we can trip up the database. Most famous way to trip up the database is with the plain simple.

Result after entering the (note: the URL encodes the into a %27 so it becomes ?cat=2%27)

So lets turn to the SQLi automation SQLmap. SQLmap is a tool We can use after finding URL with parameters which you

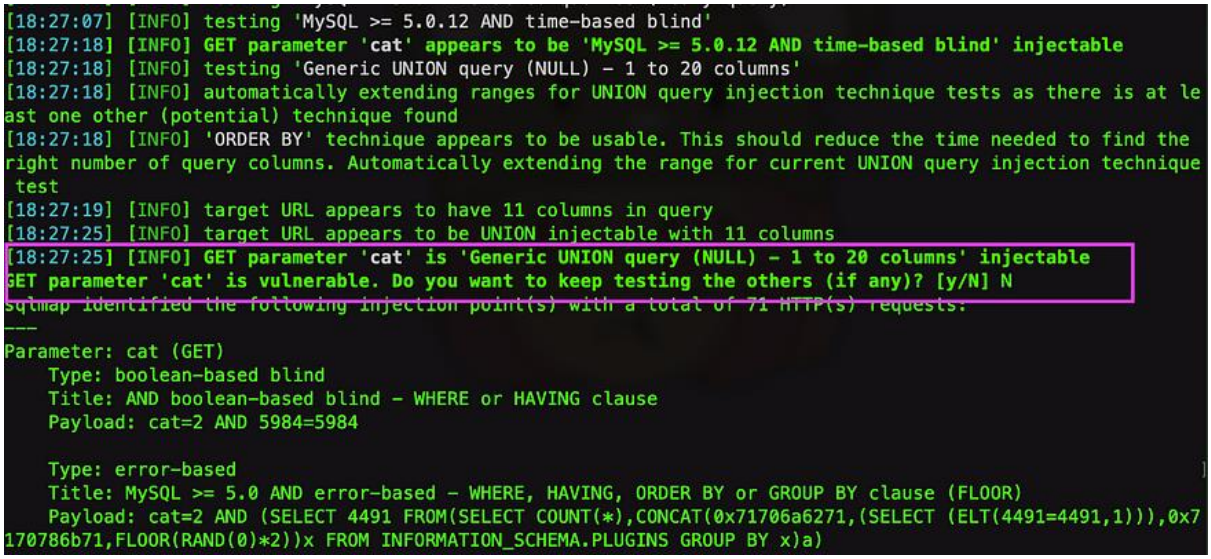
suspect can be exploited. In our example:

http://testphp.vulnweb.com/listproducts.php?cat=2 is what we would supply into SQLmap. So lets give it a go.



Command typed is: sqlmap -u http://testphp.vulnweb.com/listproducts.php?cat=2

After SQL map identifies the backend database to be mySQL Continue to begin automatic testing for SQLi!



SQLmap confirms our discovery which stated the parameter cat was injectable!

The results are what we expected that the cat parameter is Definitely vulnerable to SQLi commands such as **UNION SELECT etc!**

Business Impact of the vulnerability

In detail how the vulnerability can cause damage to organization

The Business impact of a vulnerability can be significant Leading to data breaches, financial losses, reputational Damage, and legal consequences. It may result in downtime,

Loss of customer trust, and increased cybersecurity costs for remediation. Regular risk assessments and robust security measures are crucial for minimizing such impacts.

The business impact of a vulnerability can manifest in various ways:

1. Data breaches

   Exploiting vulnerabilities can lead to unauthorized access and theft of sensitive data, potentially compromising customer information, financial records, or intellectual property.

2. Financial losses

   Breaches often result in direct financial repercussions, including costs associated with incident response, system restoration, legal fees, and regulatory fines.

3. Reputational Damage

   Customers and stakeholders may lose trust in a company that fails to secure its systems, resulting in lasting damage to the brands reputation. Rebuilding trust can be a challenging and time-consuming process.

4. Operational Distribution

Exploitation of vulnerabilities can cause downtime and disrupt normal business operations, impacting productivity and potentially leading to financial losses.

5. Legal Consequences

Failure to protect customer data or comply with data protection regulations may result in legal actions, fines, and regulatory penalties, adding to the overall financial impact.

A vulnerability can cause significant damage to an organization through various means:

1. Unauthorized Access

Exploiting a vulnerability can allow unauthorized Individuals to gain access to sensitive systems, Networks, or data, leading to potential data Breaches and unauthorized use of critical Information.

2. Intellectual Property

Vulnerabilities that allow unauthorized access to Proprietary information or trade secrets can lead to Intellectual property theft, compromising the

Organizations competitive advantage and innovation.

3. Regulatory Non-compliance

   Failure to address vulnerabilities and adhere to Industry regulations may result in non-compliance. Regulatory bodies may impose fines and penalties, further straining the organizations financial resources.

4. Supply Chain Risks

   Vulnerabilities within an organization can extend to Its supply chain partners, potentially impacting the entire ecosystem. This interconnected increases the overall risk and potentially damage caused by a security incident.

5. Manipulation

   Vulnerabilities may enable attackers to steal or manipulate sensitive data, including customer information, financial records or intellectual property, resulting in financial losses and damage to the organizations reputation.

Remediations/Recommendations on how to fix the vulnerability.

To address vulnerabilities and enhance the overall security posture of an organization, several remediation measures and recommendations can be implemented:

1. Regular Security Audits and Assessments

   Conducts regular security audits and assessment to Identify and address vulnerabilities in systems, Networks, and applications.

2. Patch management

   Keep software, operating systems, and applications up to date by promptly applying security patches. Establish a systematic patch management process.

3. Network Segmentation

   Implement network segmentation to isolate critical Systems and limit the potential impact of a security Breach.

4. Access Control

   Enforce strong access controls, including the principle of least privilege. Ensure that users have only necessary permissions for their roles.

5. Security Awareness Training

Educate employees about security best practices, social Engineering threats, and the importance of maintaining a secure environment.

6. Encryption

Implement encryption for sensitive data both in transit and at rest to protect information from unauthorized access.

7. Intrusion Detection and Prevention Systems (IDPS)

Deploy IDPS to monitor network and system activities, detect suspicious behaviour, and prevent potential security incidents.

8. Endpoint security

Utilize endpoint protection solutions to secure individual devices, detect malware, and prevent unauthorized access.

9. Incident Response Plan

Develop and regularly update an incident response plan outlining steps to be taken in the event of a security incident. Conduct regular drills to ensure preparedness.

References for the vulnerabilities

1) National Vulnerability Database (NVD)
   https://nvd.nist.gov/
2) Computer Emergency Response Team Coordination Centre (CERT/CC)
   https://www.cert.org/
3) Common Vulnerabilities and Exposures (CVE) System
   https://cve.mitre.org/

File Inclusion Vulnerability Attack Using DVWA Web Application

What is File Inclusion Attack?

It is an attack that allows an attacker to include a file on the web server through a PHP script. This vulnerability arises when a web application lets the client submit input into files or upload files to the server. A file include vulnerability is distinct from a generic directory traversal attack, in that directory traversal is a way of gaining unauthorized file system access, and a file inclusion vulnerability subverts how an application loads code for execution. Successful exploitation of a file include vulnerability will result in remote code execution on the web server that runs the affected web application.

This can lead to the following attacks:

1. Code execution on the web server
2. Cross Site Scripting Attacks (XSS)
3. Denial of service (DOS)
4. Data Manipulation Attacks

DVWA Web Application

The Damn Vulnerable Web Application is a deliberately Insecure web application designed for educational and Training purposes in the filed of web security. It provides a hands-on platform for security professionals, students, and developers to practice and enhance their skills in identifying and mitigating various web vulnerabilities.

It includes:

1) Security Levels
2) Vulnerabilities
3) Testing Environment
4) Educational Purpose
5) User Authentication
6) Community Support

It’s crucial to note that DVWA should only be used in Controlled environments, and any testing should be Conducted with proper authorization. Unauthorized security Testing can be illegal and unethical. Always follow ethical Guidelines and seek permission before using DVWA or similar Intentionally vulnerable applications.

There are two types of File Inclusion

1) Local File Inclusion

2) Remote File Inclusion

Local File Inclusion allow an attacker to read files on the Victim machine. This can be very dangerous because if the Web server is misconfigured and running with high privileges, The attacker may gain access to sensitive information. If the Attacker is able to place code on the web server through Other means, then they may be able to execute arbitrary Commands.

Remote File Inclusion vulnerabilities are easier to exploit but Less common. Instead of accessing a file on the local Machine, the attacker is able to execute code hosted on own machine.

Remote File inclusion and Local File Inclusion are Vulnerabilities that are often found in poorly-written web applications. These vulnerabilities occur when a web application allows the user to submit input into files or upload files to the server. In order to demonstrate these attacks, we will be using the Damn Vulnerable web application.

There are some pre-requisites required:

1. **XAMPP**
2. **Damn Vulnerable Web Application (DVWA)**

Local file Inclusion in Action

Performing LFI attacks through different levels of difficulty Offered by DVWA

Difficulty: LOW

Now start your machine and login to DVWA, then go to DVWA security tab and change the difficulty level to low.

Go to file Inclusion tab and change the URL from Index.php to ?

page = ../../../../../../etc/passwd.

Change the URL from ?page =../../../../../../**proc/version.**

Difficulty: MEDIUM

Now we can try the exploits we used in low difficulty. We can Notice that we can't read files like before using the directory Traversal method. So, as we can see in the below image of Source page, the server is more secure and is filtering the '../' or '..\'pattern. Let's try to access the file without '../' or '..\'.

File Inclusion Source

vulnerabilities/fi/source/medium.php

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
$file = str_replace( array( "http://", "https://" ), "", $file );
$file = str_replace( array( "../", "..\"" ), "", $file );

?>
```

Change include.php to /etc/passwd

Now, change the URL from? **page=/etc/passwd** to **?page=/proc/version.**

As we can see, it worked by directly entering the name of the File. Let's level up the difficulty to HIGH.

Difficulty: HIGH

Change the difficulty to HIGH and try all exploits from

Medium difficulty, and we will notice none of them will works Because the target is more secure, as it is only accepting “include.php” or inputs staring with word “file”. If we try anything else, it will show that “File not Found”.

File Inclusion Source

vulnerabilities/fi/source/high.php

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

In this level of security, we can still gather sensitive info using the “File” URI scheme.

Change the URL from **include.php** to

?page=file:////ect/passwd

Now we will get the data of **/etc/passwd** file.

This is how you can exploit file inclusion vulnerability using Local files on the webserver.

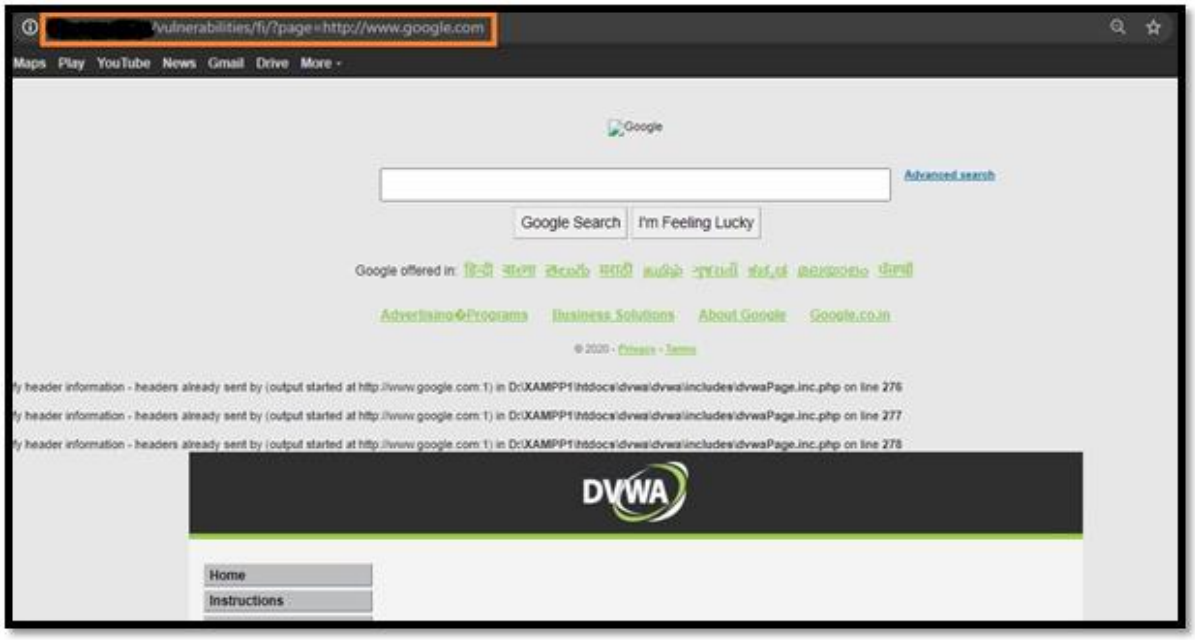
Let’s try to exploit this vulnerability using remote files.

Difficulty: LOW

Change the difficulty to low and go to file inclusion tab.

Let’s change include.php to http://www.google.com so the Final URL will be like this,

?page=http://www.google.com



Difficulty: MEDIUM

Change the difficulty to medium and check as we did it in the Low difficulty. We will notice, it's not working anymore. The Target is now filtering "http" and "https" as shown in source Page.

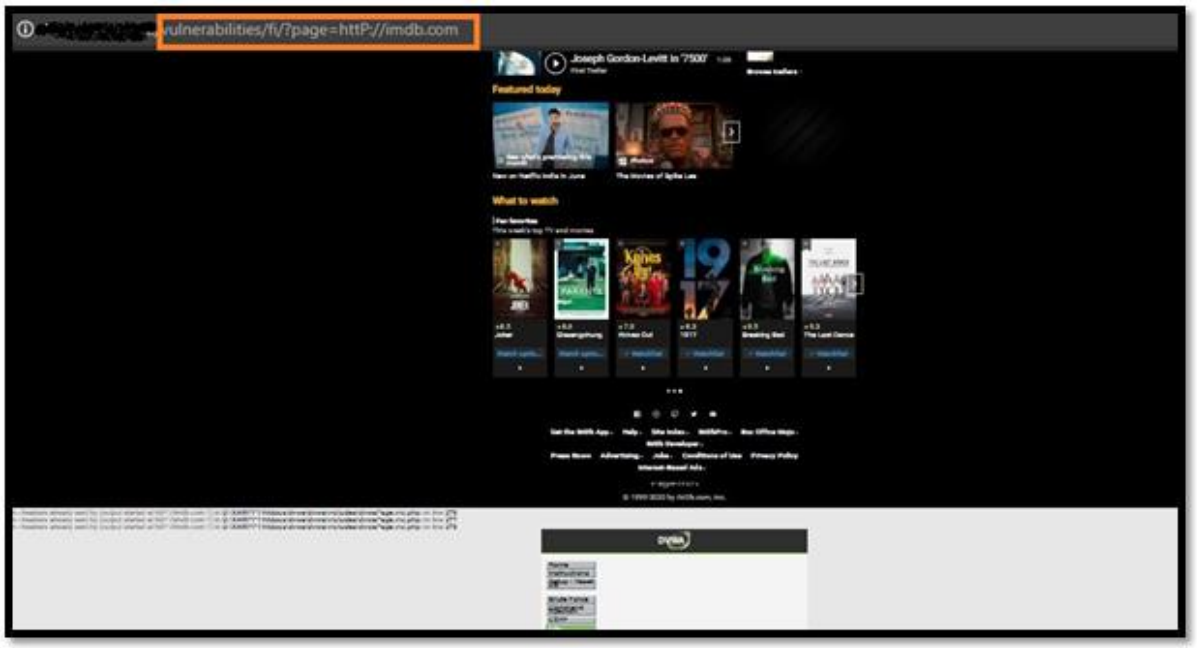
File Inclusion Source

vulnerabilities/fi/source/medium.php

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
$file = str_replace( array( "http://", "https://" ), "", $file );
$file = str_replace( array( "../", "..\"" ), "", $file );

?>
```

So try the attack with “HTTP” or any one word in caps and it Will work.

?page=httP://imdb.com



Difficulty: HIGH

```php
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

We can't exploit the high difficulty using RFI as we can see in source page, we know that the target web-server is only accepting "include.php" or anything that's starting with the word "file" that's why we can't include anything from an outside server.

Points to secure against File Inclusion Vulnerability

a) Strong Input Validation
b) A whitelist of acceptable inputs
c) Reject any inputs that do not strictly conform to Specifications.
d) For filenames, use stringent whitelist that limits the Character set to be used.
e) Exclude directory separators such as "/"
f) Use a whitelist of allowable file extensions.
g) Environment hardening

h) Develop and run your code in the most recent versions of PHP available.
i) Configure your PHP applications so that it does not use use register globals
j) Run your code using the lowest privileges.

References for the vulnerabilities

1. OWASP (Open Web Application Security Project)

   This list includes common web application security risks, some of which involve file-related vulnerabilities.

2. NIST National Vulnerability Database (NVD)

   Search for specific vulnerabilities related to file attacks.

3. CVE (Common Vulnerabilities and Exposures)

   Search for specific file-related vulnerabilities using CVE IDs.

4. Security Blogs and Journals

   Regularly check blogs and journals from security experts and organizations like Krebs on Security or Schneier on Security.