Vulnerability Assessment Summary Report

Submitted By

MuLearn Circle : Cyber Tribe

Code:CYCYBPRN123

Summary

This Vulnerability Assessment report presented by muLean Circle Cyber Tribe contains reports of various vulnerabilities findings present in the website http://testphp.vulnweb.com

1. **Cross-site scripting (self)**

Summary

Severity:Medium
Confidence:Firm
Host:http://testphp.vulnweb.com/
Path:/robots.txt

Issue in detail

The name of an arbitrarily supplied URL parameter is copied into the HTML document as plain text between tags. The payload was submitted in the name of an arbitrarily supplied URL parameter. This input was echoed unmodified in the application's response. This behaviour demonstrates that it is possible to inject new HTML tags into the returned document. An attempt was made to identify a full proof-of-concept attack for injecting arbitrary JavaScript but this was not successful. You should manually examine the application's behaviour and attempt to identify any unusual input validation or other obstacles that may be in place.

Issue background

Reflected cross-site scripting vulnerabilities arise when data is copied from a request and echoed into the application's immediate response in an unsafe way. An attacker can use the vulnerability to construct a request that, if issued by another application user, will cause JavaScript code supplied by the attacker to execute within the user's browser in the context of that user's session with the application. The attacker-supplied code can perform a wide variety of actions, such as stealing the victim's session token or login credentials, performing arbitrary actions on the victim's behalf, and logging their keystrokes. Users can be induced to issue the attacker's crafted request in various ways. For example, the attacker can send a victim a link containing a malicious URL in an email or instant message. They can submit the link to popular web sites that allow content authoring, for example in blog comments. And they can create an innocuous looking web site that causes anyone viewing it to make arbitrary cross-domain requests to the vulnerable application (using either the GET or the POST method). The security impact of cross-site scripting vulnerabilities is dependent upon the nature of the vulnerable application, the kinds of data and functionality that it contains, and the other applications that belong to the same domain and organization. If the application is used only to display non-sensitive public content, with no authentication or access control functionality, then a cross-site scripting flaw may be considered low risk. However, if the same application resides on a domain that can access cookies for other more security-critical applications, then the vulnerability could be used to attack those other applications, and so may be considered high risk. Similarly, if the organization that owns the application is a likely target for phishing attacks, then the vulnerability could be leveraged to lend credibility to such attacks, by injecting Trojan functionality into the vulnerable application and exploiting users' trust in the organization in order to capture credentials for other applications that it owns. In many kinds of application, such as those providing online banking functionality, cross-site scripting should always be considered high risk.

Issue remediation

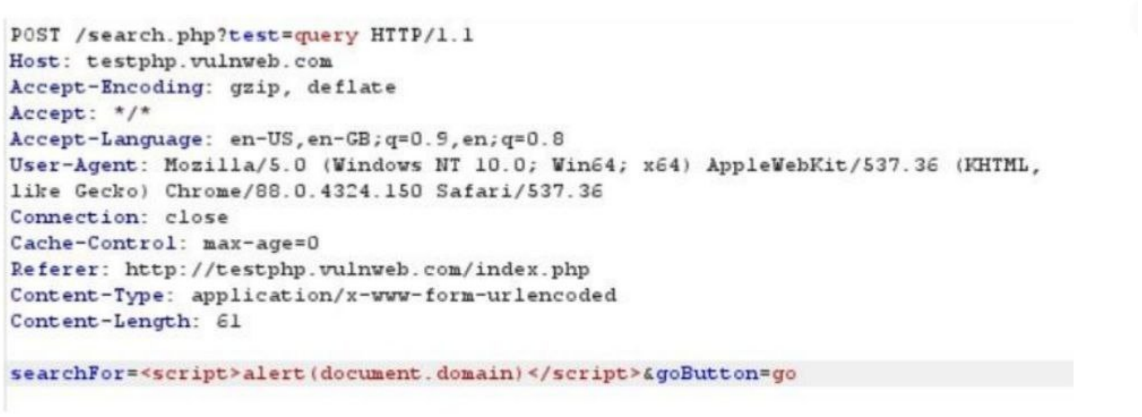
In most situations where user-controllable data is copied into application responses, cross-site scripting attacks can be prevented using two layers of defenses: ● Input should be validated as strictly as possible on arrival, given the kind of content that it is expected to contain. For example, personal 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

POC

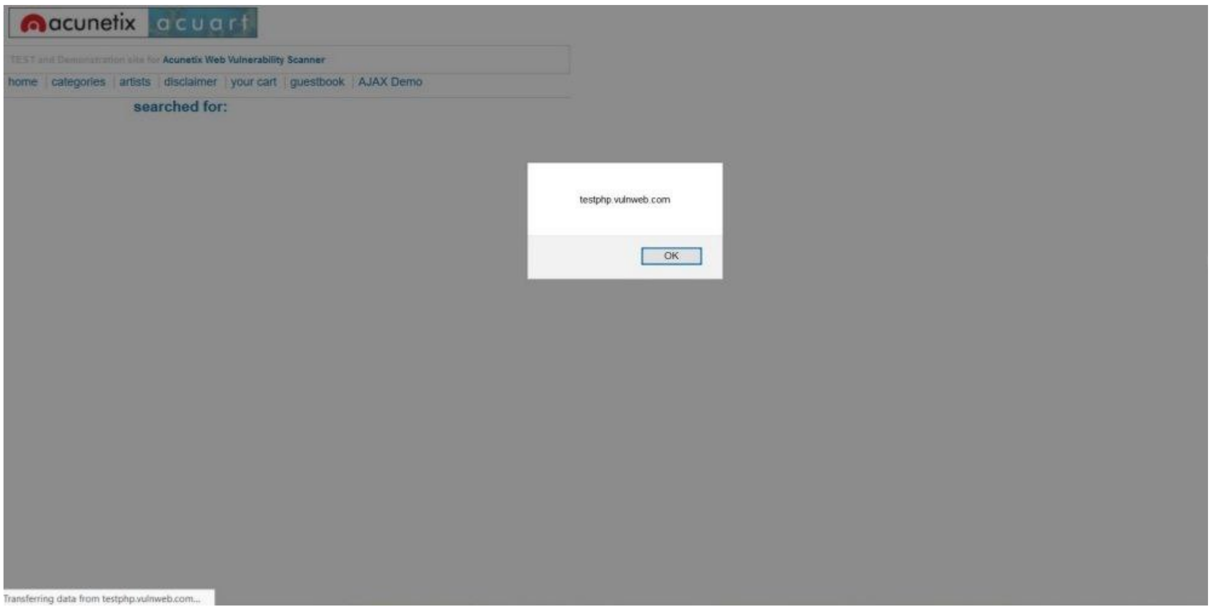
Request



```
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:



acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
OK
Transferring data from testphp.vulnweb.com...

searched for:

testphp.vulnweb.com

2.Flash cross-domain policy

Summary

Severity:High
Confidence: Certain
Host:http://testphp.vulnweb.com/
Path:/crossdomain.xml

Issue in detail

The application publishes a Flash cross-domain policy which allows access from any domain. Allowing access from all domains means that any domain can perform two-way interaction with this application. Unless the application consists entirely of unprotected public content, this policy is likely to present a significant security risk.

Issue background

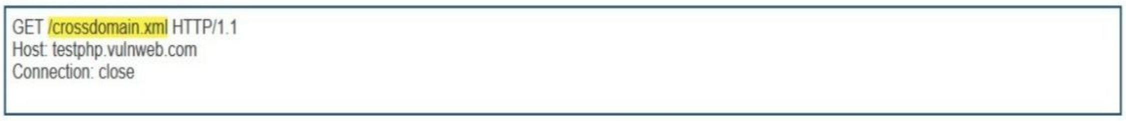
The Flash cross-domain policy controls whether Flash client components running on other domains can perform two-way interaction with the domain that publishes the policy. If another domain is allowed by the policy, then that domain can potentially attack users of the application. If a user is logged in to the application, and visits a domain allowed by the policy, then any malicious content running on that domain can potentially gain full access to the application within the security context of the logged in user. Even if an allowed domain is not overtly malicious in itself, security vulnerabilities within that domain could potentially be leveraged by a third-party attacker to exploit the trust relationship and attack the application that allows access. Any domains that are allowed by the Flash crossdomain policy should be reviewed to determine whether it is appropriate for the application to fully trust both their intentions and security posture.

Issue remediation

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

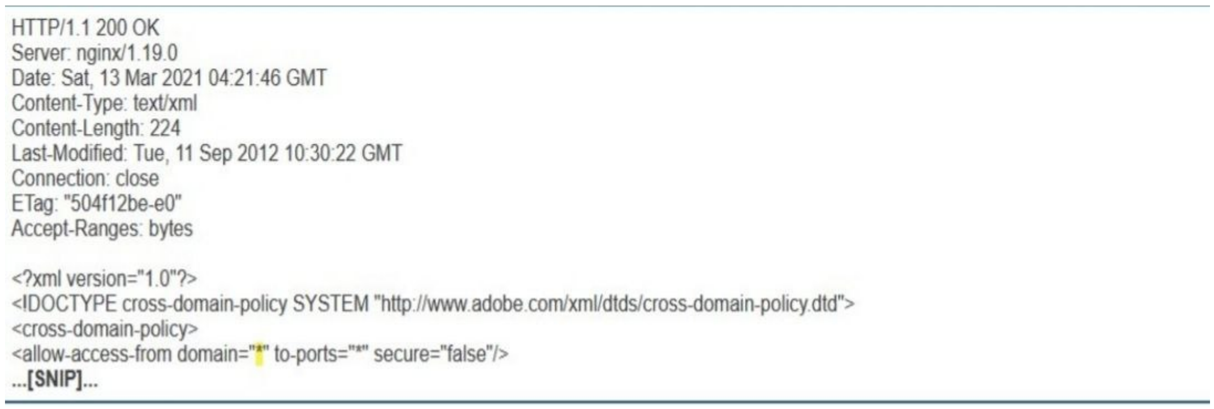
POC

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]...
```

3.Email addresses disclosed

There are 4 instances of this issue: ●

- /categories.php
- /guestbook.php
- /listproducts.php

Issue background

The presence of email addresses within application responses does not necessarily constitute a security vulnerability. Email addresses may appear intentionally within contact information, and many applications (such as web mail) include arbitrary third-party email addresses within their core content.

However, email addresses of developers and other individuals (whether appearing on-screen or hidden within page source) may disclose information that is useful to an attacker; for example, they may represent usernames that can be used at the application's login, and they may be used in social engineering attacks against the organization's personnel. Unnecessary or excessive disclosure of email addresses may also lead to an increase in the volume of spam email received.

Issue remediation

Consider removing any email addresses that are unnecessary, or replacing personal addresses with anonymous mailbox addresses (such as helpdesk@example.com). To reduce the quantity of spam sent to anonymous mailbox addresses, consider hiding the email address and instead providing a form that generates the email server-side, protected by a CAPTCHA if necessary.

Vulnerability classifications

● CWE-200: Information Exposure

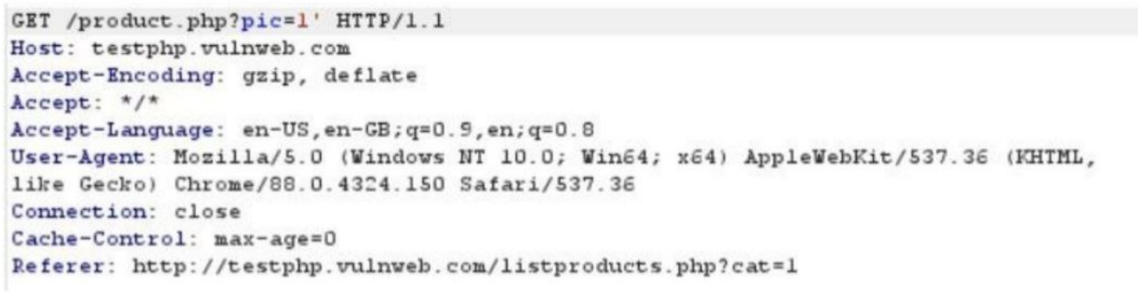
4.Blind SQL

Issue Background

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).[1] SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

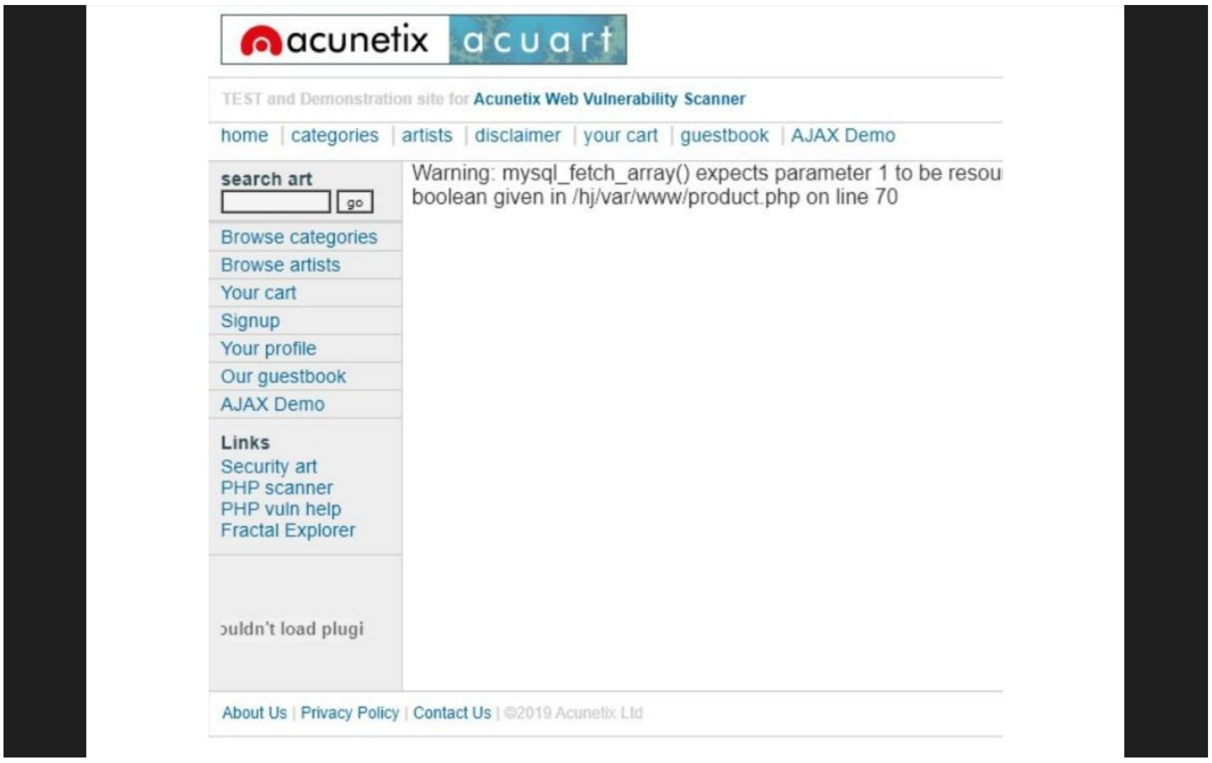
POC

Request:



```
GET /product.php?pic=1' 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/listproducts.php?cat=1
```

Response


acunetix acuart
TEST and Demonstration site for Acunetix Web Vulnerability Scanner
home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
search art
go
Browse categories
Browse artists
Your cart
Signup
Your profile
Our guestbook
AJAX Demo
Links
Security art
PHP scanner
PHP vuln help
Fractal Explorer
ɔuldn't load plugi
About Us | Privacy Policy | Contact Us | ©2019 Acunetix Ltd

```
Warning: mysql_fetch_array() expects parameter 1 to be resou
boolean given in /hj/var/www/product.php on line 70
```

Conclusion:

In thoroughly examining the system's vulnerabilities, we've carefully pinpointed and assessed potential security risks. Our analysis delved into various aspects, such as network vulnerabilities, application weaknesses, and the possible exposure of sensitive information.

What we've discovered emphasises the critical need for a proactive cybersecurity approach. It's essential to address these vulnerabilities promptly to strengthen the system against potential exploits, unauthorised access, and data breaches. The recommended remediation measures, detailed in the assessment, offer a strategic roadmap to enhance overall security.

It's crucial for stakeholders to prioritise the swift implementation of these remedial actions. Moreover, maintaining ongoing vigilance and conducting regular assessments are key elements of a robust security strategy to adapt to evolving threats and vulnerabilities.

By tackling the identified issues and fostering a culture of continuous improvement in cybersecurity practices, the organization can significantly reduce its risk landscape and enhance resilience against potential security threats.