Vulnerability Assessment Summary Report

Prepared By:

Learning Circle : Cyber olizz

Jibin Gigi

Diya Benny

Jose Thomas

Saniya Mary Jacob

Prithviraj R

Learning Circle ID: CYCYBSJC1234

1. Vulnerabilities found for server-side software

Summary

Severity : High

Confidence : Firm

Host : http://testphp.vulnweb.com/

Path : /

REPORT

CVSS	CVE	SUMMARY	EXPLOIT	AFFECTED SOFTWARE
6.8	CVE-2015-9253	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.	N/A	php 5.6.40
7.5	CVE-2019-9641	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.	N/A	php 5.6.40
6.5	CVE-2022-31629	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.	N/A	php 5.6.40
5.8	CVE-2017-7272	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.	N/A	php 5.6.40
7.5	CVE-2017-8923	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.	N/A	php 5.6.40

Risk description

These vulnerabilities expose the affected applications to the risk of unauthorized access to confidential data and possibly to denial of service attacks. An attacker could search for an appropriate exploit (or create one himself) for any of these vulnerabilities and use it to attack the system.

Recommendation

We recommend you to upgrade the affected software to the latest version in order to eliminate the risk of these vulnerabilities.

2.Cross -domain Referrer Leakage

Summary

Severity : Information

Confidence : Certain

Host : http://testphp.vulnweb.com/

Path : /listproducts.php

Issue detail

The page was loaded from a URL containing a query string:

- http://testphp.vulnweb.com/listproducts.php

The response contains the following links to other domains:

- http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
- http://www.acunetix.com/
- https://www.acunetix.com/
- https://www.acunetix.com/blog/articles/prevent-sql-injection-vulnerabilities-in-php-applications/
- https://www.acunetix.com/vulnerability-scanner/

Issue background

The Referer header is an HTTP header added to requests for resources, indicating the URL of the resource from which the request originated. It is typically included in cross-domain requests, even if the resource is on a different domain. Sensitive information, like session tokens, may be transmitted to the other domain, potentially leading to security

compromises. Browsers may withhold the Referer header in certain situations, but it should not be relied upon to protect the originating URL.

Issue remediation

Applications should never transmit any sensitive information within the URL query string. In addition to being leaked in the Referer header, such information may be logged in various locations and may be visible on-screen to untrusted parties. If placing sensitive information in the URL is unavoidable, consider using the Referer-Policy HTTP header to reduce the chance of it being disclosed to third parties.

References

Referer Policy

Vulnerability classifications

CWE-200: Information Exposure

Request

```
GET /listproducts.php?cat=1 HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:86.0) Gecko/20100101 Firefox/86.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Referer: http://testphp.vulnweb.com/categories.php
Upgrade-Insecure-Requests: 1
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Sat, 13 Mar 2021 04:34:24 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/5.6.40-38+ubuntu20.04.1+deb.sury.org+1
Content-Length: 7880

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html><!-- InstanceBegin template="/Templates/main_dynamic_template.dwt.php" codeOutsideHTMLIsLoc
...[SNIP]...
<p>
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" codebase="http://download.macromedia.com/pub/shockwave/cabs/flash
/swflash.cab#version=6,0,29,0" width="107" height="66">
<param name="movie" value="Flash/add.swf">
...[SNIP]...
<div id="siteInfo"> <a href="http://www.acunetix.com">About Us</a>
...[SNIP]...
```

3.Cross -site scripting (self)

Severity : Medium

Confidence : Firm

Host : http://testphp.vulnweb.com/

Path : /robots.txt

Issue detail

The application's response demonstrates the possibility of injecting new HTML tags into the returned document, despite attempts to identify a full proof-of-concept attack for arbitrary JavaScript injection. It is recommended to manually examine the application's behavior for unusual input validation or other obstacles.

Issue background

Cross-site scripting vulnerabilities occur when data is copied from a request and echoed into the application's response in an unsafe way. An attacker can use this vulnerability to create a request that executes JavaScript code within the user's browser, potentially stealing their session token or login credentials. The attacker can induce users to issue the crafted request through various methods, such as sending a malicious URL, submitting the link to popular websites, or creating an innocuous website. The security impact of cross-site scripting vulnerabilities depends on the application's nature, data and functionality, and other applications within the same domain and organization. High-risk applications, such as online banking, should always be considered high-risk.

Issue remediation

In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defenses:

- Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal names should consist of alphabetical and a small range of typographical characters, and be relatively short; a year of birth should consist of exactly four numerals; email addresses should match a well-defined regular expression. Input which fails the validation should be rejected, not sanitized.
- User input should be HTML-encoded at any point where it is copied into application responses. All HTML metacharacters, including < > " ' and =, should be replaced with the corresponding HTML entities (< > etc).

In cases where the application's functionality allows users to author content using a restricted subset of HTML tags and attributes (for example, blog comments which allow limited formatting and

linking), it is necessary to parse the supplied HTML to validate that it does not use any dangerous syntax; this is a non-trivial task.

References

- Cross-site scripting
- Reflected cross-site scripting
- Using Burp to Find XSS issues

Vulnerability classifications

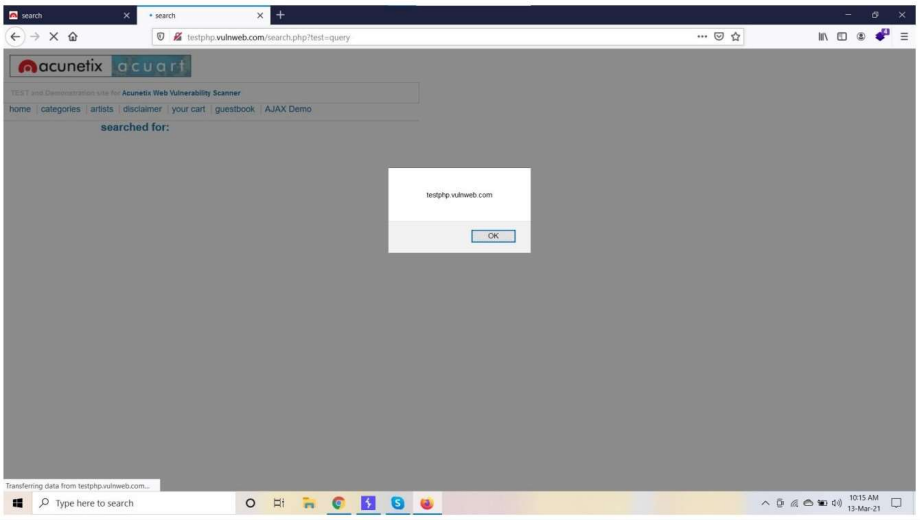
- CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
- CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
- CWE-116: Improper Encoding or Escaping of Output
- CWE-159: Failure to Sanitize Special Element

Request

```
POST /search.php?test=query HTTP/1.1
Host: testphp.vulnweb.com
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-US,en-GB;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/88.0.4324.150 Safari/537.36
Connection: close
Cache-Control: max-age=0
Referer: http://testphp.vulnweb.com/index.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 61

searchFor=<script>alert(document.domain)</script>&goButton=go
```

Response



4. Insecure client access policy

Severity : Medium

Confidence : Firm

Host : http://testphp.vulnweb.com/

Path : /crossdomain.xml

Risk description

The `crossdomain.xml` file controls the access of externally hosted Flash scripts to this website. The external websites which are permitted to read content from this website via Flash are specified in the XML tag `<allow-access-from>`. If the value of this tag is too permissive (ex. wildcard), it means that any Flash script from an external website could access content from this website, including confidential information of users. The `clientaccesspolicy.xml` file specifies that other websites can read content from this website - which is normally denied by the Same Origin Policy. If the allowed domains are too permissive (ex. wildcard) then any external website will be able to read content (including sensitive information) from this website. Flash is not supported anymore and this poses a risk only if the user's clients use older browsers, making them vulnerable to their information being accessed by a malicious external Flash script.

Recommendation

We recommend to carefully review the content of the policy file and permit access only for legitimate domains.

Issue remediation

Any inappropriate entries in the Flash cross-domain policy file should be removed.

Vulnerability classifications

- CWE-942: Overly Permissive Cross-domain Whitelist

Request

```
GET /crossdomain.xml HTTP/1.1
Host: testphp.vulnweb.com
Connection: close
```

Response

```
HTTP/1.1 200 OK
Server: nginx/1.19.0
Date: Sat, 13 Mar 2021 04:21:46 GMT
Content-Type: text/xml
Content-Length: 224
Last-Modified: Tue, 11 Sep 2012 10:30:22 GMT
Connection: close
ETag: "504f12be-e0"
Accept-Ranges: bytes

<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" to-ports="*" secure="false"/>
...[SNIP]...
```