Sri Lanka Institute of Information Technology

IE2062

Web Security

Bug Bounty Report VI

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 Canada Goose Inc. (https://canadagoose.com/).

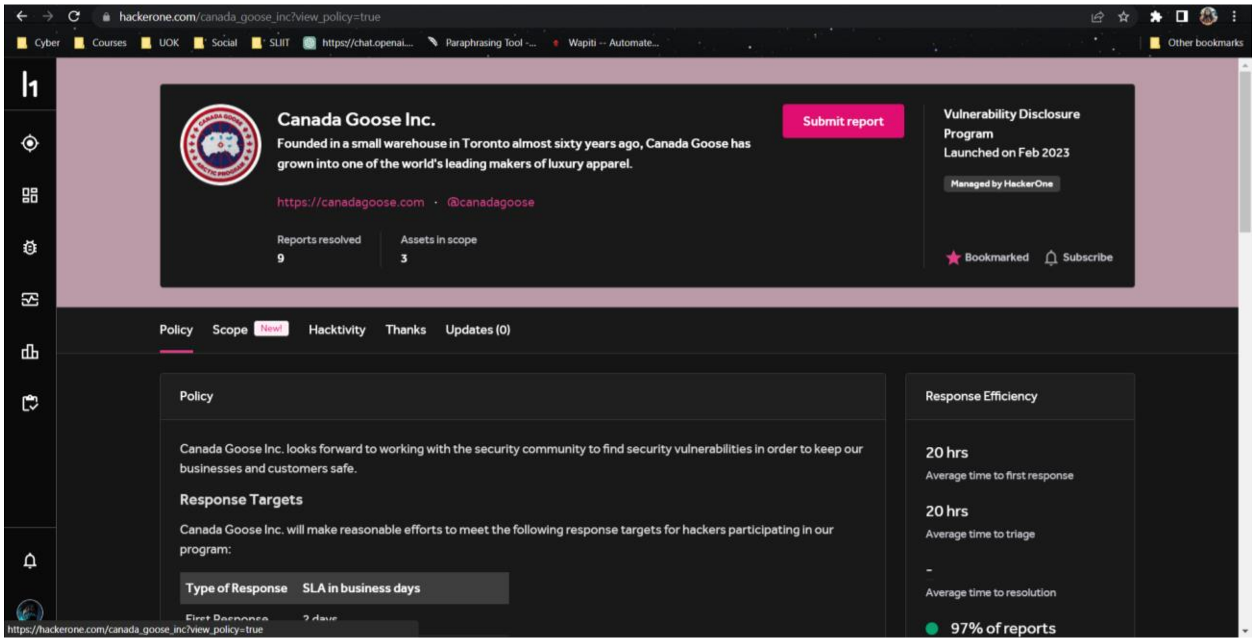
This report uses different tools to gather information detect vulnerabilities and perform penetration testing. The tool name Netsparker and Owasp Zap was mainly used to identify the vulnerability. Further this report provides the vulnerability title, vulnerability description, Affected Components, Impact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

<u>Contents</u>

Introduction

Affected URL: https://canadagoose.com/


hackerone.com/canada_goose_inc?view_policy=true

Canada Goose Inc.

Founded in a small warehouse in Toronto almost sixty years ago, Canada Goose has grown into one of the world's leading makers of luxury apparel.

https://canadagoose.com · @canadagoose

Reports resolved 9 | Assets in scope 3

Submit report

Vulnerability Disclosure Program

Launched on Feb 2023

Managed by HackerOne

Bookmarked | Subscribe

Policy | Scope New! | Hacktivity | Thanks | Updates (0)

Policy

Canada Goose Inc. looks forward to working with the security community to find security vulnerabilities in order to keep our businesses and customers safe.

Response Targets

Canada Goose Inc. will make reasonable efforts to meet the following response targets for hackers participating in our program:

Type of Response	SLA in business days
First Response	2 days

Response Efficiency

20 hrs
Average time to first response

20 hrs
Average time to triage

-
Average time to resolution

97% of reports

https://hackerone.com/canada_goose_inc?view_policy=true

Vulnerability title

Cross-site Scripting

Severity Level:

Risk Level:
HIGH

Vulnerability description

A form of online application vulnerability known as cross-site scripting (XSS) enables an attacker to insert harmful code into reputable websites that other users are seeing. It occurs when a web application incorporates user-provided input into dynamically generated web pages without adequately validating or sanitizing the input.

The attack primarily entails the injection of malicious code, typically JavaScript, into user-interactive elements like input fields, comment sections, and other user-controlled regions of a website. Other users' browsers execute the injected script when they access the harmed webpage, unintentionally running the attacker's code on the secure website. This may result in unlawful actions and a variety of security issues.

Three basic categories of XSS attacks exist:

1. Stored XSS: The server hosting the hacked website keeps the malicious script that was accidentally injected there. Other users may unintentionally run the script when they visit the impacted page, which might compromise their sessions, steal confidential data, or lead them to dangerous websites.
2. Reflected XSS: In this scenario, the website reflects back to the user any injected script that was placed in a URL or input field. The browser runs the script within the context of the trusted website when the user clicks on a malicious link containing the script or submits a form with the script as input, which could result in assaults.
3. DOM-based XSS: This kind of XSS takes place when the JavaScript code of the susceptible website modifies the Document Object Model (DOM) based on user-supplied data without the required sanitization. When the victim's browser interacts with the modified DOM, the attacker injects malicious scripts that are then executed, having a variety of negative effects.

Affected components

Different parts of a web application can be impacted by cross-site scripting (XSS) vulnerabilities. The following are some of the often impacted parts:

- Web forms: Attackers frequently target input fields, such as login forms, registration forms, comment sections, or any other areas where users can submit data, to insert harmful scripts.
- Search Fields: Fields for entering search terms in a search engine are susceptible to XSS attacks if the input is not correctly verified and sanitized before being displayed.
- URL Parameters: If the application does not correctly sanitize or validate the input, URLs including parameters given to the server, such as query strings or URL fragments, may be exploited.
- Cookies: XSS flaws can make it possible for attackers to change or steal cookies, which are little data files saved on a user's browser. Malicious scripts have the ability to access and communicate private data stored in cookies, possibly jeopardizing user sessions.
- Web application APIs: If the input parameters are not adequately checked, XSS attacks may target APIs that enable data exchange between various web apps or client-side scripts.
- User-Generated Content: Any area of a website where users can submit content, including forums, blogs, or message systems, may be vulnerable to XSS if the supplied content is not adequately filtered or sanitized before being shown to other users.
- Error Messages: If handled incorrectly, error messages or debug information presented by a web application may unintentionally reveal sensitive data or open a door for script injection.

Impact assessment

The effects of a Cross-Site Scripting (XSS) vulnerability can differ based on the vulnerability's characteristics, the attacker's objectives, and the particular circumstances surrounding the online application that is being attacked. The following are some possible effects linked to XSS vulnerabilities:

- Session Hijacking: Attackers may take advantage of XSS flaws to hijack user sessions and other authentication tokens. The attacker can assume the identity of the victim, gain illegal access to their account, and take action on their behalf by collecting the victim's session information.
- Data Theft: Using XSS, sensitive data from users or the targeted online application can be extracted. This includes any sensitive data that users enter into the impacted online application, such as personally identifiable information (PII), financial information, login passwords, or other sensitive data.
- Account Compromise: By using XSS, attackers can deceive users into performing malicious actions unintentionally, like changing passwords, altering account settings, or carrying out unwanted transactions. This may result in user accounts being compromised and a loss of control over sensitive personal or financial data.
- Defacement and Content Manipulation: XSS can be used to alter the content of web pages, including swapping out appropriate or harmful content for valid information. This may harm the website's reputation, perplex users, or disseminate false information.
- Malware Distribution: XSS flaws can be used by attackers as entry points to insert malicious scripts or links that, when clicked by unaware users, result in the download or installation of malware. Users' devices may become vulnerable to new attacks, have their security compromised, or be used to recruit botnet members as a result.
- Phishing Attacks: By injecting malicious scripts that seem like genuine login forms or ask users for personal information, attackers can utilize XSS to set up convincing phishing situations. Users may unwittingly give their login information or personal information to malevolent actors as a result of this.
- Website Defacement and Denial of Service (DoS): In some circumstances, XSS can be used to alter or deface the appearance of the entire website, impacting user confidence and harming the reputation of the website owner. Additionally, by inserting resource-intensive scripts that overwhelm the targeted server or impair the availability of the website, attackers might use XSS vulnerabilities to perform DoS attacks.

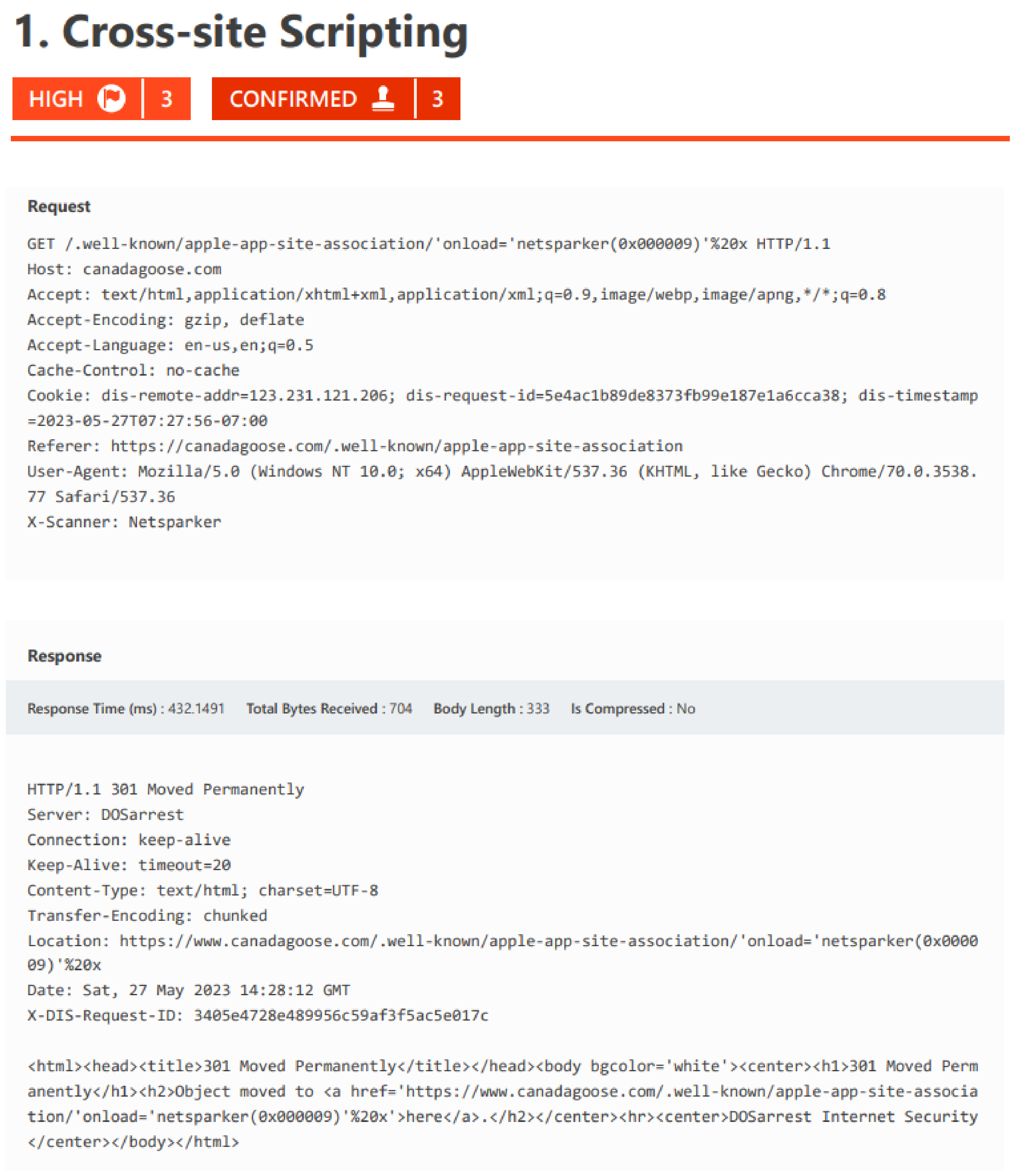
- Cross-Site Script Inclusion (XSSI): In some circumstances, XSS flaws can make it easier for malicious external scripts from third-party domains to be included. This may result in increased exploitation, data loss, or the execution of new attacks.

Steps to reproduce

Recognize the Fundamentals: Learn about and become familiar with the ideas behind the many kinds of XSS vulnerabilities, including stored XSS, reflected XSS, and DOM-based XSS. Find out how web apps handle and present user input.

- Understanding the basics: Reading trustworthy sites, security blogs, and security guidelines that include in-depth details regarding XSS vulnerabilities, actual examples, and best practices for prevention and mitigation are a good place to start your research.
- Learn about payloads: Examine several XSS payloads, including straightforward script injections, HTML elements, event handlers, and multiple encoding methods. Learn how to inject and execute these payloads into a web application.
- Establish a Testing Environment: Set up a secure, monitored environment for security testing. To operate vulnerable web applications, this can entail setting up a local server or employing virtual machines.
- Use Security Tools: Get to know security tools like Burp Suite, OWASP ZAP, or XSS testing frameworks that are built for web vulnerability testing. These tools can help with controlled identification and testing of XSS vulnerabilities.
- Responsible Disclosure: Adhere to responsible disclosure guidelines if you find a possible XSS vulnerability in a web application. Notify the website's owner or the appropriate security teams, giving them thorough details on the vulnerability, its effects, and viable countermeasures.

Proof of concept



1. Cross-site Scripting

HIGH | 3   CONFIRMED | 3

Request

```
GET /.well-known/apple-app-site-association/'onload='netsparker(0x000009)'%20x HTTP/1.1
Host: canadagoose.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: dis-remote-addr=123.231.121.206; dis-request-id=5e4ac1b89de8373fb99e187e1a6cca38; dis-timestamp
=2023-05-27T07:27:56-07:00
Referer: https://canadagoose.com/.well-known/apple-app-site-association
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) : 432.1491   **Total Bytes Received** : 704   **Body Length** : 333   **Is Compressed** : No

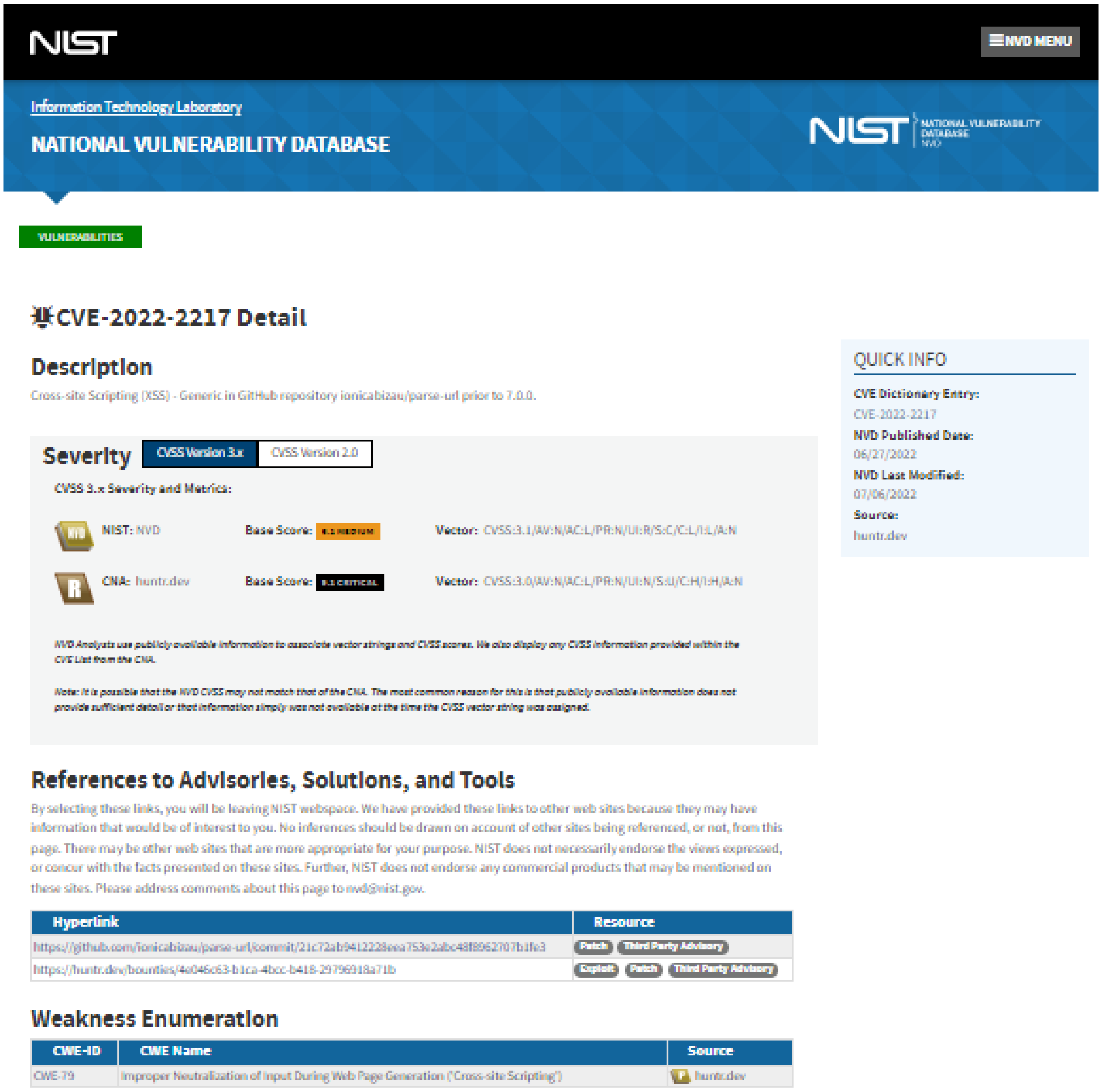
```
HTTP/1.1 301 Moved Permanently
Server: DOSarrest
Connection: keep-alive
Keep-Alive: timeout=20
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Location: https://www.canadagoose.com/.well-known/apple-app-site-association/'onload='netsparker(0x0000
09)'%20x
Date: Sat, 27 May 2023 14:28:12 GMT
X-DIS-Request-ID: 3405e4728e489956c59af3f5ac5e017c

<html><head><title>301 Moved Permanently</title></head><body bgcolor='white'><center><h1>301 Moved Perm
anently</h1><h2>Object moved to <a href='https://www.canadagoose.com/.well-known/apple-app-site-associa
tion/'onload='netsparker(0x000009)'%20x'>here</a>.</h2></center><hr><center>DOSarrest Internet Security
</center></body></html>
```

Proposed mitigation

At various stages of web application development and deployment, a number of security measures must be implemented in order to mitigate Cross-Site Scripting (XSS) vulnerabilities. The following are some suggested mitigation measures:

- Input Validation: Implement stringent input validation on both the client and server sides. Data sanitization. Ensure that before being processed or shown, all user-supplied input has been checked, sanitized, and properly encoded. Utilize whitelisting techniques to filter out or encrypt any potentially malicious characters or scripts, allowing only trusted and expected input.
- Output Encoding: Apply the appropriate output encoding to all dynamic or user-generated content presented in web pages. By translating special characters into their encoded forms, JavaScript escaping, HTML entity encoding, and URL encoding can all assist avoid script injection.
- Implement a content security policy (CSP): Outlines which content sources—such as scripts, stylesheets, or images—are permitted to be loaded and used within a web page. CSP lessens the effects of XSS attacks and aids in limiting the execution of inserted scripts.
- Cookies with the HTTP-only flag set: To block client-side script access, set the HTTP-only flag on session cookies. This minimizes the possibility of session hijacking via XSS by ensuring that hostile scripts cannot access cookies.
- Context-Specific Escaping: Depending on where user-generated input is used in the web application, apply context-specific escaping to it. To prevent script injection, different contexts, such HTML, JavaScript, CSS, or URLs, call for different escape approaches.
- Secure Coding Guidelines: Implement secure coding guidelines and encourage secure coding practices within development teams. These practices include avoiding the use of unsafe APIs or methods for processing user input and updating frameworks and libraries on a regular basis to fix known security flaws.
- Regular Security Audits and Penetration Testing: To find and fix XSS vulnerabilities, do routine security audits and penetration tests. This includes evaluating the web application's security posture using manual testing methods and automated scanning technologies.
- Web Application Firewalls: Install a web application firewall (WAF) to help identify and stop malicious requests that could take advantage of XSS flaws. By examining incoming traffic and preventing potentially malicious requests, WAFs can add an extra layer of security.
- Users' Education: Inform users about the dangers of XSS attacks and promote safe browsing habits such avoiding dubious links, exercising caution when inputting personal information, and often updating browsers and plugins.

External Reference


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

CVE-2022-2217 Detail

Description

Cross-site Scripting (XSS) - Generic in GitHub repository ionicabizau/parse-url prior to 7.0.0.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD	Base Score: 6.1 MEDIUM	Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
CNA: huntr.dev	Base Score: 9.1 CRITICAL	Vector: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/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: It is possible that the NVD CVSS may not match that of the CNA. The most common reason for this is that publicly available information does not provide sufficient detail or that information simply was not available at the time the CVSS vector string was assigned.

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/ionicabizau/parse-url/commit/21c72ab9412228eea753e2abc48f8962707b1fe3	Patch Third Party Advisory
https://huntr.dev/bounties/4e046c63-b1ca-4bcc-b418-29796918a71b	Exploit Patch Third Party Advisory

Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	huntr.dev

QUICK INFO

CVE Dictionary Entry:
CVE-2022-2217
NVD Published Date:
06/27/2022
NVD Last Modified:
07/06/2022
Source:
huntr.dev