Sri Lanka Institute of Information Technology

IE2062

Web Security

Bug Bounty Report IX

Submitted by:

Student Registration Number	Student Name
IT21197550	Nihila Premakanthan

Date of Submission: 28.05.2023

Acknowledgement

I would like to express my special thanks to our mentor Ms. Chethana Liyanapathirana and Dr. Lakmal Rupansighe for their time and efforts she provided throughout the course, and for the Web Security lecture panel for guiding us through this semester and for helping us by giving examples, guidelines, and advice about the project. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to you.

Executive Summary

This report aims to provide an overview of the vulnerability identified in a particular domain. The bug bounty platform called Hackerone was used for this purpose. This report analyses the domain of Paypal (http://paypal.com/).

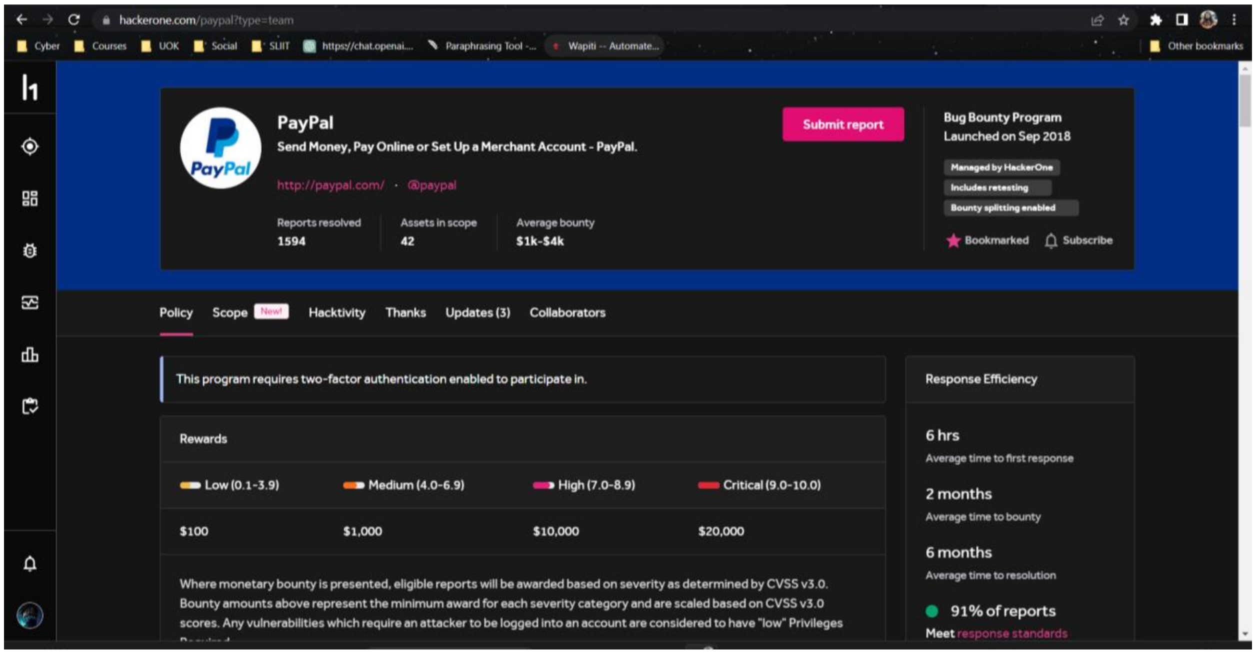
This report uses different tools to gather information detect vulnerabilities and perform penetration testing. The tool name Netsparker and Owasp Zap was mainly used to identify the vulnerability. Further this report provides the vulnerability title, vulnerability description, Affected Components, Impact Assessment, Steps to reproduce the vulnerability, proof of concept and the proposed mitigation.

By including these comprehensive details for each vulnerability, the report provides a comprehensive overview of the security weaknesses present within the system and offers actionable insights for remediation and improvement.

Contents

Introduction

Affected URL: http://paypal.com/



PayPal
Send Money, Pay Online or Set Up a Merchant Account - PayPal.
http://paypal.com/ · @paypal

Reports resolved	Assets in scope	Average bounty
1594	42	$1k-$4k

Bug Bounty Program
Launched on Sep 2018
Managed by HackerOne
Includes retesting
Bounty splitting enabled

Policy | Scope New! | Hacktivity | Thanks | Updates (3) | Collaborators

This program requires two-factor authentication enabled to participate in.

Rewards

Low (0.1-3.9)	Medium (4.0-6.9)	High (7.0-8.9)	Critical (9.0-10.0)
$100	$1,000	$10,000	$20,000

Where monetary bounty is presented, eligible reports will be awarded based on severity as determined by CVSS v3.0. Bounty amounts above represent the minimum award for each severity category and are scaled based on CVSS v3.0 scores. Any vulnerabilities which require an attacker to be logged into an account are considered to have "low" Privileges

Response Efficiency

6 hrs
Average time to first response

2 months
Average time to bounty

6 months
Average time to resolution

● 91% of reports
Meet response standards

Vulnerability title

HTTP Header Injection

Security Level:

Risk Level:
MEDIUM

Vulnerability description

A web security flaw known as HTTP Header Injection happens when an attacker is able to insert harmful text into a web application's HTTP headers. The HTTP headers are a component of the communication that takes place between a client (such a web browser) and a server, and they offer further details about the request or response.

Headers in a typical HTTP request include details like the user agent, cookies, content types, and more. The functionality of the communication between the client and the server depends on these headers. However, an application may be vulnerable to HTTP Header Injection if it does not adequately check or sanitize user-supplied input used in creating the headers.

Affected components

The communication process of a web application can be impacted by HTTP Header Injection in a number of ways. These are the parts that are frequently impacted:

- Request Headers: Attackers are able to modify the request headers that are sent by the client to the server. Headers like User-Agent, Referer, Cookie, and Content-Type fall under this category. An attacker may try to take advantage of flaws in the server-side processing of these headers by including harmful content in the headers.
- Response Headers: Headers delivered by the server in its response to the client can also be injected with malicious content by attackers. Headers like Set-Cookie, Location, Content-Disposition, and Content-Type fall under this category. Attacks like session hijacking, cache poisoning, or cross-site scripting (XSS) can result from manipulating these headers.
- Proxy Headers: The headers used for proxy communication may also be susceptible to injection if the web application communicates with a proxy server. It is possible to modify headers like X-Forwarded-For, X-Forwarded-Host, or X-Forwarded-Proto to get around security measures or fake the source IP address.
- Server-Side Processing: HTTP Header Injection can have a significant impact on the vulnerable server-side processing of HTTP headers. Security flaws may arise if the server does not correctly validate, sanitize, or manage user-supplied input in the headers. Response splitting attacks may occur, for instance, if the server immediately adds user input to the response headers without sufficient encoding.
- Downstream Systems: Systems or components that depend on the headers for processing may be affected by HTTP Header Injection. Backend applications, caching servers, load balancers, and any other system that utilizes or modifies the headers fall under this category. The usual operation of these systems can be interfered with by an injection in the headers, which could also result in security breaches.

Impact assessment

Depending on the particular environment and the method used to exploit the vulnerability, the impact of an HTTP Header Injection vulnerability can change. The following are a few possible effects of HTTP Header Injection:

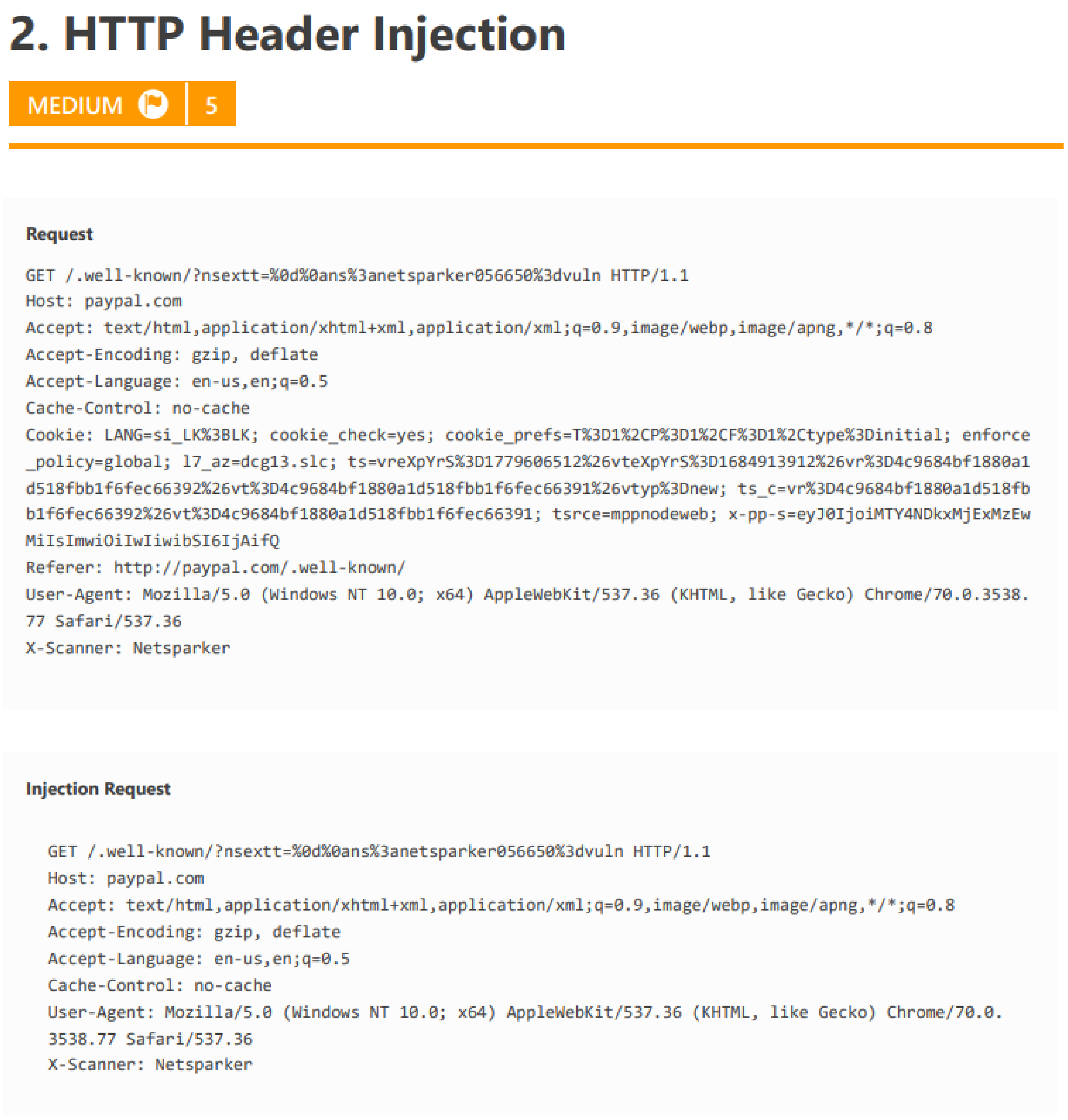
- Cross-Site Scripting (XSS): The ability to insert malicious JavaScript code into the response headers is one of the frequent effects of HTTP Header Injection. Cross-Site Scripting (XSS) attacks may result from this, in which the injected code is run while other users are viewing the compromised page. XSS can cause session hijacking, website defacement, data theft, or the spread of malware.
- Redirection and Phishing Attacks: An attacker can reroute users to a malicious website by changing the Location header in an HTTP response. This can be applied in phishing attacks to gather sensitive data like login credentials, credit card information, or personal data. Malware distribution and drive-by download assaults can both be started with redirection attacks.
- Session Hijacking: By inserting malicious headers that change session-related data, HTTP Header Injection enables an attacker to hijack user sessions. This could give the attacker the chance to pose as the victim, get access to private information without authorization, act on behalf of the user, or escalate privileges.
- Cache poisoning: By injecting headers that regulate cache behavior, HTTP Header Injection can occasionally be used to trick caching systems or proxy servers. By injecting malicious content into the cache, an attacker can launch cache poisoning attacks that give compromised or malicious replies to several users.
- Remote Code Execution (RCE): In specific circumstances, HTTP Header Injection may result in remote code execution if an application has weak server-side processing of headers and permits the execution of arbitrary code. As a result, the server may be compromised completely and sensitive information may be accessed without authorization by an attacker.

Steps to reproduce

In order to reproduce an HTTP Header Injection vulnerability, malicious content must be injected into a web application's headers in order to take advantage of that application's weak behavior. The general procedures to exploit this issue are as follows:

- Select the Target: Decide the online application or target system you believe to be vulnerable to HTTP Header Injection. This might be a system you are allowed to test or one of your own applications.
- Understand the input: Recognize the user-controllable input that is utilized to build the headers by understanding the input. Input fields, query parameters, and any other user-supplied information that is included into the headers fall under this category.
- Make a malicious payload to be injected into the headers: Make a malicious payload. The payload ought to be created to take advantage of the particular weakness you identify. For instance, the payload may contain CRLF (Carriage Return Line Feed) characters to alter the answer if you detect a response splitting issue.
- Inject the Payload: Insert the prepared payload into the appropriate header field to submit it. Various tools, browser add-ons, or manually submitting requests to the target system can all be used for this.
- Observe the Behavior: Examine the response the target system gives in response to the payload injection. Keep an eye out for any unusual behavior or clues indicating the server successfully injected and processed the payload.
- Verify the Vulnerability: Check to see if the injected payload has any unexpected effects on the system to establish the existence of an HTTP Header Injection vulnerability. For instance, determine whether the payload enabled session hijacking, triggered an XSS attack, or resulted in cache poisoning.
- Report and document: Make thorough notes of the actions you took, the payload you used, and the behavior you saw. Record your results and succinctly describe the vulnerability, along with its effect and any potential hazards. Report the vulnerability via responsible disclosure procedures to the relevant stakeholders or the application owner.

Proof of concept



2. HTTP Header Injection

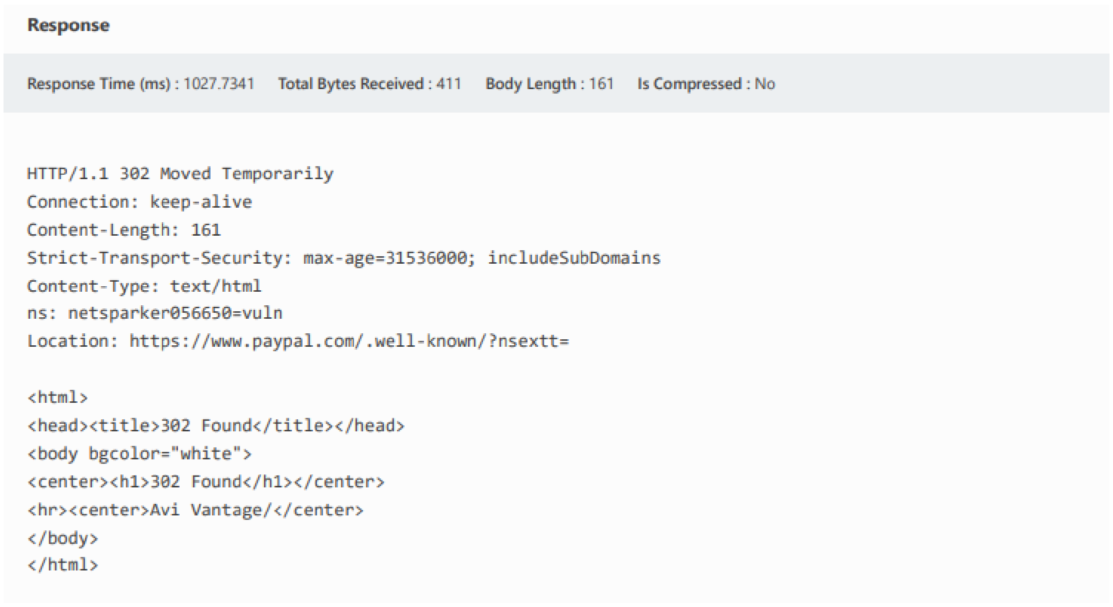
MEDIUM | 5

Request

```
GET /.well-known/?nsextt=%0d%0ans%3anetsparker056650%3dvuln HTTP/1.1
Host: paypal.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
Cookie: LANG=si_LK%3BLK; cookie_check=yes; cookie_prefs=T%3D1%2CP%3D1%2CF%3D1%2Ctype%3Dinitial; enforce
_policy=global; l7_az=dcg13.slc; ts=vreXpYrS%3D1779606512%26vteXpYrS%3D1684913912%26vr%3D4c9684bf1880a1
d518fbb1f6fec66392%26vt%3D4c9684bf1880a1d518fbb1f6fec66391%26vtyp%3Dnew; ts_c=vr%3D4c9684bf1880a1d518fb
b1f6fec66392%26vt%3D4c9684bf1880a1d518fbb1f6fec66391; tsrce=mppnodeweb; x-pp-s=eyJ0IjoiMTY4NDkxMjExMzEw
MiIsImwiOiIwIiwibSI6IjAifQ
Referer: http://paypal.com/.well-known/
User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.
77 Safari/537.36
X-Scanner: Netsparker
```

Injection Request

```
  GET /.well-known/?nsextt=%0d%0ans%3anetsparker056650%3dvuln HTTP/1.1
  Host: paypal.com
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
  Accept-Encoding: gzip, deflate
  Accept-Language: en-us,en;q=0.5
  Cache-Control: no-cache
  User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.
  3538.77 Safari/537.36
  X-Scanner: Netsparker
```

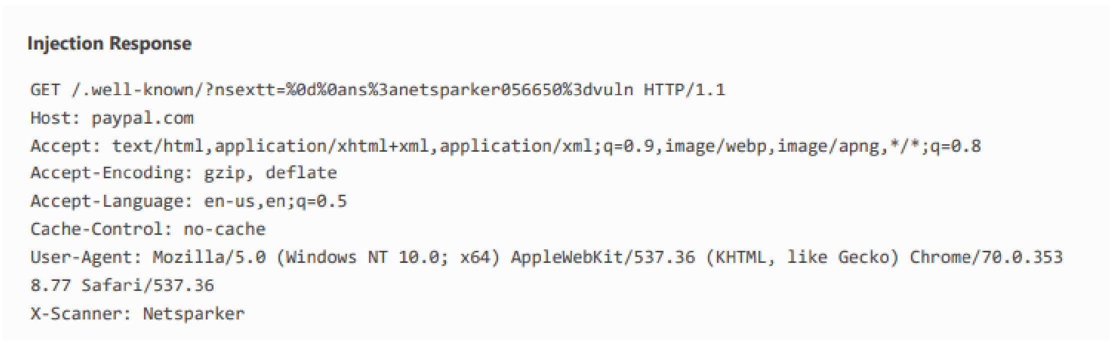


Response

Response Time (ms) : 1027.7341 Total Bytes Received : 411 Body Length : 161 Is Compressed : No

```
HTTP/1.1 302 Moved Temporarily
Connection: keep-alive
Content-Length: 161
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Type: text/html
ns: netsparker056650=vuln
Location: https://www.paypal.com/.well-known/?nsextt=

<html>
<head><title>302 Found</title></head>
<body bgcolor="white">
<center><h1>302 Found</h1></center>
<hr><center>Avi Vantage/</center>
</body>
</html>
```



Injection Response

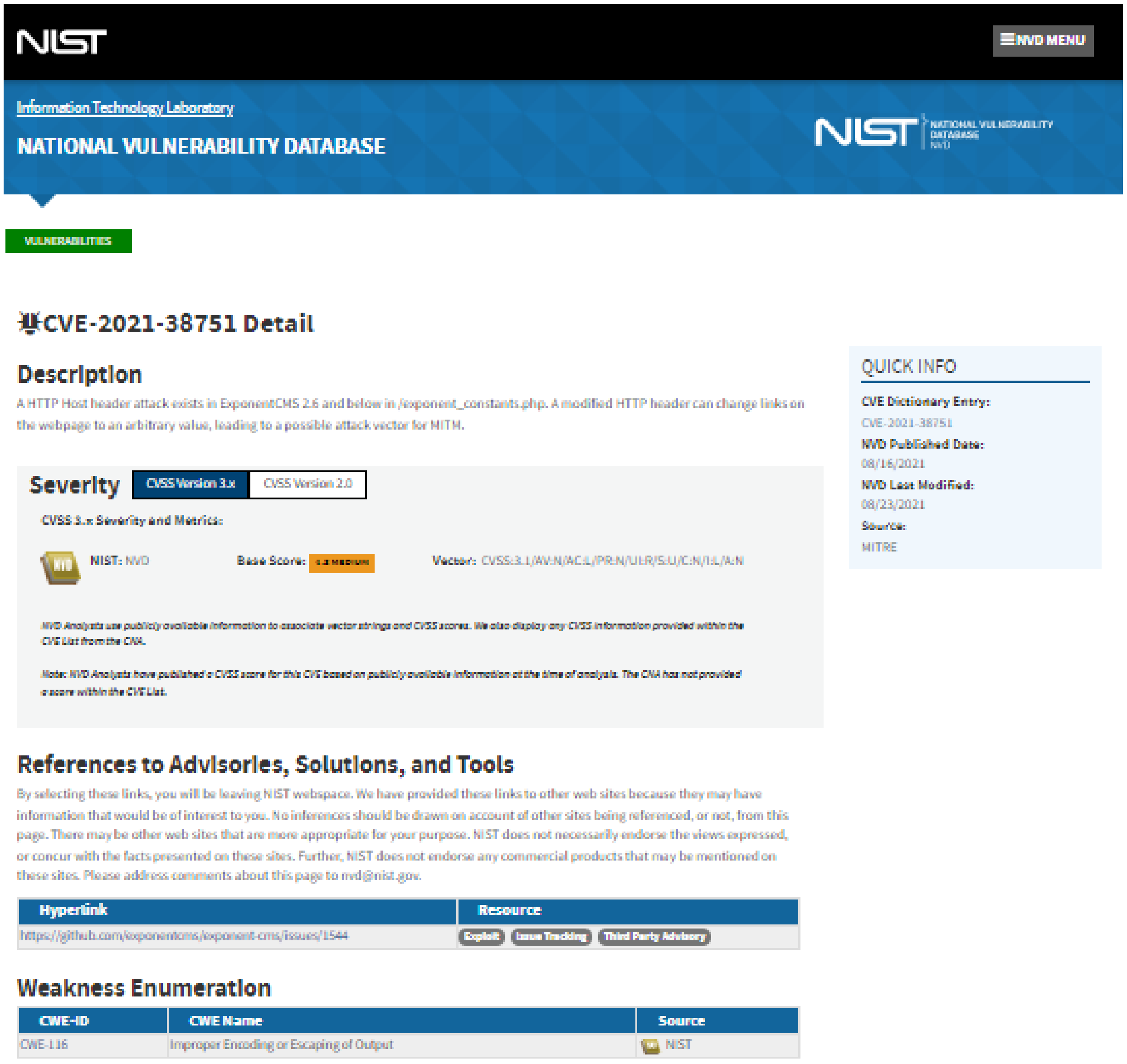
```
GET /.well-known/?nsextt=%0d%0ans%3anetsparker056650%3dvuln HTTP/1.1
Host: paypal.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-us,en;q=0.5
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 10.0; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.353
8.77 Safari/537.36
X-Scanner: Netsparker
```

Proposed mitigation

The following secure coding best practices should be followed by developers to prevent HTTP Header Injection vulnerabilities:

- Validation and sanitization of user input: To make sure that no dangerous characters or sequences are present, every user input used to create HTTP headers must be properly validated and sanitized.
- Proper encoding and escaping: To prevent the interpretation of special characters, all user-controlled input that is utilized to build headers should be encoded or escaped.
- Context-aware output encoding: Developers should employ context-aware output encoding methods when displaying user-supplied data in response headers to make sure that the data is appropriately handled in light of its unique context.
- Security testing: Regular security testing, including as penetration testing and vulnerability assessments, can help find and address potential HTTP Header Injection problems.

External Reference


NIST
NVD MENU
Information Technology Laboratory
NATIONAL VULNERABILITY DATABASE
NIST NATIONAL VULNERABILITY DATABASE NVD
VULNERABILITIES

CVE-2021-38751 Detail

Description

A HTTP Host header attack exists in ExponentCMS 2.6 and below in /exponent_constants.php. A modified HTTP header can change links on the webpage to an arbitrary value, leading to a possible attack vector for MITM.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD Base Score: [illegible] MEDIUM Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:U/C:N/I:L/A:N

NVD Analysts use publicly available information to associate vector strings and CVSS scores. We also display any CVSS information provided within the CVE List from the CNA.

Note: NVD Analysts have published a CVSS score for this CVE based on publicly available information at the time of analysis. The CNA has not provided a score within the CVE List.

References to Advisories, Solutions, and Tools

By selecting these links, you will be leaving NIST webspace. We have provided these links to other web sites because they may have information that would be of interest to you. No inferences should be drawn on account of other sites being referenced, or not, from this page. There may be other web sites that are more appropriate for your purpose. NIST does not necessarily endorse the views expressed, or concur with the facts presented on these sites. Further, NIST does not endorse any commercial products that may be mentioned on these sites. Please address comments about this page to nvd@nist.gov.

Hyperlink	Resource
https://github.com/exponentcms/exponent-cms/issues/1544	Exploit Issue Tracking Third Party Advisory

Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-116	Improper Encoding or Escaping of Output	NIST

QUICK INFO

CVE Dictionary Entry:
CVE-2021-38751
NVD Published Date:
08/16/2021
NVD Last Modified:
08/23/2021
Source:
MITRE