**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**

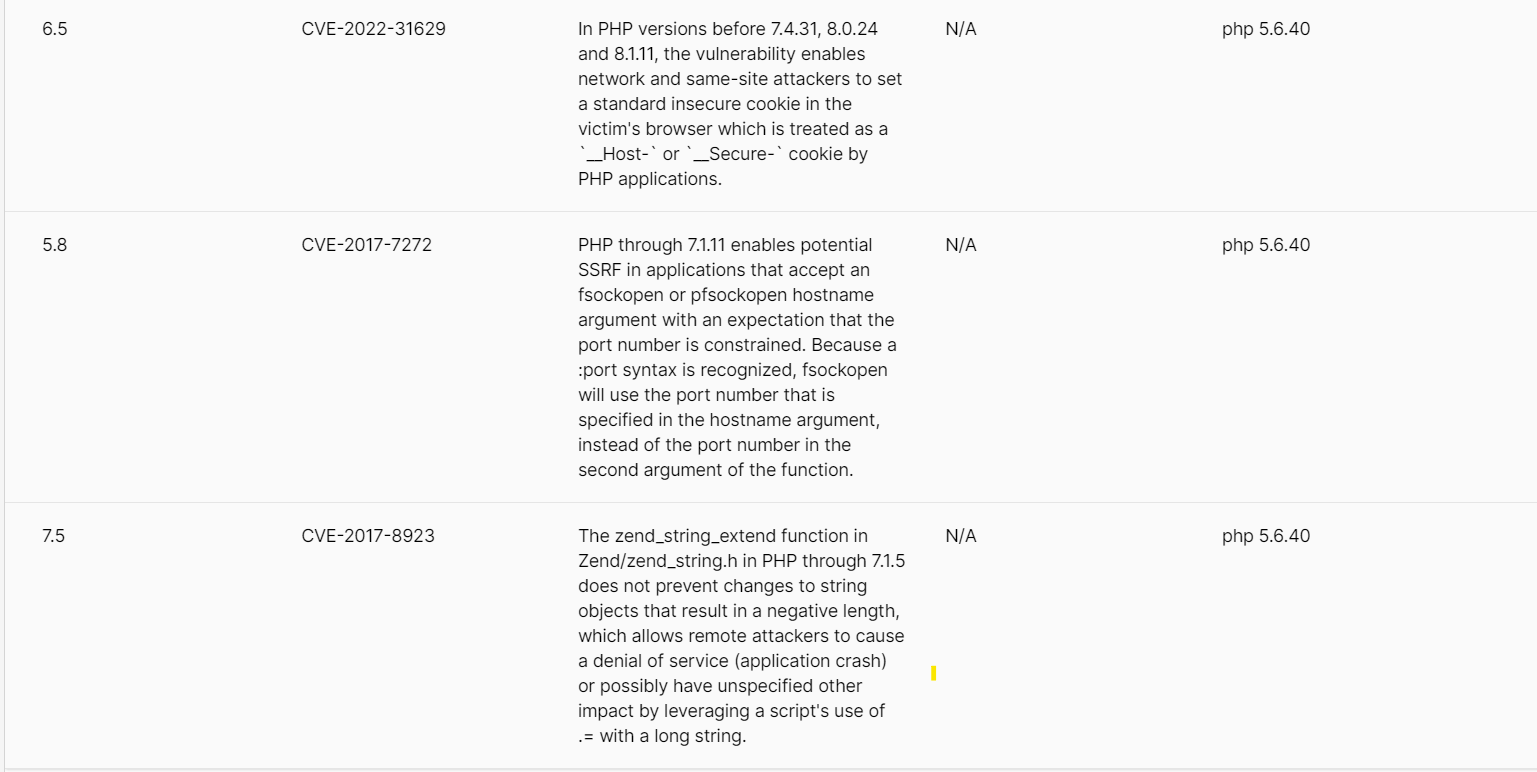
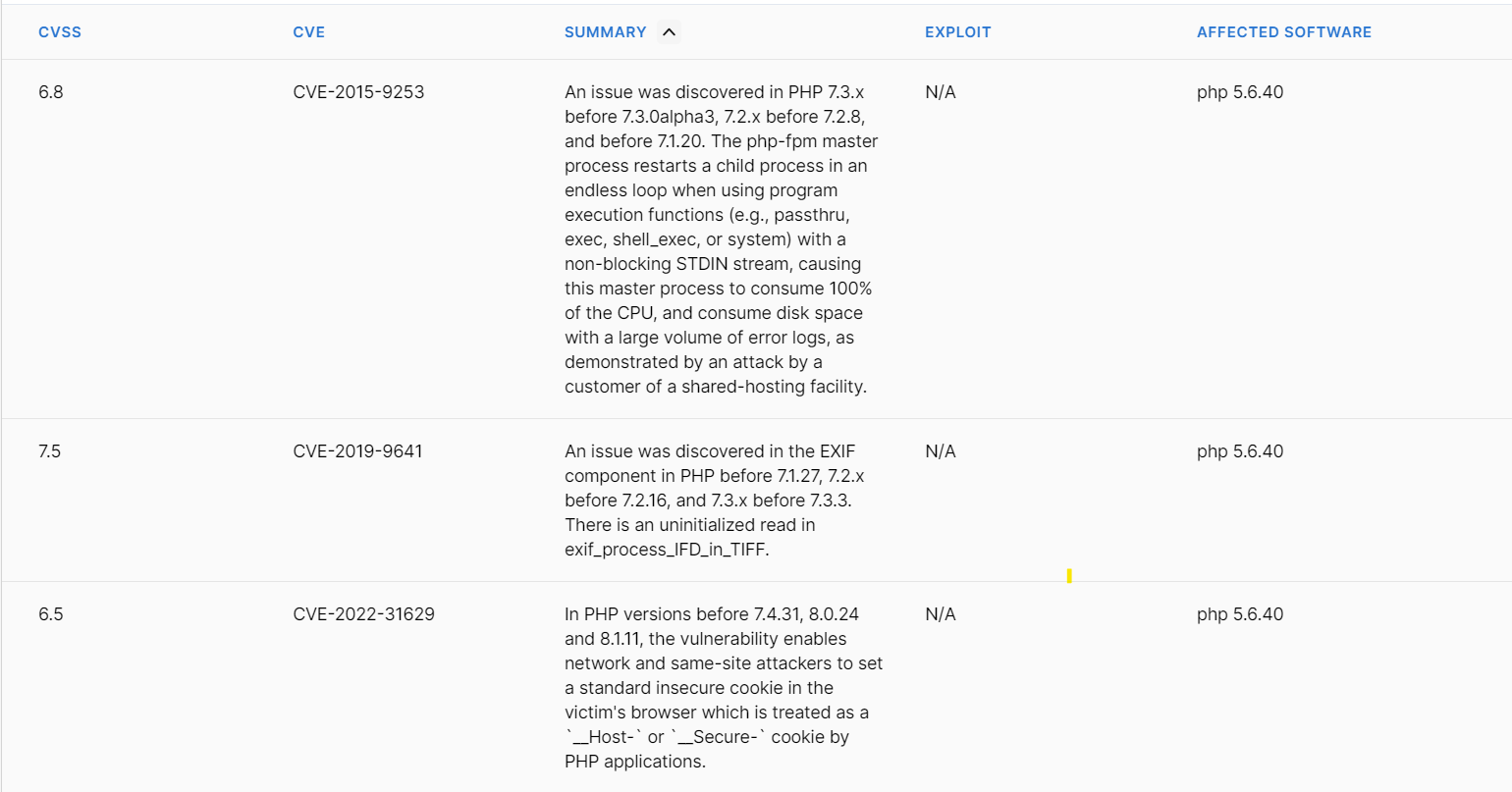
Severity : High

Confidence : Firm

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

Path : /

**REPORT**



**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.

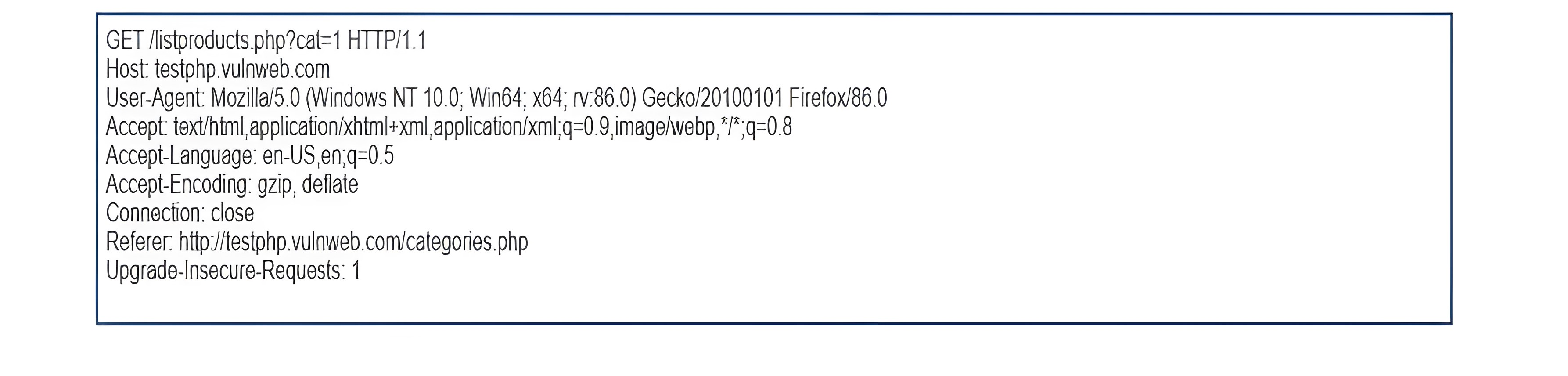
**References**

Referer Policy

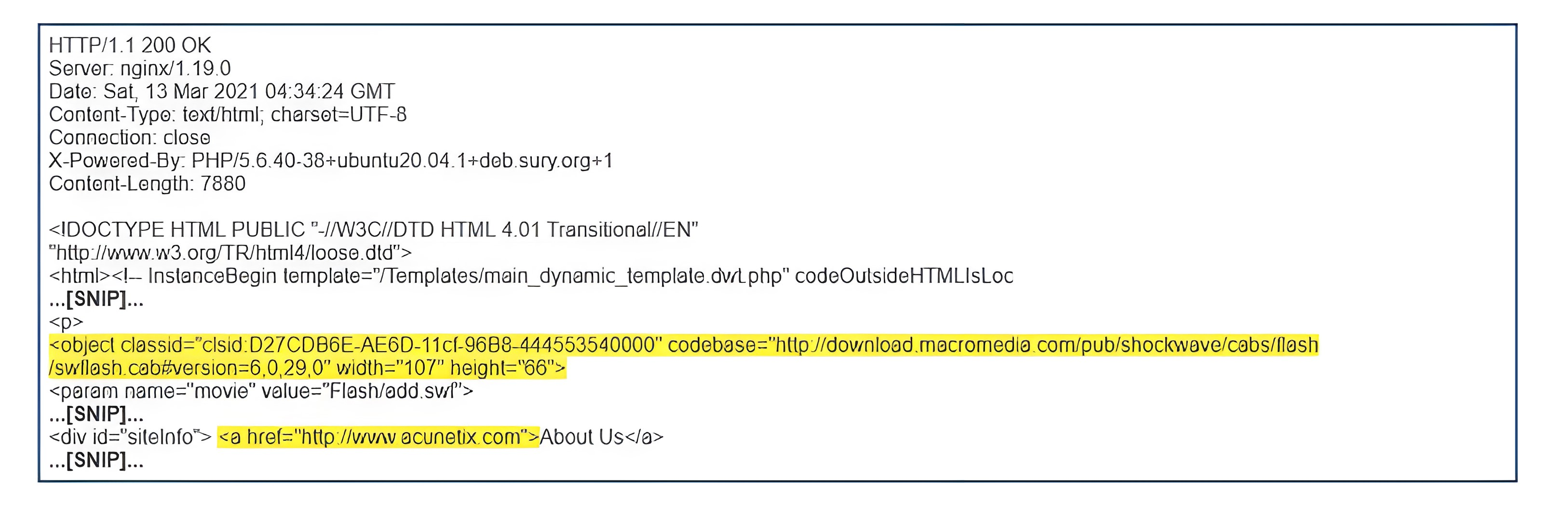
**Vulnerability classifications**

CWE-200: Information Exposure

**Request**



**Response**



**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.

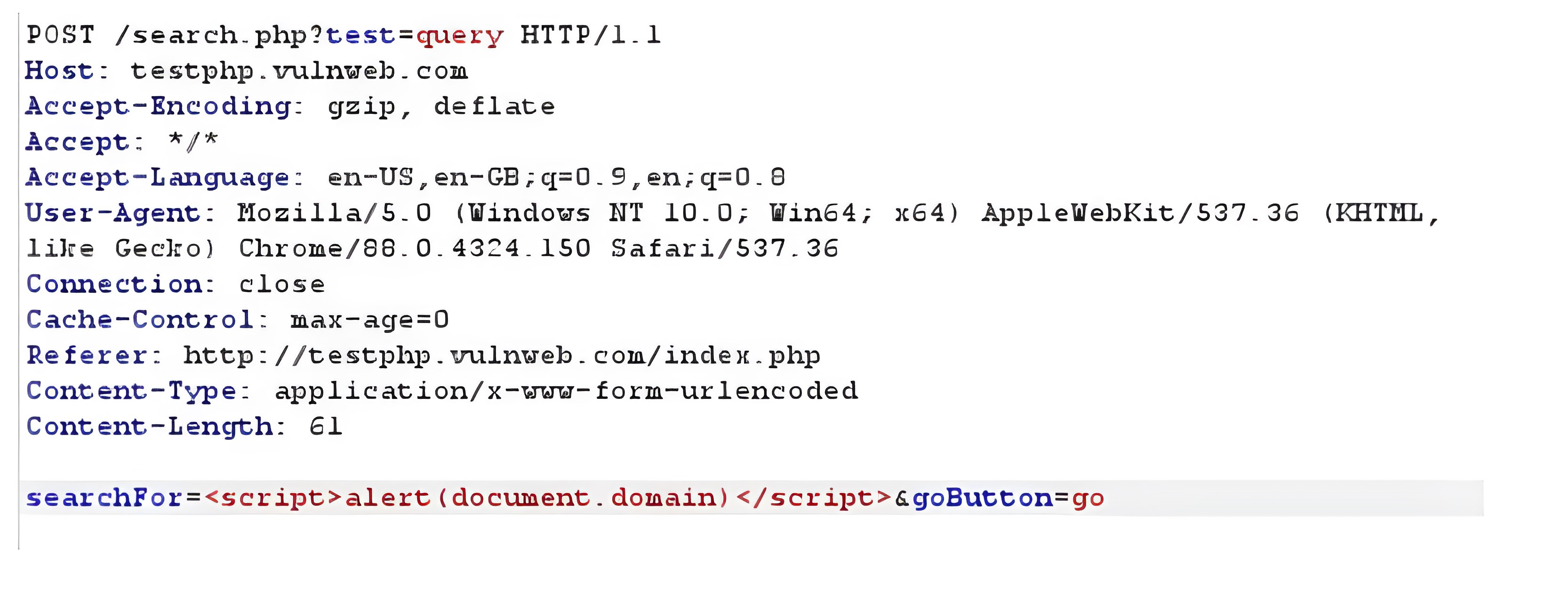
**References**

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

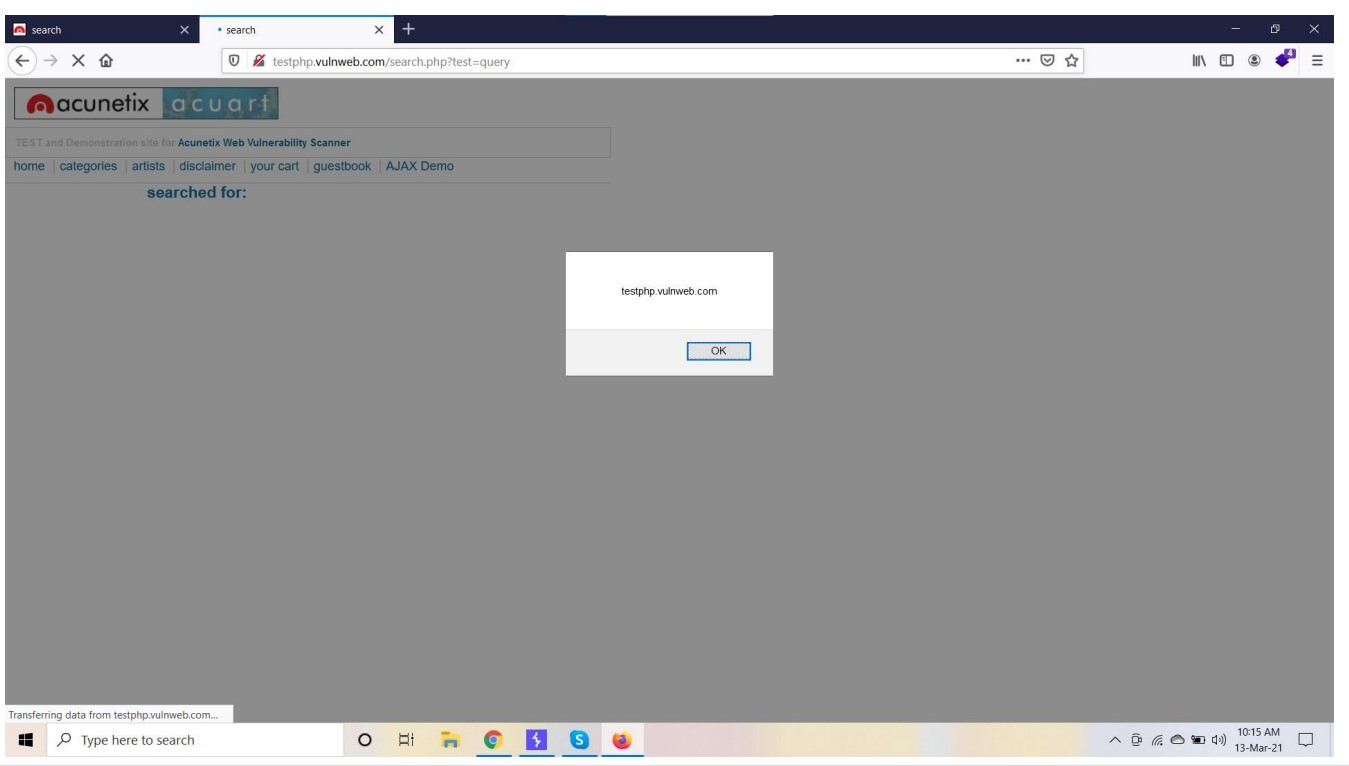
**Vulnerability classifications**

* 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**



**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**

****

**Response **