|  |
| --- |
| vulnerabulity testing |
| Vulnerability Report |
| Tuesday, May 18, 2021 |



modifications history

| **Version** | **Date** | **Author** | **Description** |
| --- | --- | --- | --- |
| 1.0 | 05/18/2021 | SUMANTH tadiboina | Initial Version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. General Information 4

1.1 Scope 4

1.2 Organisation 4

2. Executive Summary 5

3. Technical Details 6

3.1 title 6

4. Vulnerabilities summary 8

# General Information

## Scope

VITAP has mandated us to perform security tests on the following scope:

* to find the vulnerabilities in the tested site

## Organisation

The testing activities were performed between 05/17/2021 and 05/18/2021.

# Executive Summary

# Vulnerabilities summary

Following vulnerabilities have been discovered:

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **ID** | **Vulnerability** | **Affected Scope** |
| High | IDX-002 | DOM XSS |  |
| High | IDX-001 | clcikjacking |  |
| High | IDX-003 | buffer overflow |  |

# Technical Details

## DOM XSS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | High | | CVSSv3 Score | | 8.2 |
| **CVSSv3 criterias** | Attack Vector : | **Network** | Scope : | **Changed** | |
| Attack Complexity : | **Low** | Confidentiality : | **High** | |
| Required Privileges : | **Low** | Integrity : | **Low** | |
| User Interaction : | **Required** | Availability : | **Low** | |
| **Affected scope** |  | | | | |
| **Description** | DOM-based XSS vulnerabilities usually arise when JavaScript takes data from an attacker-controllable source, such as the URL, and passes it to a sink that supports dynamic code execution, such as eval() or innerHTML. This enables attackers to execute malicious JavaScript, which typically allows them to hijack other users' accounts. | | | | |
| **Observation** | in reflective and stored Cross-site scripting attacks you can see the vulnerability payload in the response page but in DOM based cross-site scripting, the HTML source code and response of the attack will be exactly the same, i.e. the payload cannot be found in the response. It can only be observed on runtime or by investigating the DOM of the page. | | | | |
| **Test details** | | | | | |
| **Remediation** |  | | | | |
| **References** |  | | | | |

## clcikjacking

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | High | | CVSSv3 Score | | 8.0 |
| **CVSSv3 criterias** | Attack Vector : | **Network** | Scope : | **Changed** | |
| Attack Complexity : | **High** | Confidentiality : | **High** | |
| Required Privileges : | **Low** | Integrity : | **High** | |
| User Interaction : | **Required** | Availability : | **High** | |
| **Affected scope** |  | | | | |
| **Description** | Clickjacking, also known as a “UI redress attack”, is when an attacker uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page. Thus, the attacker is “hijacking” clicks meant for their page and routing them to another page, most likely owned by another application, domain, or both. | | | | |
| **Observation** | an attacker who builds a web site that has a button on it that says “click here for a free iPod”. However, on top of that web page, the attacker has loaded an iframe with your mail account, and lined up exactly the “delete all messages” button directly on top of the “free iPod” button. The victim tries to click on the “free iPod” button but instead actually clicked on the invisible “delete all messages” button. In essence, the attacker has “hijacked” the user’s click, hence the name “Clickjacking”. | | | | |
| **Test details** | | | | | |
| **Remediation** |  | | | | |
| **References** |  | | | | |

## buffer overflow

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CVSS Severity | High | | CVSSv3 Score | | 7.6 |
| **CVSSv3 criterias** | Attack Vector : | **Local** | Scope : | **Changed** | |
| Attack Complexity : | **High** | Confidentiality : | **High** | |
| Required Privileges : | **None** | Integrity : | **Low** | |
| User Interaction : | **Required** | Availability : | **High** | |
| **Affected scope** |  | | | | |
| **Description** | A buffer overflow, or buffer overrun, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations. It exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer. In this case, a buffer is a sequential section of memory allocated to contain anything from a character string to an array of integers. Writing outside the bounds of a block of allocated memory can corrupt data, crash the program, or cause the execution of malicious code. | | | | |
| **Observation** | We have observed that this buffer overflow can potentially crash an application and unknowingly allows command injection attacks. | | | | |
| **Test details**  image  Image 1 – SUMANTH.png  image  Image 1 – STOPPED.png | | | | | |
| **Remediation** | 1. Address space randomization (ASLR)  2. Data execution prevention (DEP)  3. Structured exception handler overwrite protection (SEHOP) | | | | |
| **References** |  | | | | |