**Sri Lanka Institute of Information Technology**

A blue and orange logo

Description automatically generated

**Web security – IE2062**

B.Sc. (Hons) in Information Technology Specialization in cyber security.

**Bug Bounty Reports**

By: R.M.C.A RATHANAYAKA

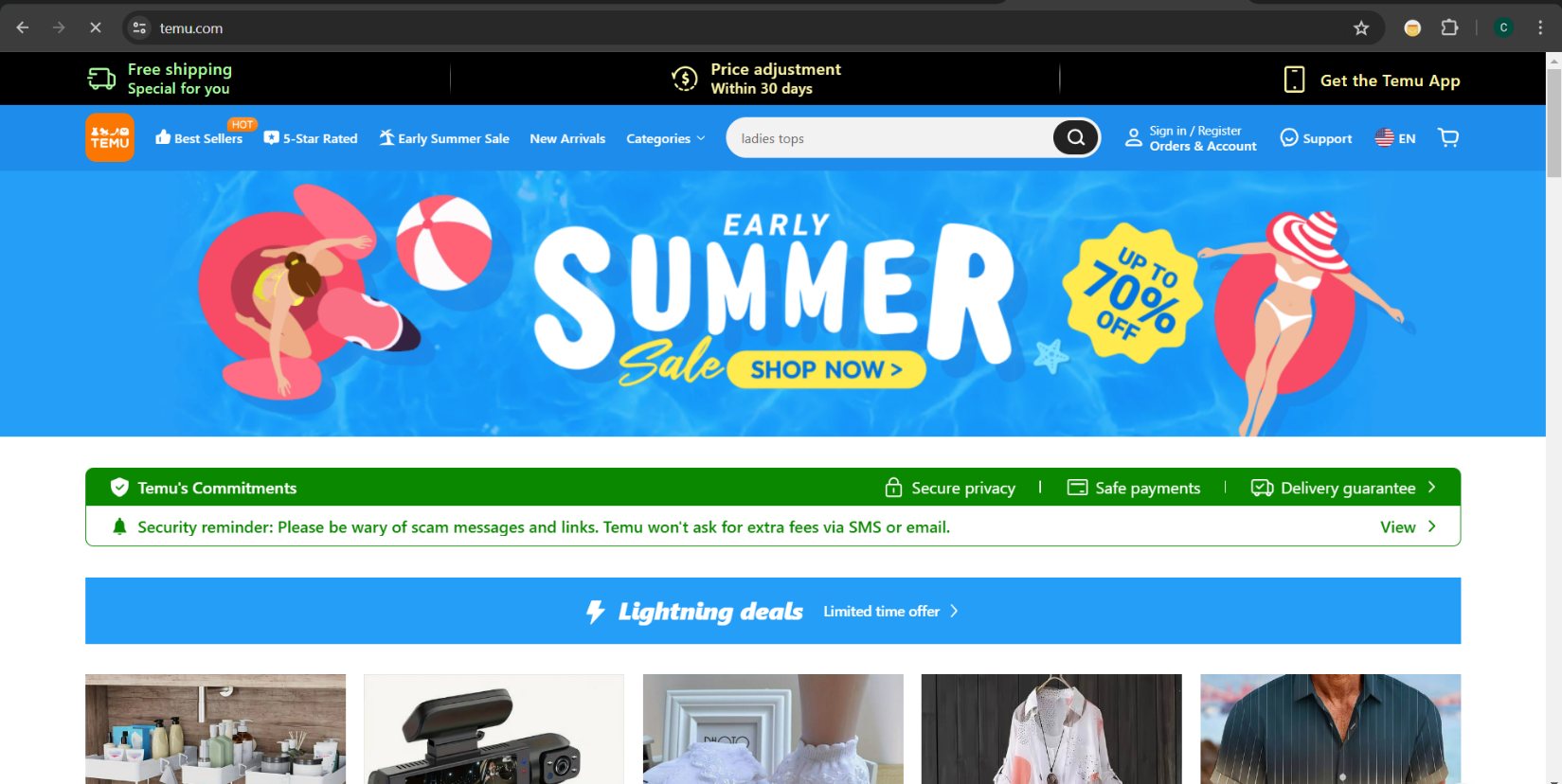
IT22353184

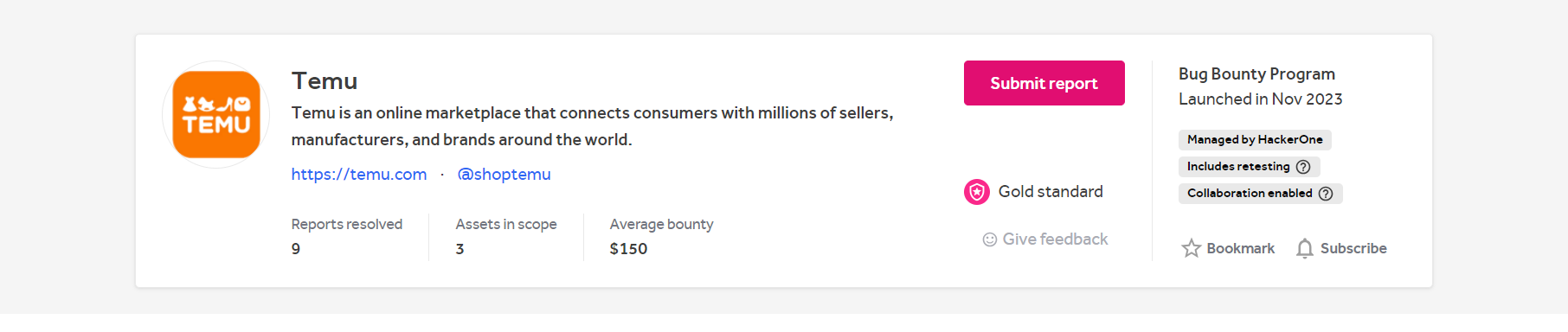
CS Y2 S2 1.

**Report 01)**

**Domain:** [**http://www.temu.com/**](http://www.temu.com/)

Temu is an online marketplace owned by a Chinese commerce company PDD Holdings which offers discounted consumer goods. This platform connects millions of consumers and vendors and manufacturers around the world. Temu is known for data privacy of consumers and quality of its marketplace goods.





**Title: - Insecure Transportation Security Protocol Supported (TLS 1.0)**

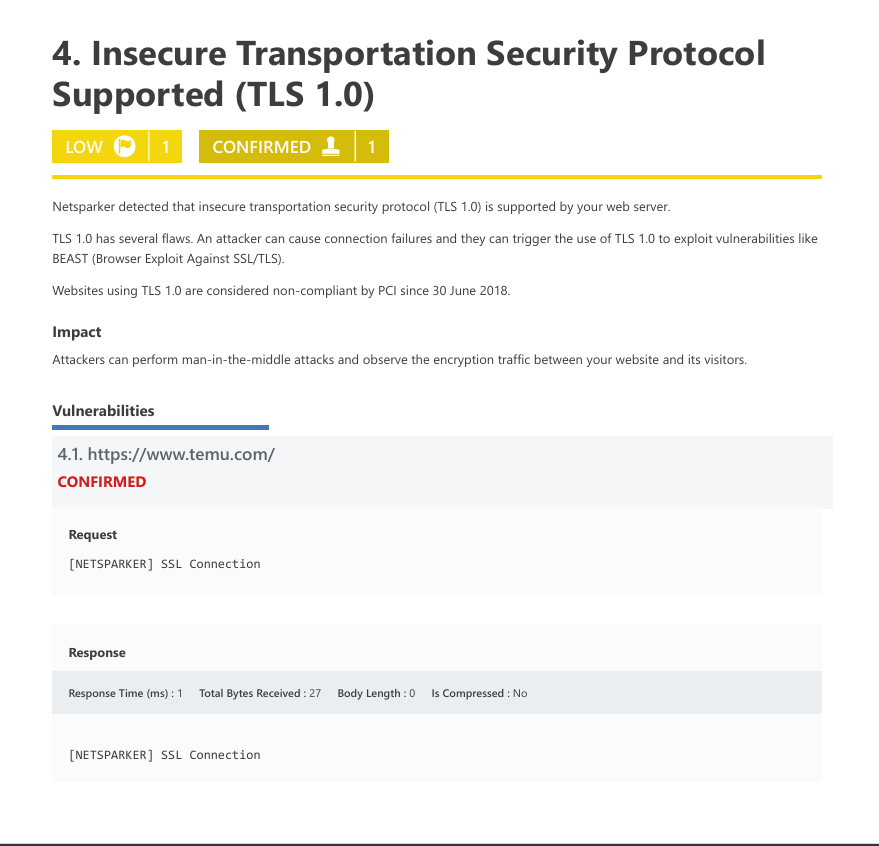
I have found a Insecure Transportation security protocol supported flaw within the vulnerability system. Transport Layer Security 1.0 contains various issues and is widely known to attacks like BEAST. And industry standards have recommended the discontinuation of TLS 1.0 support. Using this an attacker can cause connection failures and they can induce connection failures and trigger the use of TLS1.0 to exploit vulnerabilities. Since June 30,2018, websites that use TLS 1.0 are declared non-compliant by PCI standards.

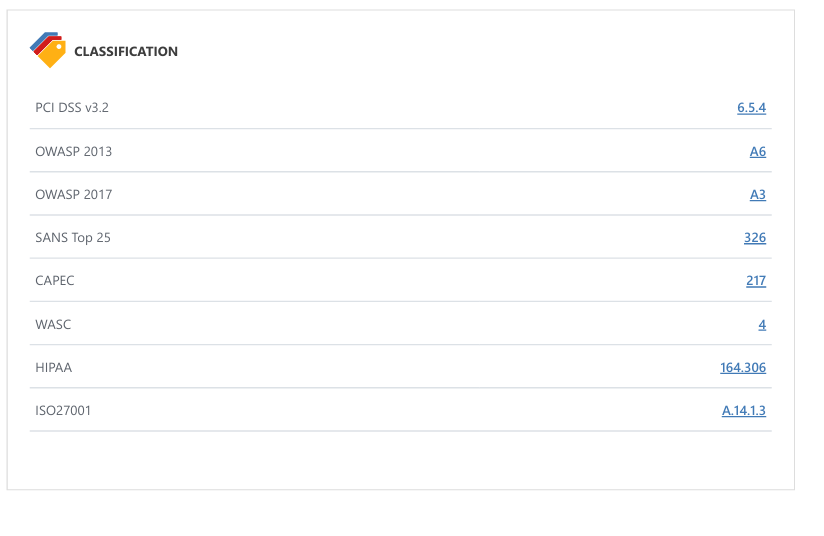
* Severity: Low
* OWASP 2013: A6
* OWASP 2017: A3
* **Effected components:**

1. web server configuration.

2.TSL Implementation

1. **Impact Assessment**
2. Use of support of TLS 1.0 could bring security risk that could get exploited by attackers to intercept an decrypt sensitive data transmitted over network and cause significant risk to web servers and its data processes.
3. In regard of the compliance issues many organizations and regulation like PCI DSS requires the use of more secure protocols than TSL1.0 and use of this protocol in regarding credit card transactions may results in legal and financial repercussions.
4. There is an increased risk of data breaches due to weaker encryption which can lead to interception and leaks of sensitive data. And when consumers are aware about the insecure transmission of their data the affected parties will lose the consumer trust and damage the organizations brand reputation.
5. In the remediation process it can disrupt web server activities because of the required down time to fix this vulnerability but as this vulnerability possess great risk and expanded attack surface, quick mitigations must be done.
6. **Steps to reproduce**
7. First need to determine the URL or IP address of the server that needed to test TSL1.0 support. I did the test on [**http://www.temu.com/**](http://www.temu.com/)domain.
8. Use tools like Nmap to identify presence of the TLS1.0 in the targeted web werver.
9. Then use tools OpenSSL, SSlScan or automated tools like netsparker to do a vulnerability scan. Configure the tools to specially check the TSL1.0 when connecting to the server.
10. If the server accepts the connection and establishes a session and the TSL.10 supports are identified the vulnerability is reproduced.
11. Then confirm that the vulnerability has exploited successfully by intercepting sensitive data transmitted over the TSL1.0 connection.
12. **Proof of concepts**





1. Proposed Mitigations
2. As TSL 1.0 is known to have vulnerabilities, first step to take is configure the web server to support TSL1.2 or higher which can simply done by adjusting SSL/TSl configuration setting on the web server.
3. Configure your web server to disallow using weak ciphers and protocols. You need to restart web server to apply those changes.

* For Apache, adjust the SSLProtocol directive provided by the mod\_ssl module. This directive can be set either at the server level or in a virtual host configuration.

**SSLProtocol +TSLv1.2**

* For Nginx, locate any use of the directive ssl\_protocols in the nginx.conffile and remove TLSv1**.**

**SSLProtocols TLVSv1.2;**

* For Microsoft IIS, you should make some changes on the system registry.

1. Click on Start and then Run, type regedt32or regedit, and then click OK.

2. In Registry Editor, locate the following registry key or create if it does not exist:

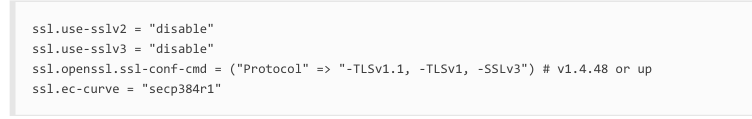
H**KEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\**

3. Locate a key named Serveror create if it doesn't exist.

4. Under the Serverkey, locate a DWORD value named Enabledor create if it doesn't exist and set its

value to "0".

* For lighttpd, put the following lines in your configuration file:

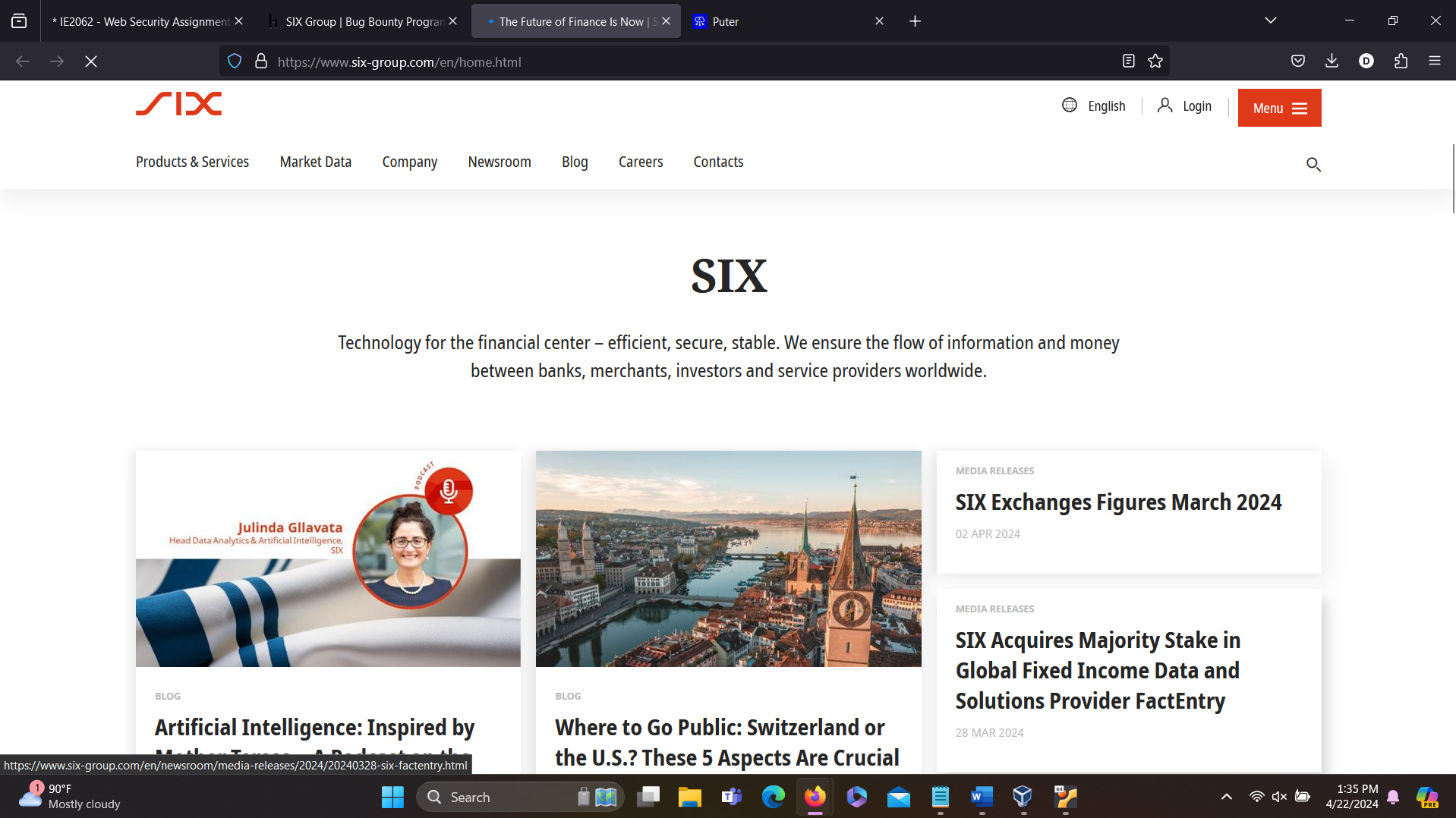


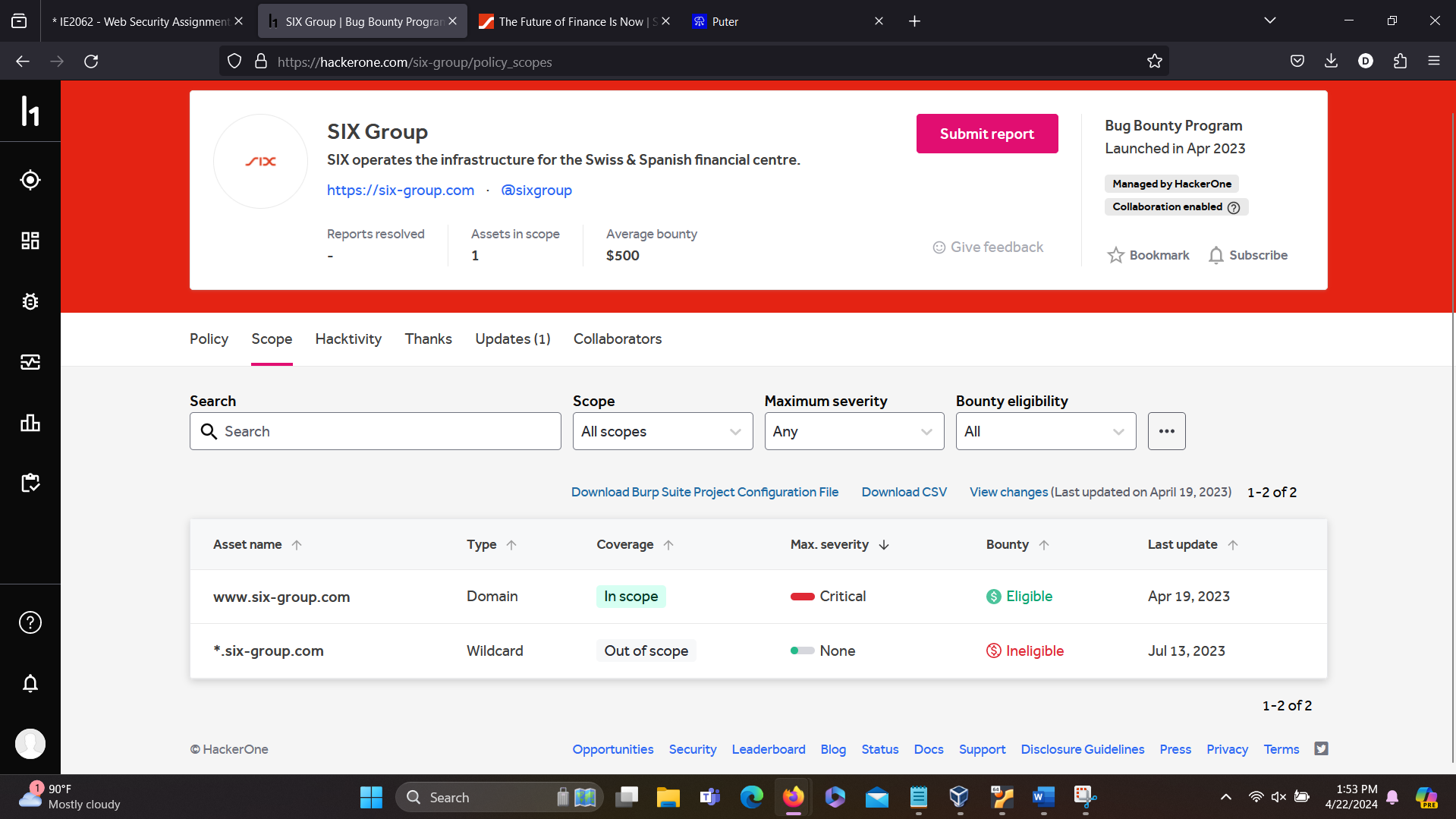
1. To identify potential security issues do regular vulnerability scans and audits ensure the server remains secure and be up to date on SSL/TSL protocols and standards.

**Report 2)**

**Domain:** [**http://www.six-group.com/**](http://www.six-group.com/)

SIX is a service provider of financial services and technology solutions for the global financial market. The SIX operates the infrastructure for financial centers in Switzerland and Spain and provides a range of services in securities, financial information, and payment sectors.

****



**Title: -** **HTTP Strict Transport Security (HSTS) Errors**

**and Warnings**

HTTP Strict Transport(HSTS) is a web security tool that helps and protect websites from cookie hijacking and protocol downgrade attacks. It enables a web server to declare that the only means to access it are over secure connection, such HTTPS .This ensures that the website is safe from vulnerabilities and platforms for vulnerabilities.

The error I found on this Domain is that “preload” directive is missing from the HSTS header configuration of this website. The server must contain a distinct HTTP header in its response to client request for HSTS to work. IN order to add the domain of the website to the built in HSTS preload lists of browser, this directive is essential.This header has a directive in that instruct the web browser to see the website for a certain period of time solely using HTTPS.

And proper usage of HSTS ate essential for ensuring security and reliability of the website it self. This helps to safeguard and prevent cyber attacks during online data transactions.

* Severity: Medium
* OWASP 2013: A5
* OWASP 2017: A6
* **Effected components**:

1. Security integrity which can leads to attacks like SSL Stripping
2. Data Confidentiality which can lead to data interception in insecure HTTP connections.
3. The impact of SEO standpoints which may leads to lower rankings in search engine results pages due to not enforcing HTTPS.

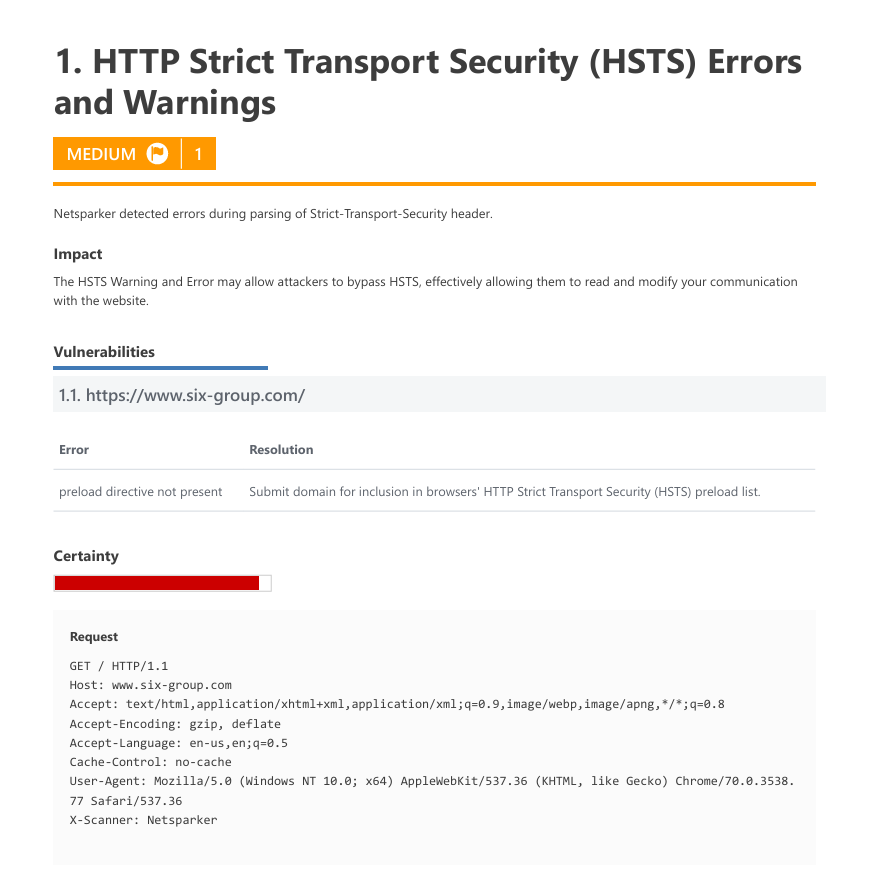
1. **Impact Assessment:**
2. Impact in security: This vulnerability increases the risk of attacks such as man-in-the-middle attack and without proper HSTS configurations confidentiality of sensitive data sent between the user and the server may be at risk. Attacker can jeopardize the integrity and security of the of this website using this error to carry out attacks like SSL Stripping or downgrade assaults.
3. Due to HSTS issues, the website might not be able to create private links compatible browsers. And from that user can have issues regarding that link may not be secure.
4. User trust: The websites security measures and lacking proper use of HTTPS enforcement can lead to decreased trust and negative user experience, and gradually discouraging users from interacting with the website.
5. Legal violations: failure to implement proper security measures including HTTPS and HSTS with preload can lead to legal violations and regulatory infractions. Also may bring harm to the website’s reputation causing financial fines.

**b)Steps to Reproduce**

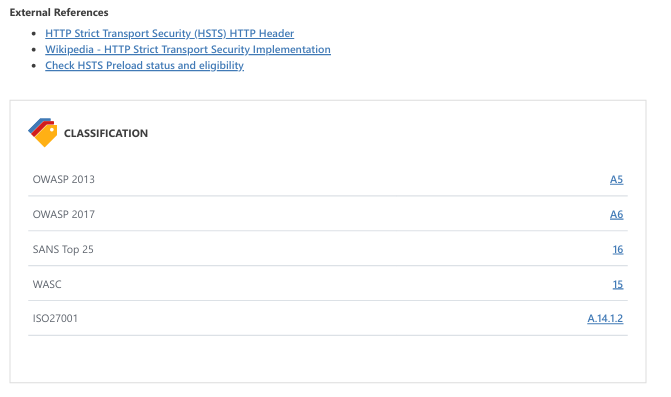
1. Verify if the HSTS(HTTP Strict Transport Security) is enabled on the website by checking the answer messages on the web site and check the HTTP response for the presence of the HSTS header.
2. Clear the HSTS cache in browser because if you have viewed the website before the browser may have saved the HSTS policy on the computer. To test the website clear HSTS cache on the computer to start over.
3. Make an attempt to access the website using HTTP instead of HTTPS. To do that enter the URL using http:// instead of using https://.
4. Observe the error or the warning. Based on your browser and HSTS implementation Error message and Warning message may differ. Usually browser may show a error page that you cant access the website HTTPS or a warning to use HTTPS instead of HTTP or in some cases browser will automatically redirect the HTTP request to HTTPS.
5. To identify vulnerabilities caused by not having a preload directive must run a security test which can exploited using network sniffing, man-in-the-middle attacks and SSL stripping attacks.

c**)proof of concept**

**Scanned report by Net sparker**

****





**d)Proposed Mitigations**

1. First need to find the errors and issues and fix those problems. After the issues with the HSTS header has been fixed the website must be added to the HSTS preload list.

Browsers instantly connect to the website via HTTPS when the name is added to preload list, stopping users from making HTTPS calls to the server. You have to make sure that the website sends HSTS header with appropriate directives.

Example HSTS header.

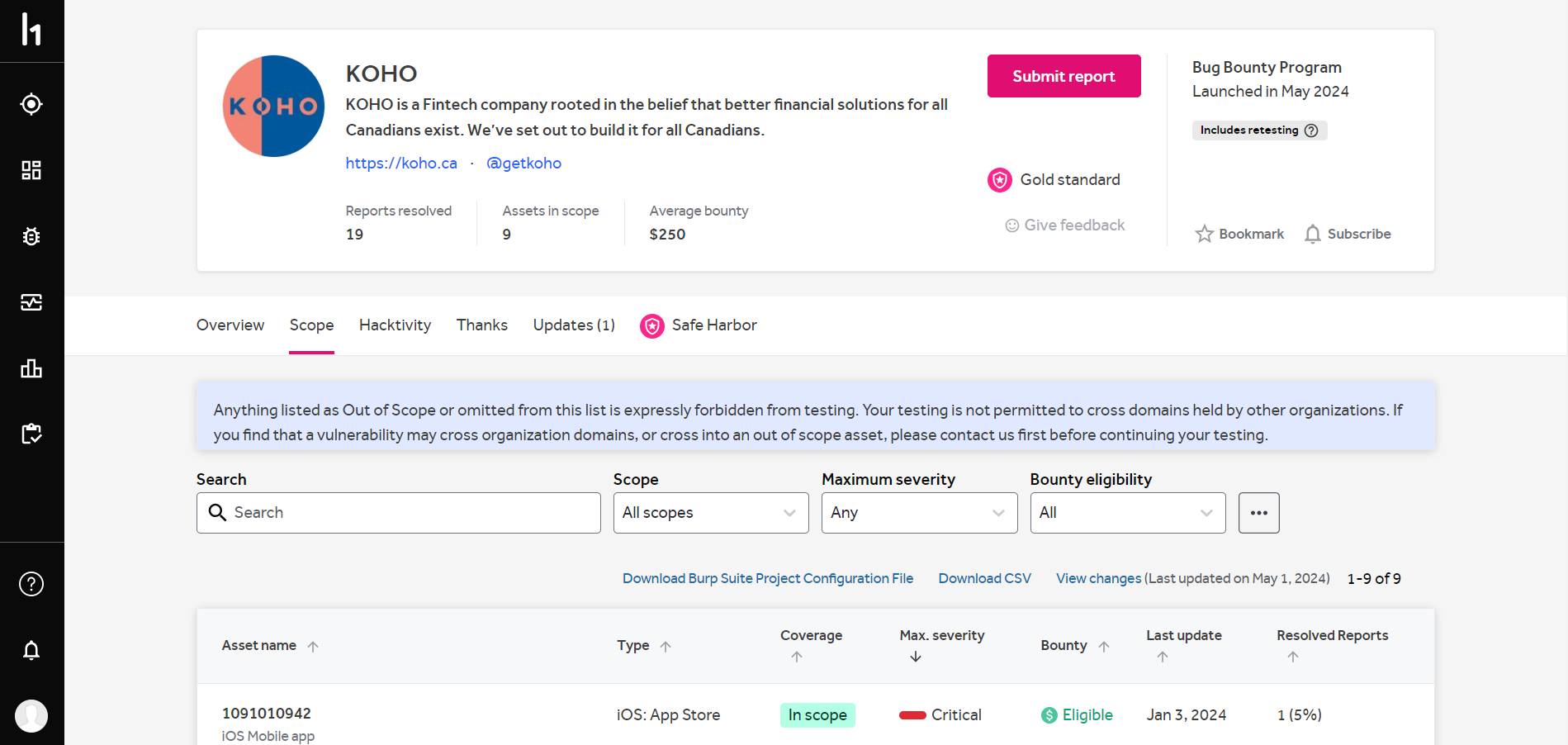
**Strict-Transport-Security: max-age=31536000; includeSubDomains; preload**

1. Also make sure SSL/TLS Is configured correctly. And make sure you get a certificate issued by a reputable certificate authority and it is installed and set up correctly into you web site.
2. Configure your server to redirect all HTTP requests to HTTPS and make sure your website has a secure connection when accessing your site.

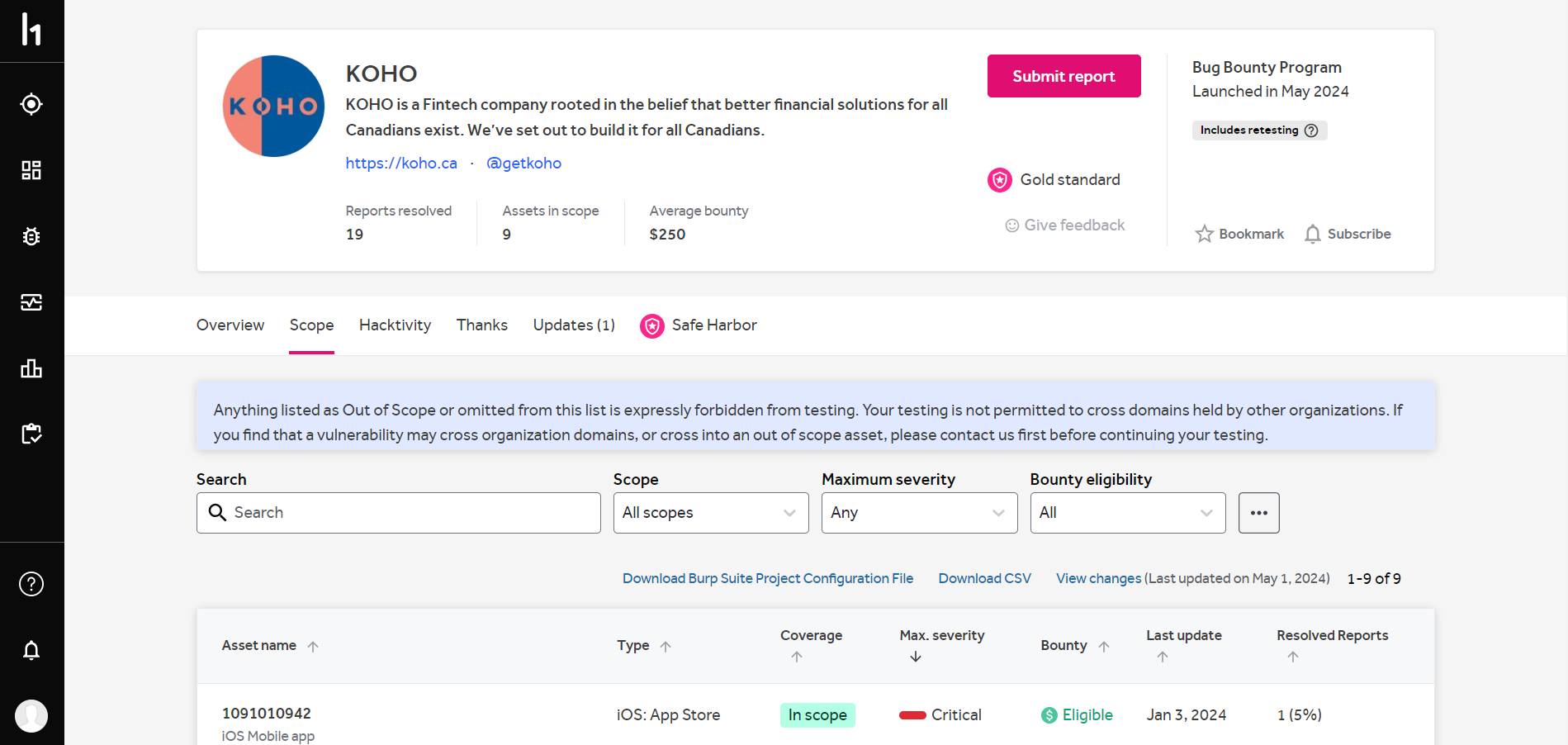
**Report 03)**

**Domain: -** **<https://koho.ca/>**

KOHO is a Canadian Fintech company based in Toronto. With KOHO you can save money and get instant cash back while earning interest on you account’s full balance. The account is free of interest and KOHO is not a bank. And you receive a prepaid Mastercard. And this website have an app which helps you manage your own money by giving a real-time update about your account activities.



And after inspecting the website I found Cross-site Scripting vulnerability .

****

**Title: - Cross-site Scripting (XXS)**

Cross-site Scripting is a commonly found vulnerability in web applications. This happens when an attacker inserts dangerous code, usually made of javaScript, into websites which are viewed by other users. These malicious codes may be injected through HTTP headers, URL parameters and input fields. This allows attackers several attack opportunities, mostly session hijacking which is done by changing current session of the user or changing the look of the page by changing the HTML on the fly to steal user’s credentials. Mainly the named attacks are steal cookies, session Hijacking, phishing attacks, defacement and keylogging.

Severity: Medium

OWASP: 2013 A3

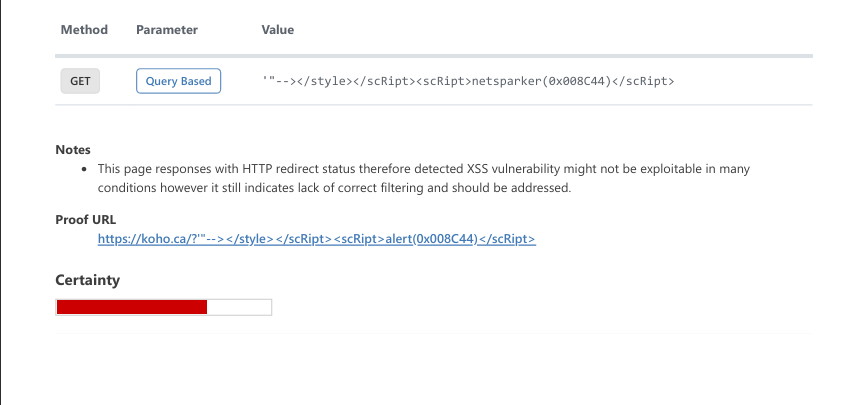
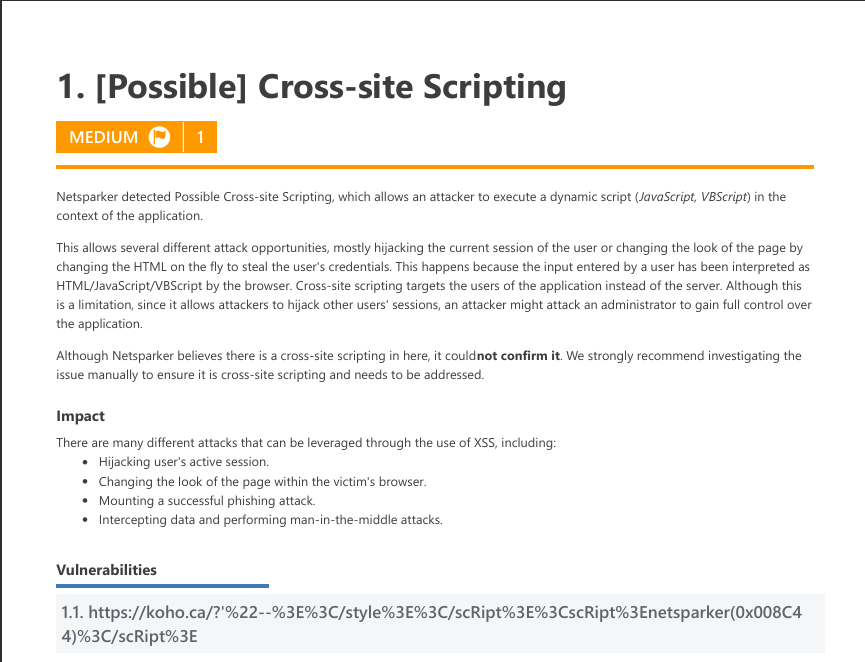
OWASP: 2017 A7

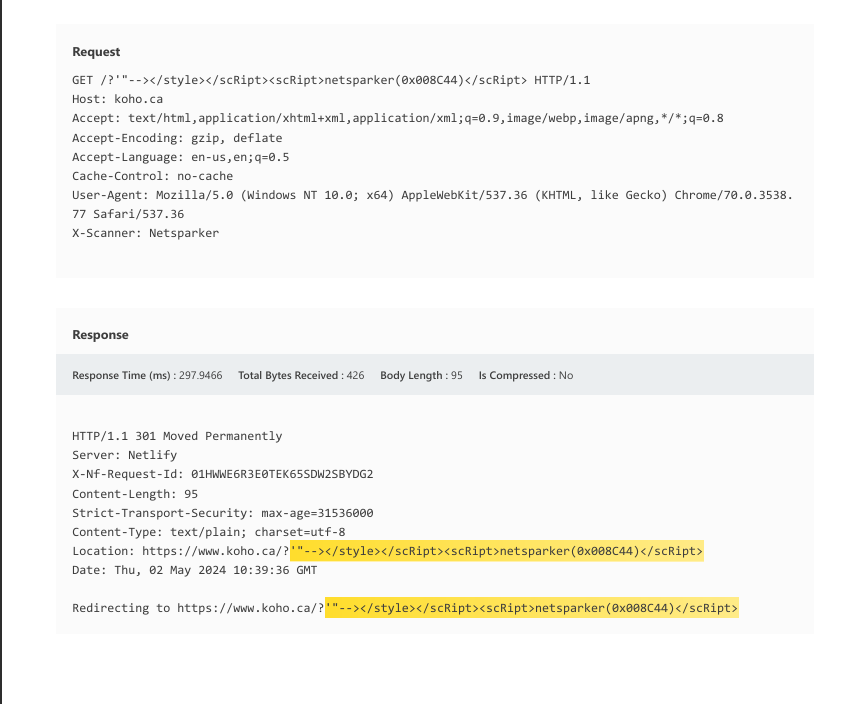
**Effected Components**:

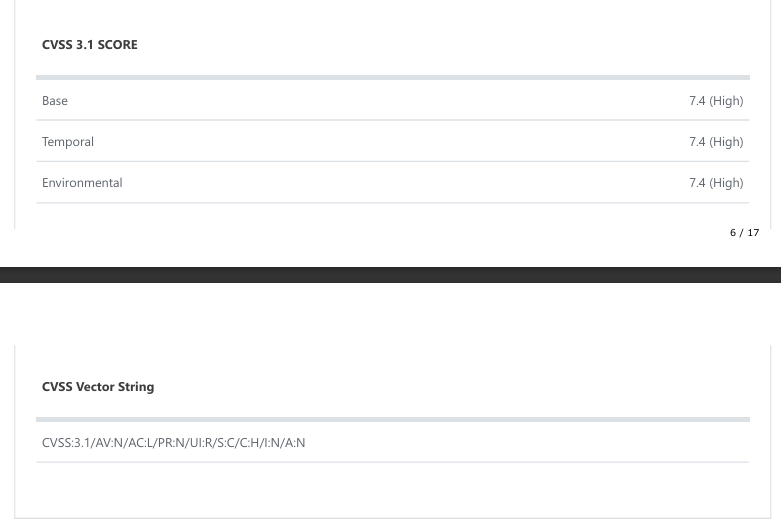
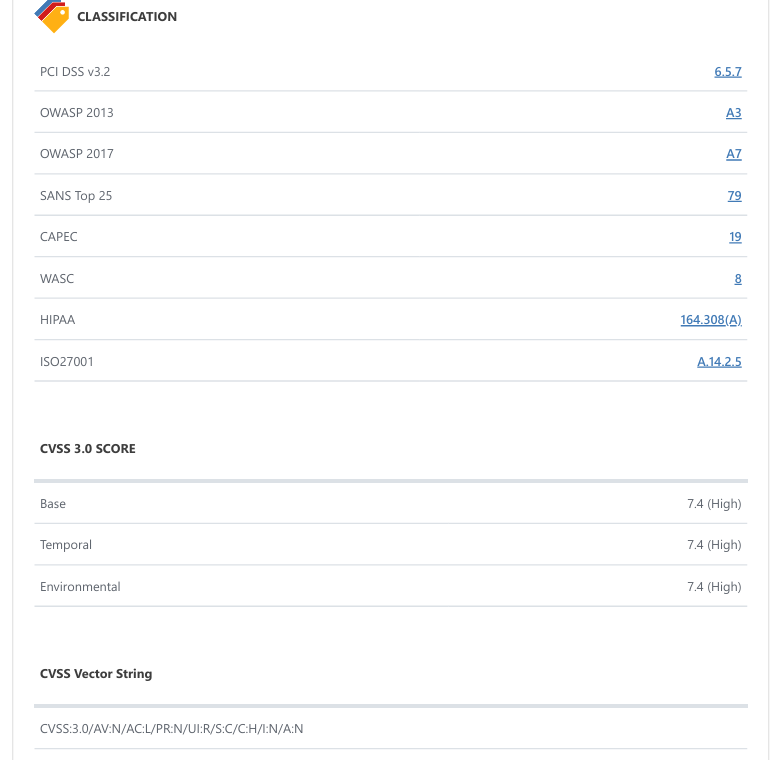
* Any input field that a user can input data.
* Parameters that are passed in URL.
* HTTP headers
* Session cookies
* Client-side Scripts

**a)Impact Assessment:**

1. Session hijacking involves attackers taking over a user’s active session of the hacked website, assuming the identity of the authentic user and accessing their accounts without authorization and taking actions on their behalf. This compromises the authentic user’s account’s availability, confidentiality and integrity.
2. Credential theft involves where an attacker injects malicious code into the website, causing the website to dynamically change the appearance allowing attackers to use user credentials for illegal activities and identity theft.
3. Using XXS vulnerability attackers can modify the content on the website and inject malicious scripts to mislead the users and damaging reputation of the website through Data Manipulation.
4. Also, attackers can execute client side attacks which gives the ability to run any scripting code within the client’s browser which can lead to phishing attempts, downloading certain malwares by taking advantage of the user’s browser vulnerabilities.
5. These types of attacks mainly leads to affected applications reputational damage making it loose users and which leads to financial loses and also penalties for not having secured methods to enforce the website.
6. **Proof of concept:**





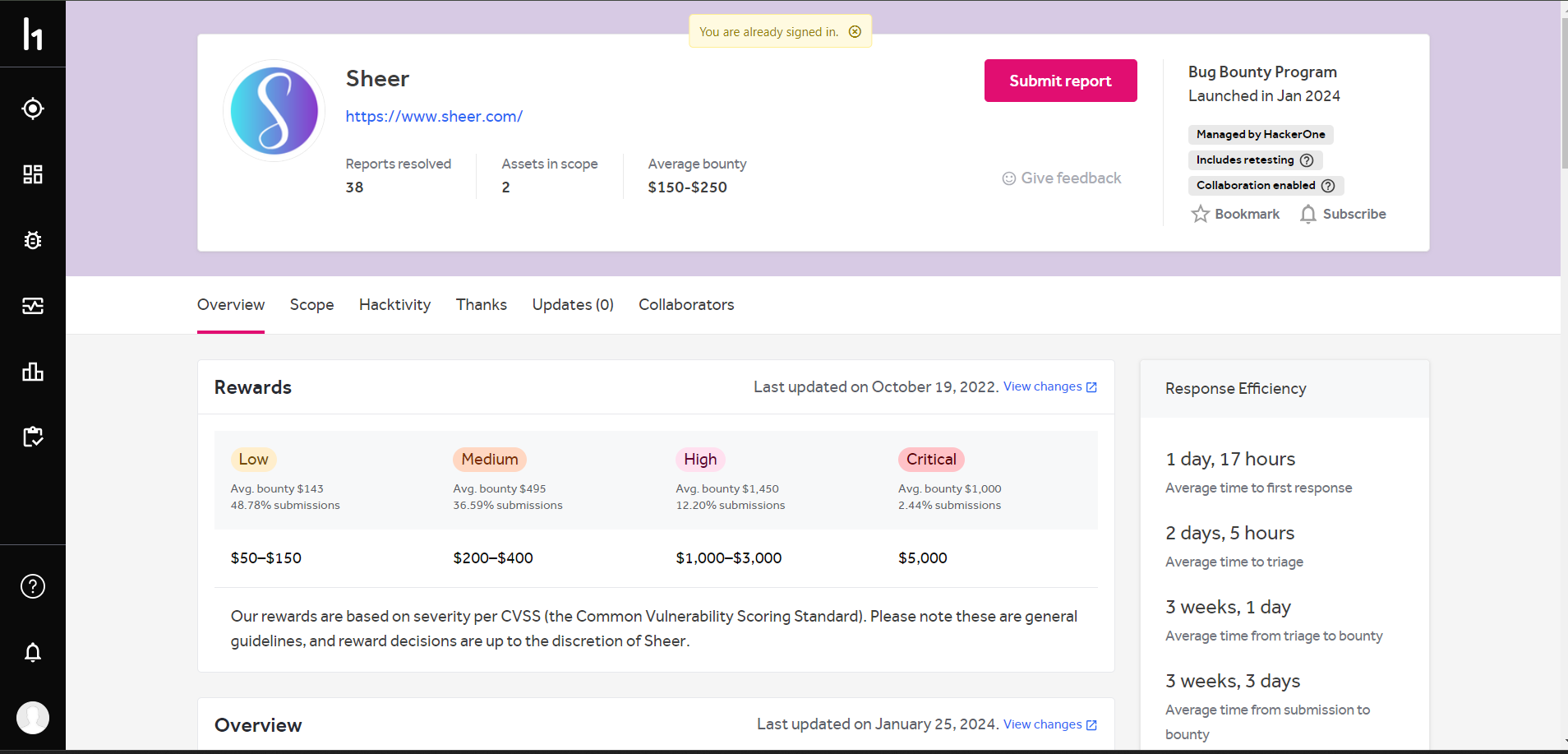


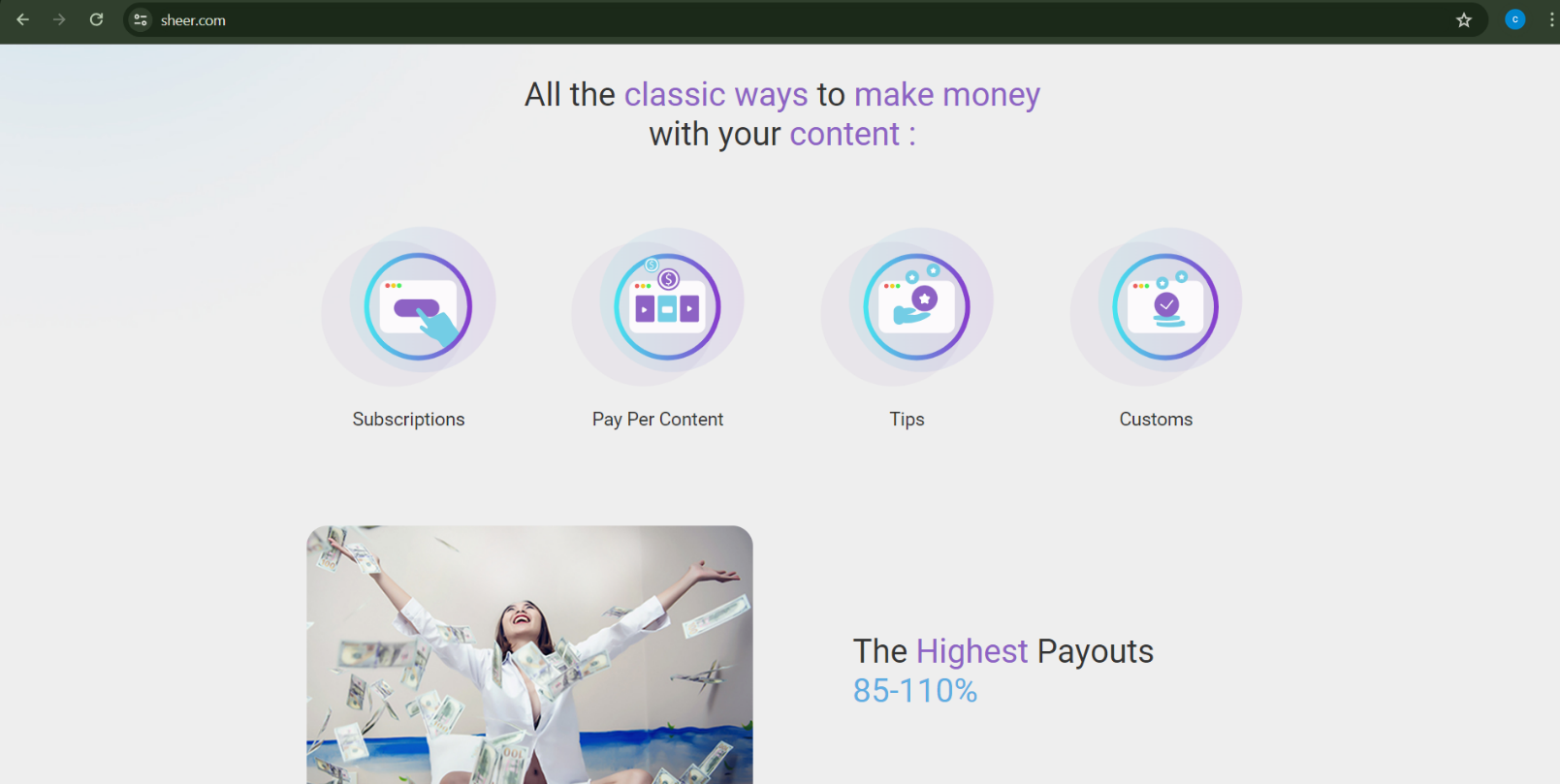
**c)Steps to Reproduce:**

1. First, pinpoint the areas in the web application such as input fields or form submission areas, where user input is accepted and shown back to users.
2. Attempt input validation bypass: see whether you can enter special characters, HTML tags, or java Script code by trying to go around any validation systems, whether they are implemented client-side or server side.
3. Fill up the input fields with malicious scripts, most often java script. Simple alert messages to intricate programs intended to steal cookies or carry out other malicious deed might fall under this category.
4. Then provide the application with the input that has injected script in it and check if the user receives a reflection of the injected script without any issues.
5. To explore further vulnerabilities try injecting malicious scripts into different contexts like HTTP header or URL parameters.
6. **proposed mitigations:**
7. Before utilizing any user input within an application, it is imperative to carefully review and purify it by input validation and sanitization. This procedure entails checking that the input complies with established guidelines and eliminating any potentially hazardous information. We can stop dangerous programs and HTML markup from running by doing this.
8. Encoding dynamic content correctly is necessary before it can be shown on webpages. To prevent special characters from being interpreted by browsers as executable code, encoding transforms them into the appropriate HTML entities.
9. Malicious scripts in incoming HTTP requests or answers can be found and blocked by using web application firewalls (WAFs) or specific XSS filtering techniques. Real-time web traffic analysis is performed by these filters.

**Report 04)**

**Domain: -** [**http://www.sheer.com/**](http://www.sheer.com/)

Sheer is a New York city based media platform. Using this platform, you can easily manage an control your campaigns. Through this website, Offers are available, campaign planning is simple and safe, and funds are processed quickly. Our platform will handle the technical aspects, allowing you to concentrate on producing exceptional content and working with brands.



**Title: -** **Out-of-date Version (jQuery)**

The vulnerability known as "Out-of-date Version (jQuery)" concerns the use of an outdated and possibly unsafe version of the jQuery library in a web application. One popular JavaScript library that is used for DOM manipulation, event handling, and AJAX requests is called jQuery.

Using out-of-date versions of jQuery might cause developers to inadvertently add security flaws to their apps. The aforementioned vulnerabilities may include well-known problems like cross-site scripting (XSS), cross-site request forgery (CSRF), or additional client-side attack techniques. Here, this vulnerability affects websites by creating possible surfaces to attacks like cross-site Scripting and prototype pollution.

Severity: Medium

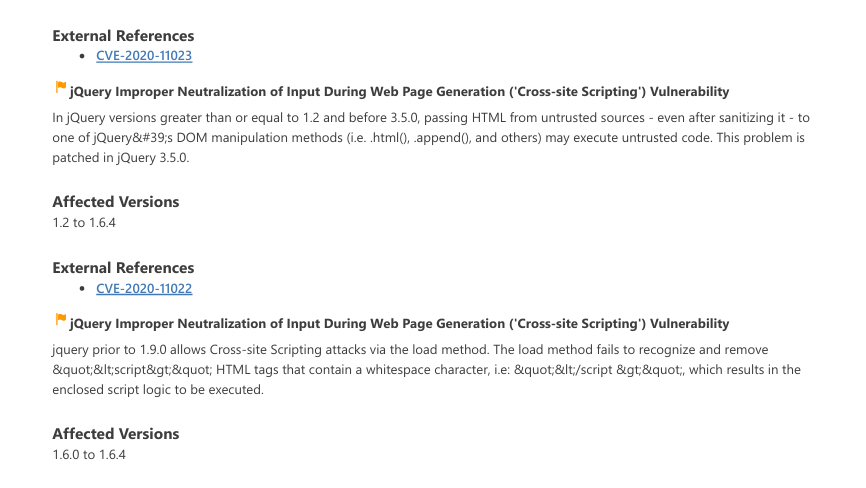
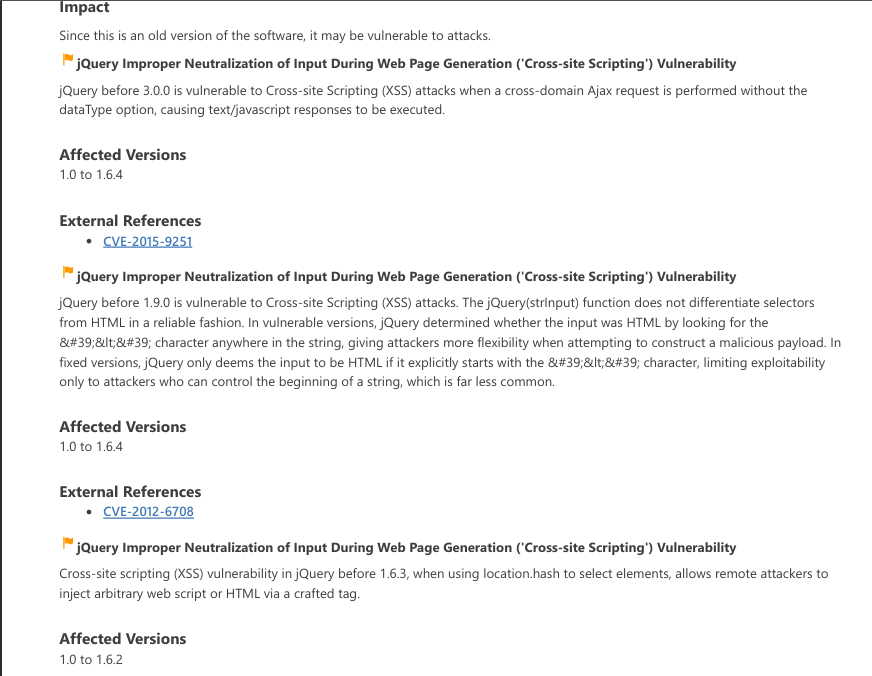
Owasp 2013: A9

Owasp 2017: A9

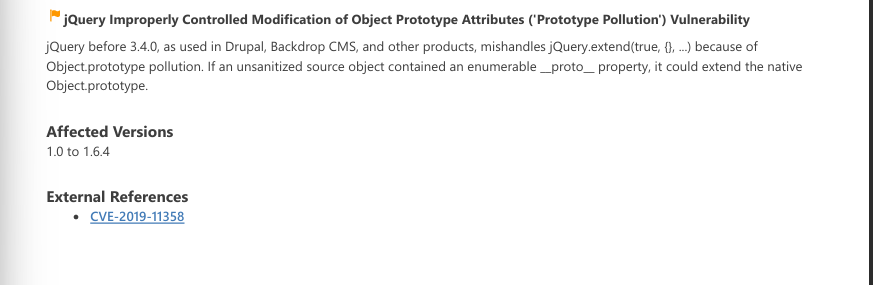
**Effected components:**

* + - jQuary Library
    - Cross-domain Ajax Requests
    - Execution of javaScript components

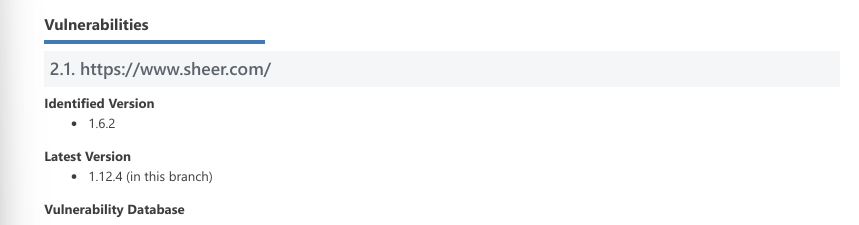
1. **Impact Assessment:**
2. There are major ramifications for online applications for the vulnerability "jQuery Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')" such as, By taking advantage of this weakness, attackers may be able to insert and run malicious scripts inside the web application. This compromises the integrity of the system and puts user data security at risk.
3. This vulnerability's XSS attacks have the potential to provide unauthorized parties access to private user data, including login credentials. Users are at danger of identity theft and other nefarious behaviors as a result of this.
4. Possibility of Session Compromise: Via XSS vulnerabilities, attackers can take over active user sessions and take control of accounts without authorization. This increases the degree of compromise by allowing them to act on behalf of consumers.



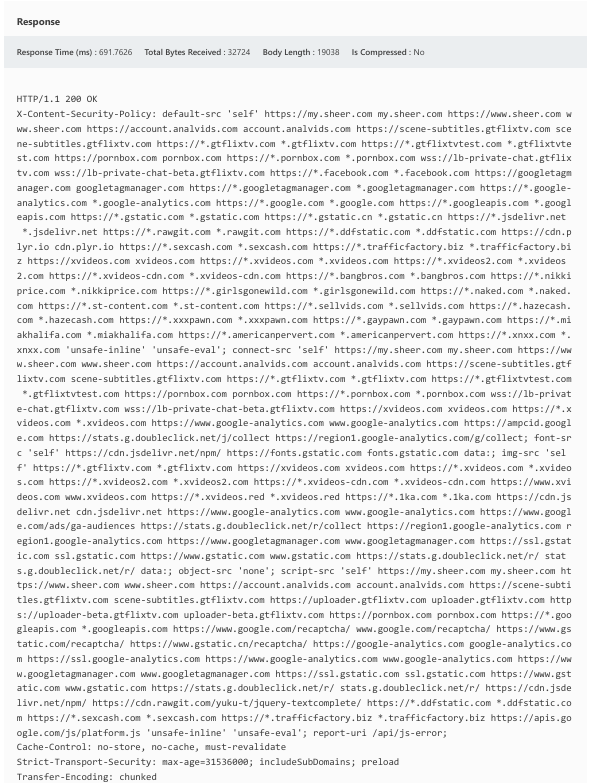
4.And other than XXS attackes, this vulnerability leads to prototype pollution.

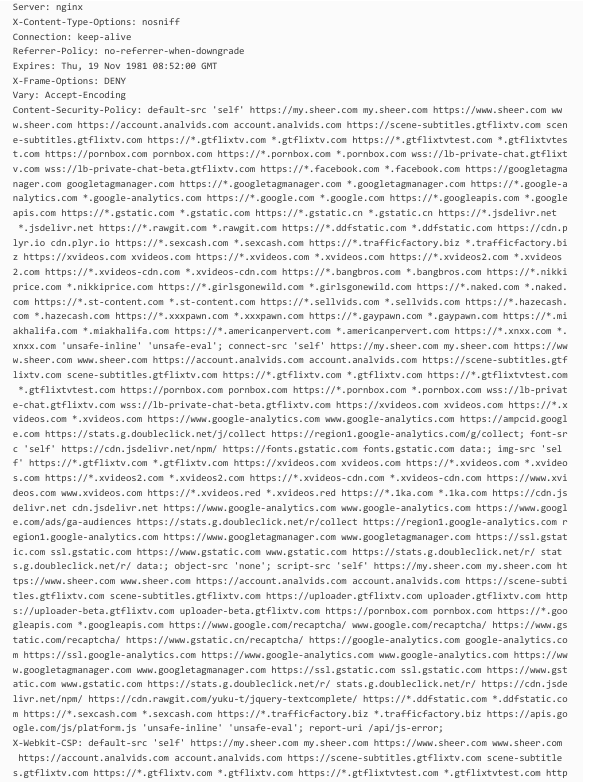


1. **Proof of Concept:**

 **A screenshot of a computer

Description automatically generated**







**c)Steps to reproduce:**

1. Find a web application that is using an out-of-date version of jQuery, particularly any version older than 3.0.0.
2. Find a feature or function that uses cross-domain Ajax requests in the web application.
3. A cross-domain Ajax request should be sent without the dataType parameter specified. This may be done by creating a unique script or by modifying built-in features with browser developer tools.
4. Examine the answer to the cross-domain Ajax request that you received. A possible vulnerability is indicated if the answer contains JavaScript code that is formatted as text/javascript.
5. Add nefarious JavaScript code to the answer. Typically, this code comprises of payloads like <script>alert('XSS')</script> that are intended to take advantage of XSS vulnerabilities.
6. Send the updated request and watch how the web application responds. If the JavaScript code that was injected runs in the context of the online application, it verifies the presence of the vulnerability.

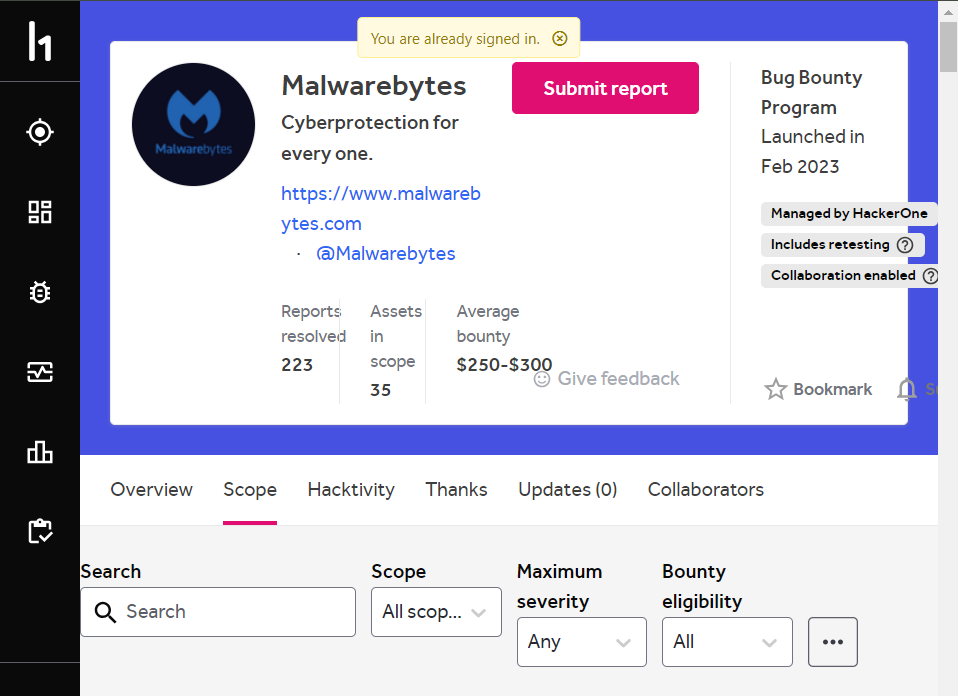
**d)Proposed mitigations**:

1. Update jQuery: Make sure that the web application is using the most recent version of the jQuery library, preferably 3.0.0 or above. The likelihood of XSS attacks is decreased by the security improvements and bug fixes that are frequently included in newer versions.
2. Appropriate Ajax Configuration: Always include the dataType option in your cross-domain Ajax queries. The program reduces the danger of running JavaScript replies as code by declaring the dataType as json or html, which guarantees that responses are handled correctly.
3. Implement content security measures to lessen the possibility of cross-site scripting (XSS) attacks. With CSP, you may implement a strategy that restricts the types of content that can be loaded and shown on your webpages. By correctly designing CSP directives, malicious scripts that are loaded using weak components can be prevented from initiating.

**Report 05):**

**Domain: -** [**http://www.malwarebytes.com/**](http://www.malwarebytes.com/)

Primarily, Malwarebytes functions as a scanner, identifying and eliminating harmful software such as spyware, adware, and rogue security software. Rather of scanning every file that is accessed, Malwarebytes scans in batch mode, which minimizes interference from other on-demand anti-malware programs that may be operating on the same machine**.**

****

**Title:** - **Cross-site Request Forgery (CSRF)**

Cross-Site Request Forgery (CSRF) is a common vulnerability in online applications where malicious actors deceive authorized users into unintentionally performing actions. They usually create a script or link that exploits the user's authorized session, allowing the attacker to operate on the user's behalf without authorization. CSRF attacks pose significant threats to the security and integrity of the affected web application and its users, leading to data theft, account takeover, and illegal transactions.

Severity: Low

OWASP 2013: A8

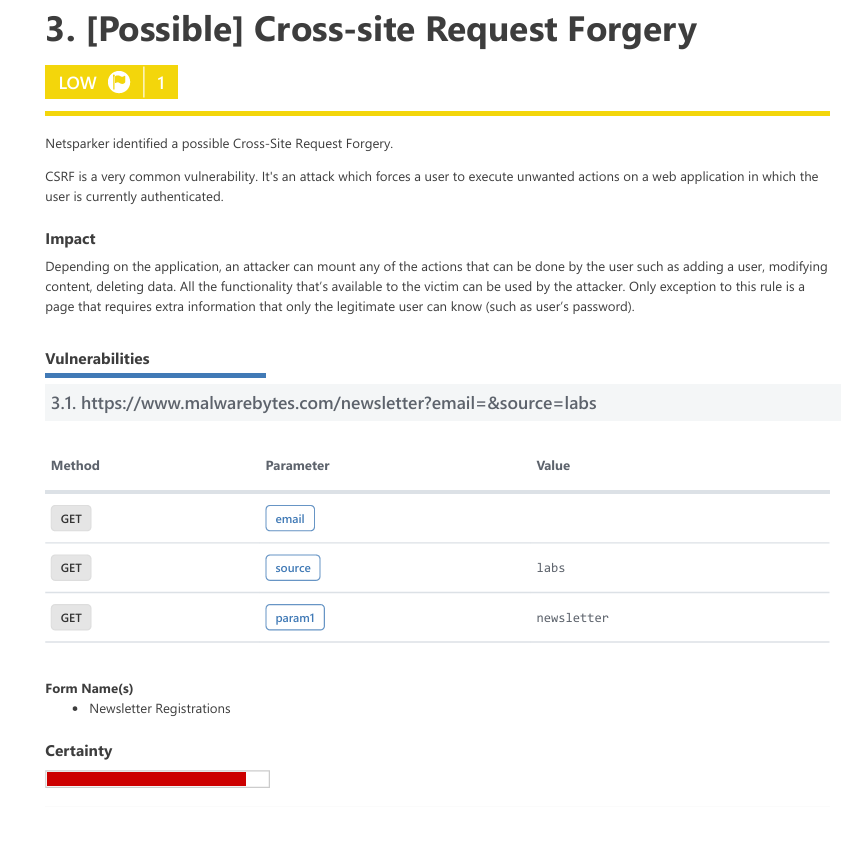
OWASP 2017: A5

**Effected components:**

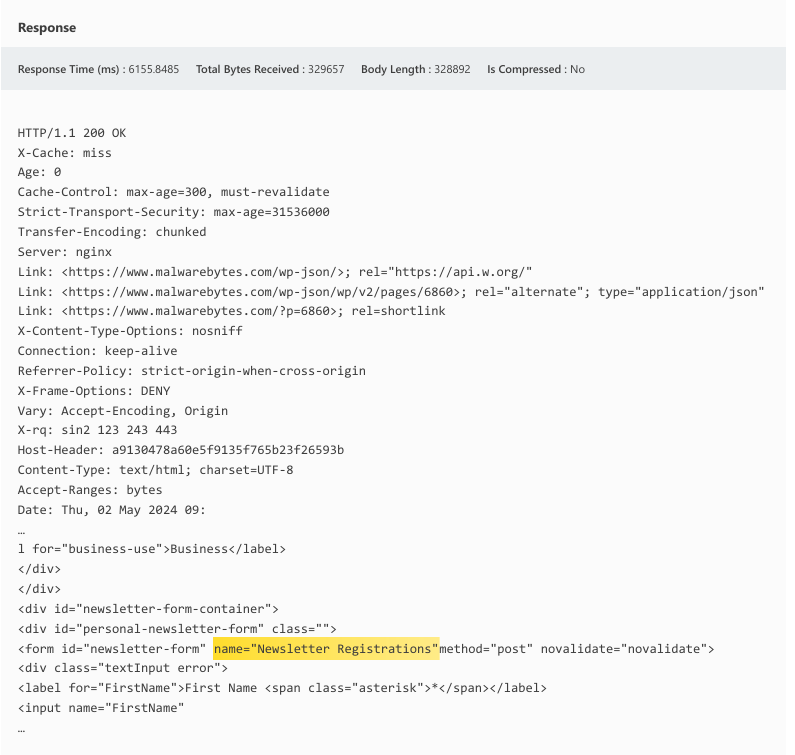
* User Authentication
* Web application Endpoints
* Web Browser

1. **Impact Assessment:**
2. Unauthorized activities: Using CSRF attacks, attackers can carry out unauthorized activities—that is, acts taken without the user's knowledge or consent—on behalf of verified users. This might include adjusting account preferences, transferring money, erasing information, or carrying out other delicate tasks inside the online program.
3. Data Integrity: User data held in a web application may be compromised by cross-site scripting (CSRF) attacks. Attackers have the ability to alter or remove user data, which can result in data loss, corruption, or unapproved changes.
4. Cross-site scripting (CSRF) attacks can cause financial losses for both users and affected companies by targeting financial transactions or payment processing features, allowing attackers to initiate fraudulent transactions, transfer funds to unauthorized accounts, or make unlawful purchases.
5. CSRF vulnerabilities can lead to non-compliance with privacy, data protection, or industry standards, with fines, penalties, or legal repercussions depending on the type of exposed data and extent of the assault.

**b)Proof Of Concept:**

****





1. **Steps to reproduce:**
2. To begin, pick a website that is not sufficiently guarded against cross-site request forgeries. These vulnerabilities are frequently discovered in online applications that lack additional safeguards, such as CSRF tokens.
3. Determine which important online application operations, if altered by an attacker, might have unfavorable effects. These might include removing user data, transferring money, or altering account settings.
4. Construct a spoof HTTP request that instructs the target website to perform the specified activity. This request comes from an unapproved source that the attacker controls, but it should look and feel like a valid request made by verified users.
5. Allow the attacker to control a webpage or script that contains the forged request. This might entail integrating or configuring a server or embedding malicious content within an email or post.
6. Social engineering strategies, such as phishing emails and social media links, can be employed to persuade legitimate users to engage with harmful information on the intended website.
7. The attacker initiates an attack by triggering a forged request when an unsuspecting user interacts with malicious content, trusting the website to execute the action, unaware of the attacker's intentions.
8. Monitor the effects of the CSRF attack on the target website. Verify that the unauthorized action, such as modifying account settings or initiating a financial transaction, has been successfully carried out.

**c)Proposed Mitigations:**

1. Create and append distinct CSRF tokens to every request or form that is submitted. To be sure that requests come from reliable sources and haven't been tampered with, validate these tokens server-side.
2. To limit the reach of cookies to the same origin as the web application, use the SameSite property. This lessens the possibility that hostile websites may make requests using credentials, hence preventing CSRF attacks.
3. To stop unwanted submissions, add anti-CSRF tokens to AJAX calls and web forms. To prevent attackers from attempting to fabricate requests, these tokens have to be erratic and associated with the user's session.
4. Origin validation ensures incoming requests come from expected sources and rejects requests from unknown or unreliable ones to reduce cross-site request forgeries.
5. CAPTCHA challenges are used to thwart automated cross-site request forgeries (CSRF) attacks by making it harder for attackers to send false requests due to human interaction.

**Report 06)**

**Domain:** [**http://www.elastic.co/**](http://www.elastic.co/)

This is an organization that helps its users to manage their organizations, their employees and their customer smoothly along with giving security against cyber threats.

A screenshot of a website

Description automatically generated

**Title: - Out-of-date Version (MathJax)**

The usage of out-of-date MathJax library versions in online applications is known as the "Out-of-date Version (MathJax)" vulnerability. This can result in security issues including cross-site scripting (XXS) and denial-of-service attacks (DoS). Attackers are continually looking for websites that utilize MathJax because of these vulnerabilities, which they may use to steal user data, implant malicious scripts, or carry out other harmful actions. Here, this vulnerability leads to Regular expression Denial of Service in Mathjax.js and XXS attack in the javaScript running in the browser.

* Severity: High
* OWASP 2013 A9
* OWASP 2017 A9

**Effected Components:**

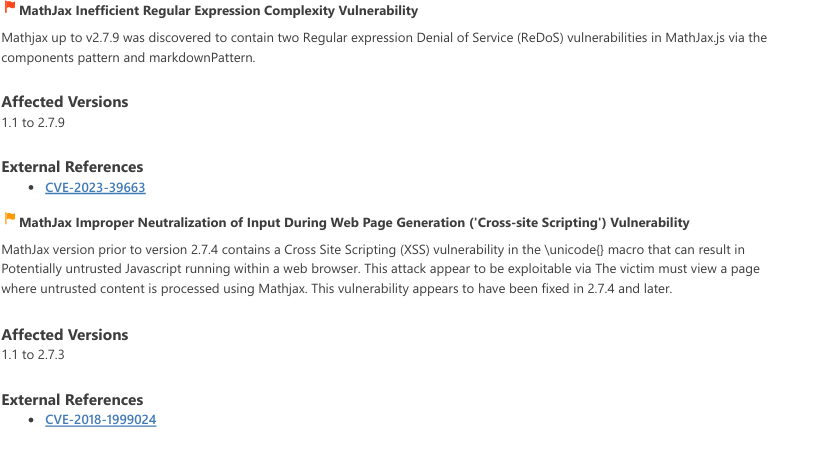
* Web application that incorporates the outdated version
* Outdated version of the MathJAx Library
* Client-side Scripts
* User interfaces which use Mathjax

1. **Impact Assessment:**
2. Vulnerabilities related to Regular Expression Denial of Service (ReDoS):

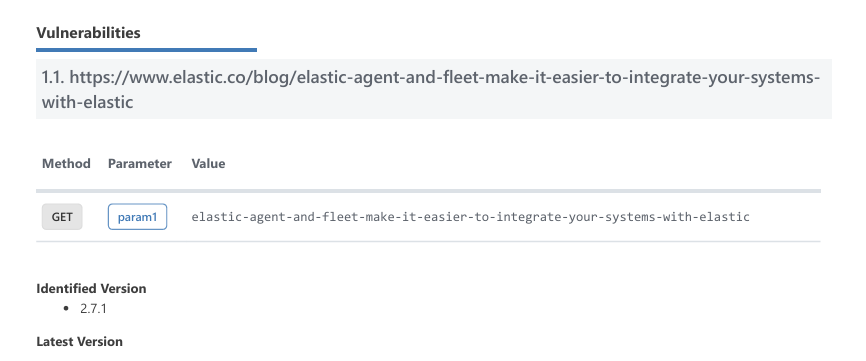
Up until version 2.79 of MathJax, there were ReDoS vulnerabilities that allowed attackers to take advantage of regular expression inefficiencies to launch denial-of-service (DoS) attacks. These assaults may lead to high CPU usage and resource depletion, disrupting services and blocking access to the web application for authorized users.

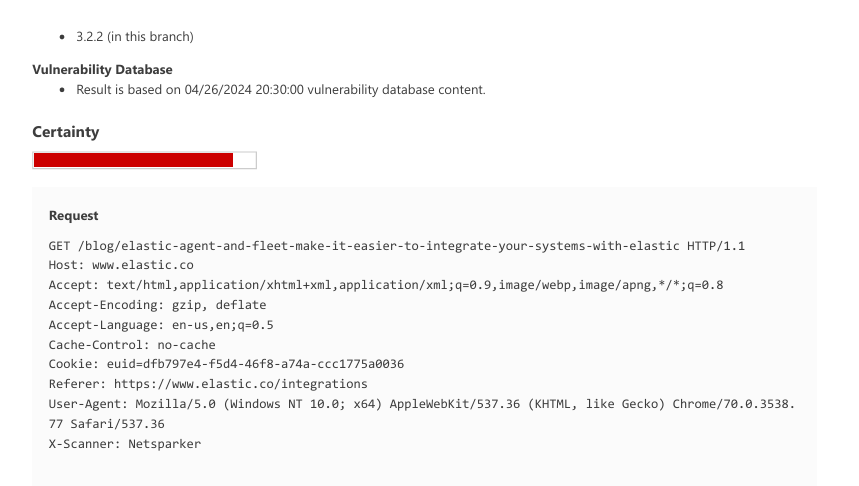
1. Cross-Site Scripting (XSS) Vulnerabilities:

Inadequate input neutralization during web page creation makes MathJax versions less than 2.7.4 vulnerable to XSS attacks. Attackers can run unauthorized scripts in users' browsers by inserting malicious JavaScript code into mathematical formulas that MathJax processes. Serious hazards result from this, such as data theft, session hijacking, and unauthorized access to private data.



1. **Proof of Concept:**

****

****



**Steps to reproduce:**

1. Verify the MathJax version that is being used in the target web application and determine whether it is between 1.1 and 2.79, which is the susceptible range for the vulnerabilities that have been found.
2. Determine whether parts or functionalities of the website depend on MathJax to render mathematical material. Usually, these domains deal with the processing or presentation of mathematical expressions.
3. Create payloads designed to take advantage of the vulnerabilities found. Create inputs for ReDoS vulnerabilities (CVE-2023-39663) that cause inefficient regular expressions to run at high CPU use. Create harmful mathematical formulas that include JavaScript code to exploit XSS vulnerabilities (CVE-2018-199902).
4. Introduce the modified payloads into the web application's weak points. This might involve completing forms, entering information into fields, or tampering with URLs in order to trigger the susceptible MathJax features.
5. After deploying exploitation payloads, monitor the web application's behavior, especially for ReDoS vulnerabilities. Look for signs of denial of service (DoS) conditions like increased CPU consumption or unresponsiveness.
6. Check if injected JavaScript code runs in users' browsers for cross-site scripting (XSS) vulnerabilities. Verify successful reproduction by verifying DoS circumstances or unapproved scripts in users' browsers

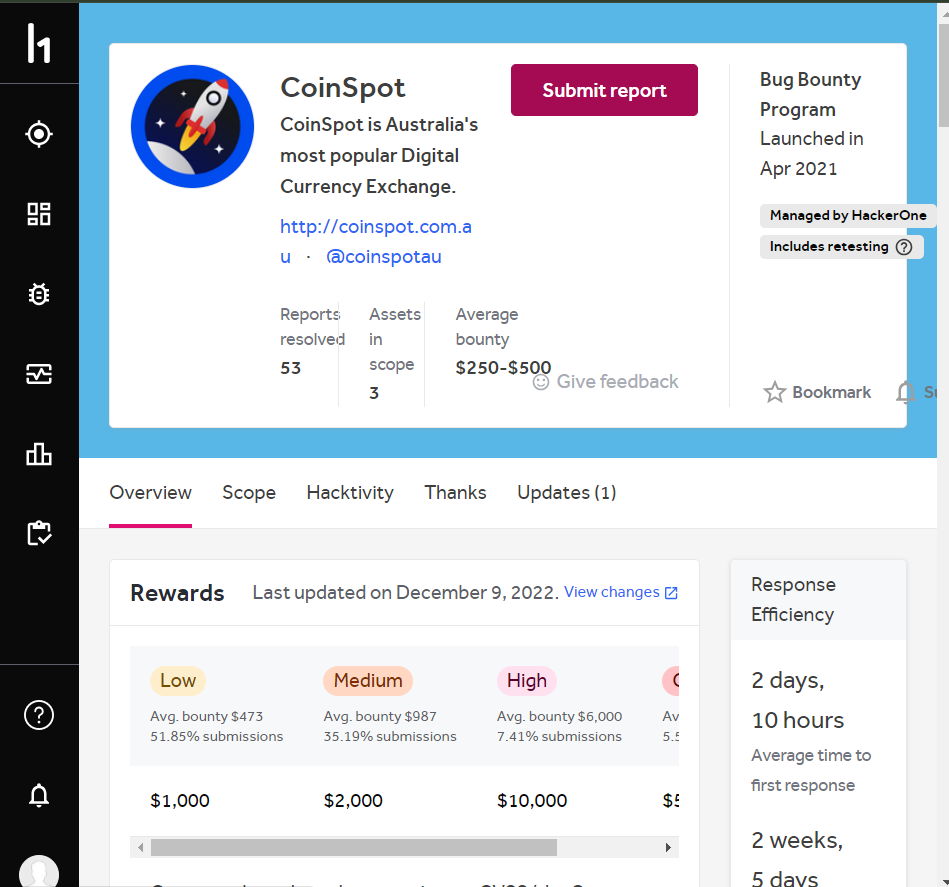
**Proposed Mitigations:**

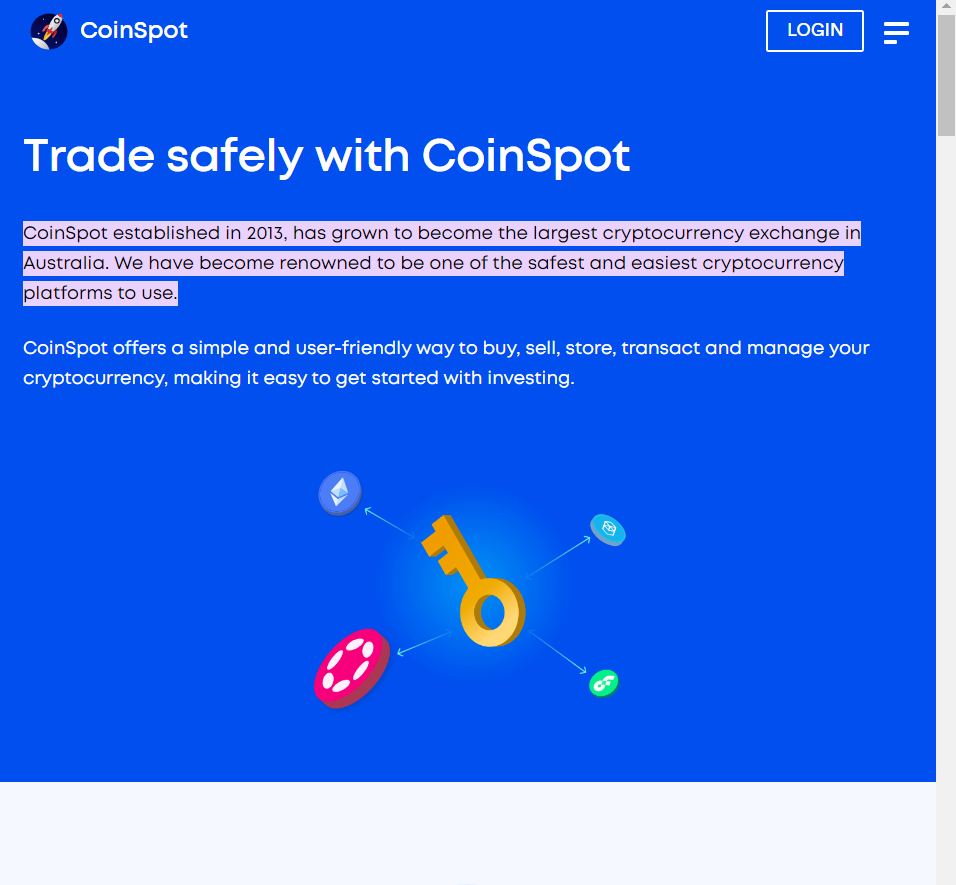
1. Update MathJax: Make sure the web application's MathJax library is always up to date with the most recent version.
2. Enable strong input validation procedures to clean user input before using MathJax to process it. You can stop malicious inputs from taking advantage of holes in the library by verifying and cleaning input data.
3. To lessen the possibility of Regular Expression Denial of Service (ReDoS) attacks, go over and improve the regular expressions that are utilized in MathJax. Regular expression optimization helps lessen the effect of inefficient patterns that might cause a denial of service.
4. Before using MathJax to produce user-generated mathematical equations, sanitize them using input sanitization procedures.
5. Perform routine code reviews and security audits of the web application with an emphasis on MathJax usage. Maintaining a secure environment and reducing the chance of exploitation.

**Report 07)**

**Domain: -** [**http://www.coinspot.com.au/**](http://www.coinspot.com.au/)

CoinSpot is a reputable Australian Cryptocurrency, which has become the largest cryptocurrency exchange in Australia. This is a userfriendly interface which supports buy, sell and trading in cryptocurrency.





**Title:** **- Weak Ciphers Enabled**

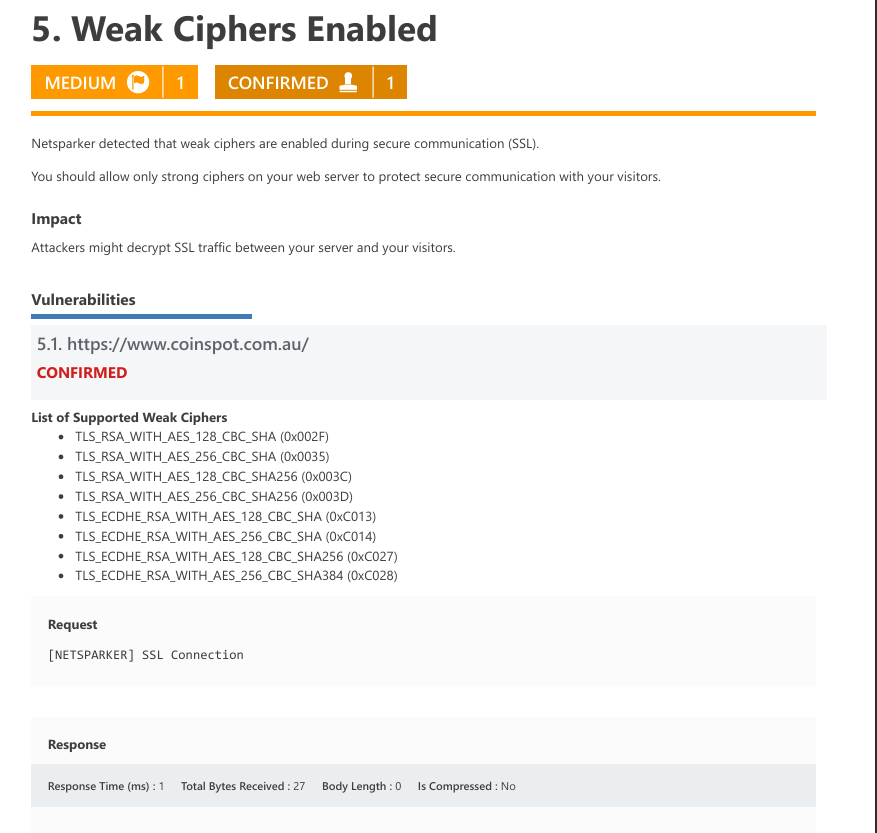
I have found a weak cipher enabled weakness within the vulnerable system. This weakness allows an attacker to do Cipher suite downgrade, Eavesdropping, man-in-the-middle, or data tempering attacks. By exposing this flaw, an intruder could possibly steal private user information, change website content, or perform other harmful actions.

* Severity: Medium
* OWASP 2013 A6
* OWASP 2017 A3

**Effected components:**

* SSL/TLS communication.
* Authentication.
* PCI/DSS security frameworks.

1. **Impact assessment:**
2. Allowing weak ciphers to be used during SSL connection has the effect of giving attackers the opportunity, with the right technique, to decode the SSL communications between the client and server.
3. The web application is more vulnerable to various cryptographic attacks using shoddy ciphers, such chosen-plaintext, cipher-text-only, and brute force. These vulnerabilities provide hackers access to confidential data sent between the machine and the server.
4. If weak ciphers are permitted, security norms and regulations might not be observed. Numerous security standards, such as the Payment Card Industry Data Security Standard (PCI DSS) and the General Data Protection Regulation (GDPR), stipulate that robust cryptographic techniques and ciphers are required to safeguard confidential information. Should these requirements not be fulfilled, there can be legal and regulatory ramifications.
5. Some weaker ciphers could have a larger processing cost than stronger ones. If such ciphers are permitted, the server's processing time and resource consumption may increase, which may impact the web application's responsiveness and performance.
6. **Proof of concept:**

****

1. **Steps to Reproduce:**
2. There are several online tools that can scan the SSL/TLS settings of your web application and show the ciphers that are allowed. The SSL Server Test from SSL Labs is one such tool.
3. The tool will perform a full review of your SSL/TLS setup, including the available ciphers, after you enter the URL of your web service.
4. Review the check results when the program has finished its study to find any ciphers that are allowed in your web application that are weak or out-of-date.
5. Run a cipher negotiation test against your web service using a network testing tool like OpenSSL or Nmap. In this test, you'll create a link to your web service and watch as the agreed cipher suite is formed during the handshake.
6. Examine the cipher suite that was agreed upon: See if any weak ciphers were picked for the link by looking at the results of the encryption negotiation test. Search for any ciphers that are regarded as being weak, revealed, or old.

**d)Proposed Mitigations:**

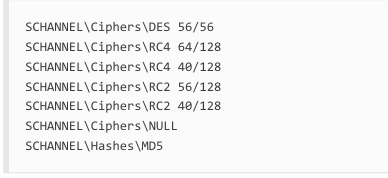
Solution may differ mechanism used to implement the cipher suite. Can suggest some solutions for Apache HTTP server, Lightpd, Microsoft IIS, and node js.

1.For Apache, you should modify the SSLCipherSuite directive in the httpd.conf.



2.Lighttpd:



3.For Microsoft IIS, you should make some changes to the system registry.

**Report 08)**

**Domain:** [**https://node.e2ro.com/\***](https://node.e2ro.com/*)

**Title: - HTTP Strict Transport Security (HSTS) Policy Not Enabled**

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure (HTTPS) connections. The HSTS Policy is communicated by the server to the user agent via a HTTP response header field named "Strict Transport-Security". HSTS Policy specifies a period of time during which the user agent shall access the server in only secure fashion.

Severity: Medium

OWASP 2013 A6

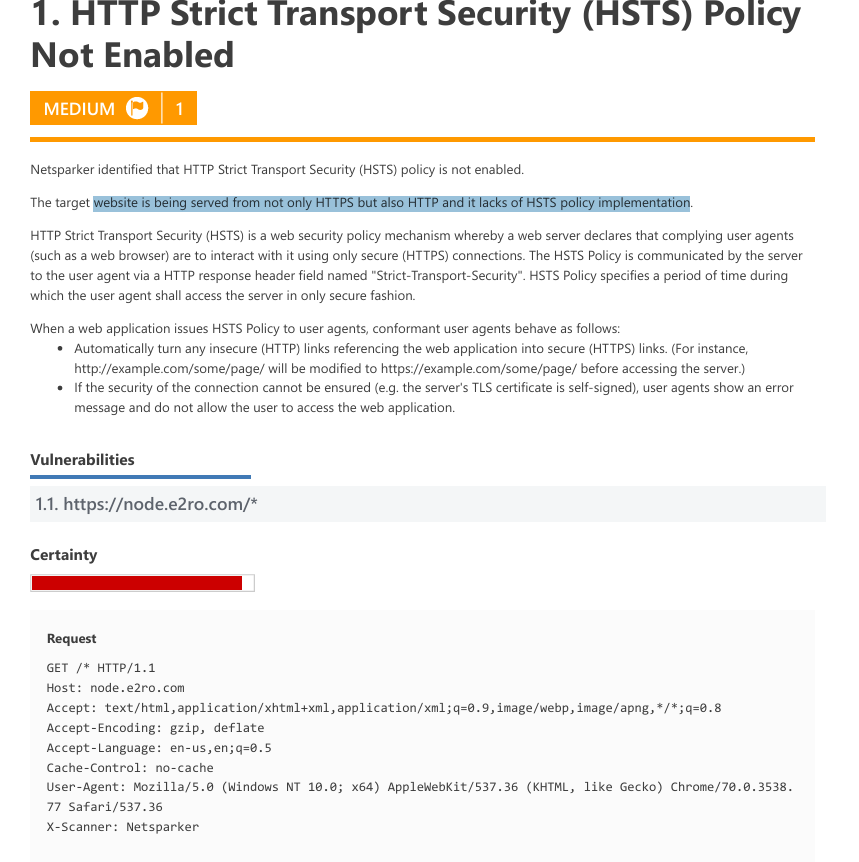
OWASP 2017 A3

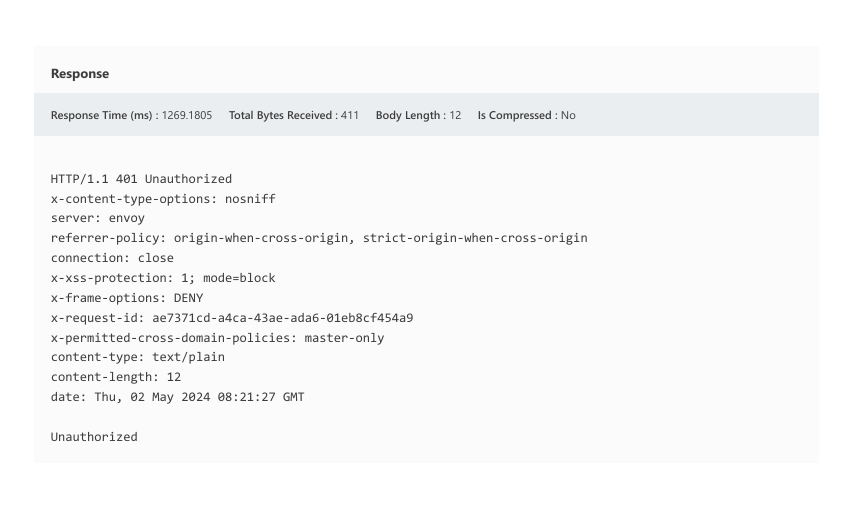
**Effected Components:**

* Communication channels
* Web server configuration
* Security of the data

1. **Impact assessment:**
2. Users are exposed to security threats, especially with regard to data security, when both HTTP and HTTPS connections are present. Because HTTP is not encrypted, data transferred over it can be intercepted and manipulated by nefarious parties. This greatly raises the possibility that private data, including bank account information and user credentials, may be stolen.
3. In the absence of HSTS policy enforcement, users may be forced to utilize insecure HTTP connections rather than HTTPS by attackers in the event of a protocol downgrade assault. Due to its increased vulnerability to unwanted change without notice, this compromises the integrity of data that is delivered.
4. Users' faith in the website is undermined by inconsistent security measures. Users may have doubts about the platform's dependability and security, which would reduce participation and confidence.
5. Keeping both HTTP and HTTPS connections open increases the attack surface and leaves the website vulnerable to many types of assaults, including session hijacking and man-in-the-middle (MITM) attacks. Attackers employ the lax enforcement of HSTS policies to intercept traffic, alter it, assume the identity of the website, and steal confidential user data.

**Proof of Concept:**

****

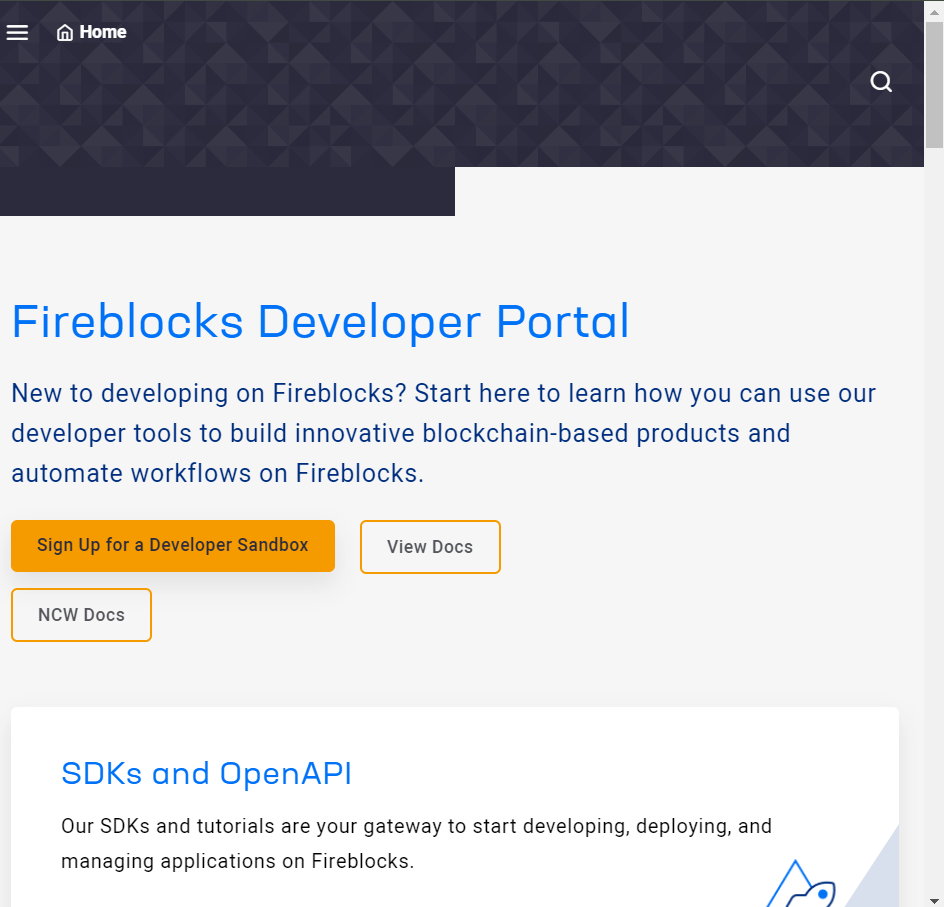


**Steps to Reproduce:**

1. Determine the URL: Find a website that does not enforce HSTS rules and that provides content via both HTTP and HTTPS protocols. This may be determined by using the "http://" and "https://" prefixes in the URL to visit the website.
2. Monitor Network Traffic: To record and examine network traffic between the client and the server, use tools for network monitoring, such as Wireshark or browser development tools. See if any HTTP requests or answers were sent back and forth throughout the browsing session.
3. Observe Connection Security: When using HTTP and HTTPS to view a website, take note of the security signs that the browser displays. HTTP connections that are not secure might not indicate the absence of encryption with a padlock symbol or other warnings.
4. To intercept and modify HTTP requests, use proxy programs like Burp Suite or mitmproxy to intercept client-server HTTP and HTTPS communication. Modify HTTP requests to force browsers to convert insecure connections to HTTPS, and monitor if HTTPS is enforced and the website's response.
5. Check to see if the website complies with HSTS rules by looking through the response headers that the server sends. Keep an eye out for directives in the "Strict-Transport-Security" header that indicate if subdomains are included and the maximum age.
6. Attackers can manipulate browser settings or alter HTTPS answers to cause a browser to downgrade from HTTPS to HTTP. Monitor the browser's actions to prevent unsafe connections. Look for mixed content, where resources loaded over HTTP are present on HTTPS sites, as it can compromise HTTPS connection security.
7. **Proposed Mitigations:**
8. To guarantee that all communication is encrypted and safe, set up the web server to automatically redirect all HTTP traffic to HTTPS.
9. To lower the danger of data interception over insecure channels, enable HSTS policy on the web server to require the usage of HTTPS for all connections.
10. By designating trustworthy content sources and limiting the execution of scripts from untrusted origins, CSP may be used to fight protocol downgrade attacks and prevent mixed content problems.
11. To protect sensitive data from exposure on unsecured channels, set the "secure" flag on cookies to only allow transmission over secure HTTPS connections.

**Report 08)**

**Domain:** [**https://developers.fireblocks.com**](https://developers.fireblocks.com)

****

**Title:** BREACH Attack Detected

An exploit known as the BREACH attack targets the HTTP compression algorithms used in answers, especially those that use the HTTP compression standard, such GZIP. The attack makes use of the capacity to track variations in compressed data sizes across encrypted connections in order to deduce sensitive data that is delivered inside the response body, such as authentication tokens or CSRF tokens. An attacker might infer the content of the plaintext and perhaps extract sensitive data by introducing controlled plaintext into requests on a regular basis and tracking the changes in compressed response sizes that result.

Severity: Medium

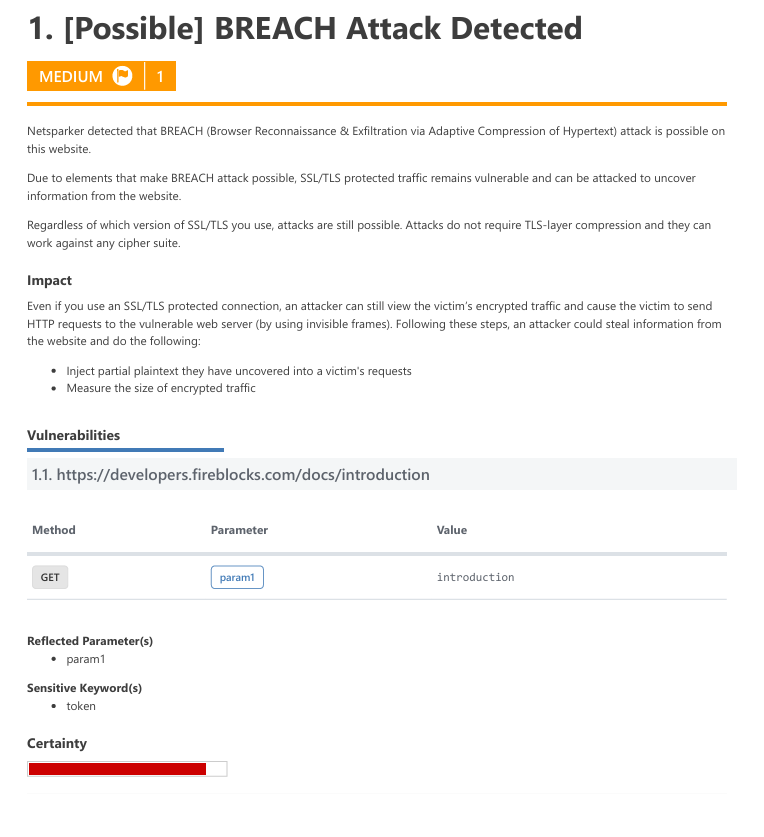
OWASP 2013 A9

OWASP 2017 A9

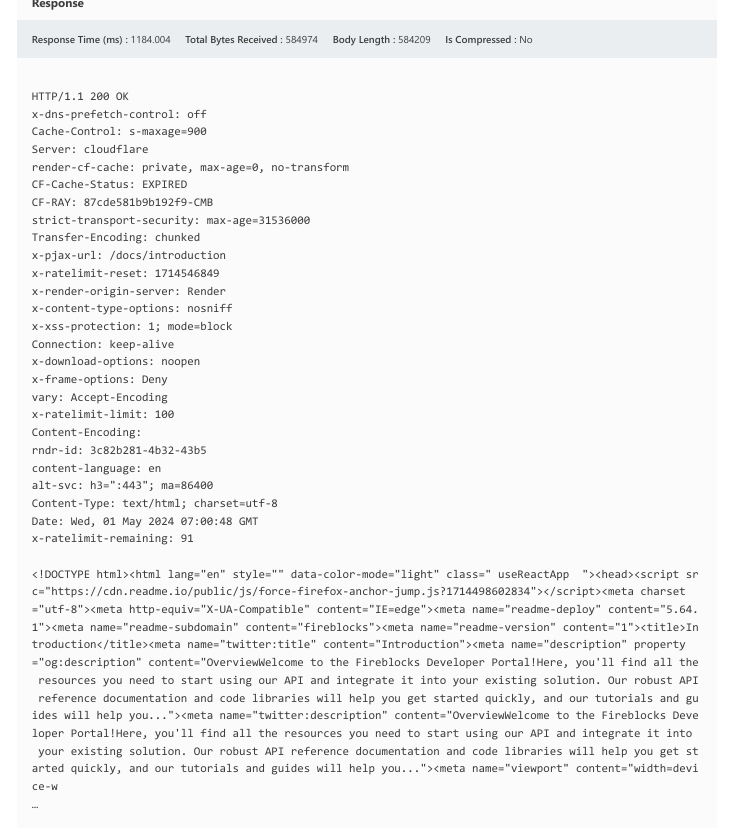
**Effected Components:**

* User accounts.
* Server infrastructure.
* Data confidentiality.

1. **Impact Assessment:**
2. The result of this weakness is that an attacker may be able to use this online application to conduct a BREACH (Browser Reconnaissance & Exfiltration through Adaptive Compression of Hypertext) attack.
3. It is still possible to carry out this type of attack and to breach the data from the online application due to factors that make the BREACH attack conceivable on this web site, even if it is secured with SSL/TLS protected communication.
4. Compression feature of HTTP exploited: BREACH makes use of the compression function of HTTP to lower the size of the answer body. An attacker can determine whether certain information is present in the answer by changing the content and tracking changes in the compressed response size.
5. Side-channel attack vector: The BREACH attack uses numerous requests to track changes in the size of compressed replies as a side-channel attack vector. An attacker can receive private information by watching the differences in the compressed answer sizes and conducting controlled searches while assessing these changes.
6. **Proof of concept:**

****





c)**Steps to reproduce:**

1. Choose a website that compresses HTTP replies using GZIP and handles sensitive user data, like CSRF or authentication tokens, for optimal performance.
2. To intercept and examine communication between the attacker's machine and the target website, use tools such as Wireshark or Burp Suite for network monitoring. Watch the payloads that are transferred during site interactions—both requests and responses.
3. An attacker can inject controlled plaintext into HTTP queries to a destination website, aiming to extract known values or patterns from compressed answers, such as CSRF tokens or sensitive data segments.
4. Monitoring plaintext injection variations in response sizes is crucial as HTTP compression methods like GZIP eliminate redundant data. By continuously injecting controlled plaintext and assessing response sizes, attackers can refine their attack and modify the injected plaintext as needed to obtain the required data from compressed replies.
5. Through scrutinizing changes in response sizes due to the injected plaintext, attackers can deduce sensitive information embedded within compressed responses. This may involve extracting authentication tokens, CSRF tokens, or other sensitive data transmitted by the target website.

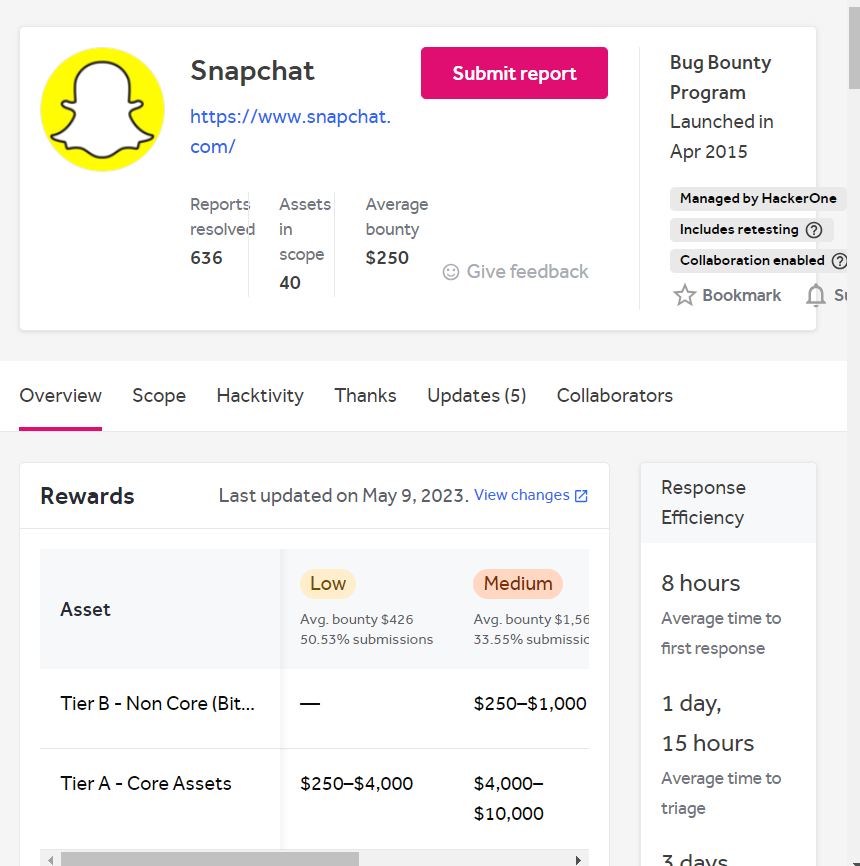
**d)Proposed Mitigations:**

1. If possible, disable HTTP level compression.
2. Separate sensitive information from user input.
3. Protect vulnerable pages with CSRF token. The SameSite Cookie attribute will mitigate this issue, because to exploit this issue an attacker forces the victim to visit a target website using invisible frames.
4. With the SameSite cookie attribute added, cookies that belong to the target won't be sent with a request that does not include top level navigation.
5. Hide the length of the traffic by adding a random number of bytes to the responses.
6. Add in a rate limit, so that the page maximum is reached five times per minute.

**Report 10)**

**Domain:** [**http://accounts.snapchat.com/**](http://accounts.snapchat.com/)

Snapchat is a mobile app that allows users to send and receive "self-destructing" photos and videos. Snaps are images and videos captured with the app. Snapchat takes pictures with the device's camera and distributes them to other Snapchat users.



**Title: - Internal IP Address Disclosure**

When a web application unintentionally gives external users access to internal IP addresses, an internal IP address disclosure vulnerability arises. Internal network information may be disclosed via error messages, headers, or answers. Attackers may use this data to perform additional assaults, such as targeted scans or reconnaissance operations, and to learn more about the topology of the network. To reduce the possibility of illegal access and exploitation of internal systems, it is imperative to stop these kinds of disclosures.

Severity: Low

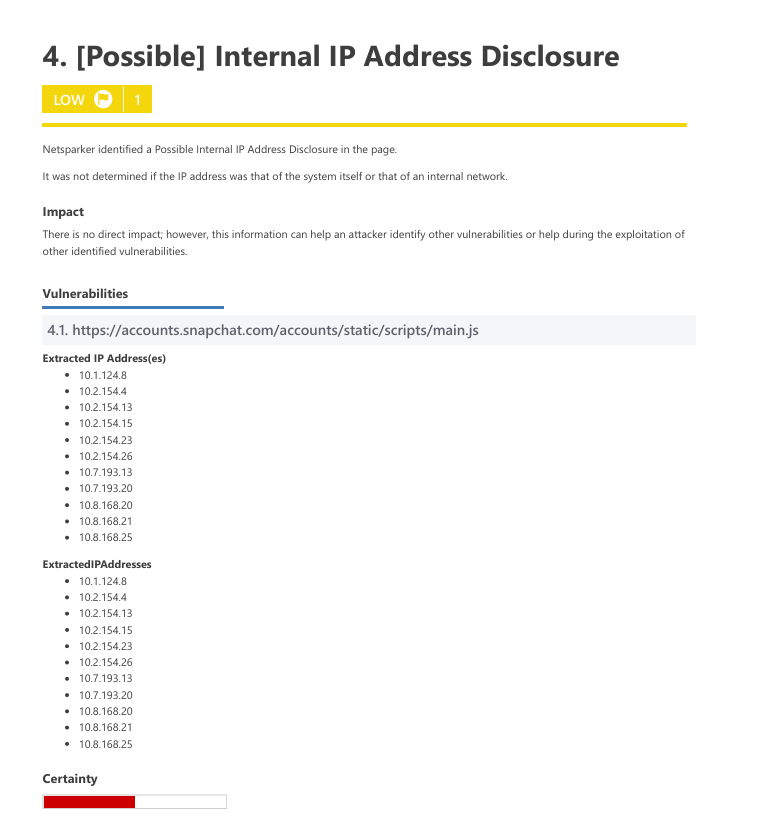
OWASP 2013 A6

OWASP 2017 A3

**Effected Components:**

* + - **Web application**
    - **Server configuration**
    - **Network infrastructure**
    - **End Users**

1. **Impact Assessment:**
2. By using the publicly available internal IP addresses, attackers can map out the network architecture, finding potentially weak points in the system and launching focused assaults.
3. Information about the organization's network infrastructure, including as servers, routers, and firewalls, is made public when internal IP addresses are disclosed. Attackers may use this information to design and carry out more complex assaults.
4. The vulnerability increases the attack surface by disclosing internal IP addresses, which makes it simpler for attackers to locate and target susceptible network systems.
5. The internal IP addresses that have been made public can be used by attackers to carry out further operations, such as directed assaults against vulnerable systems or focused scanning.
6. Sensitive data transferred across the internal network may become less secret if internal IP addresses are made public. Attackers could use this information to obtain sensitive data or internal resources without authorization.
7. **Proof of Concept:**

****





1. **Steps to Reproduce:**
2. Choose Your Target: Select a web application that may unintentionally disclose internal IP addresses through error messages or answer text.
3. Start Requests: Send different requests to the target web application to manipulate input parameters, view different pages, or submit forms.
4. Examine Reactions: Examine the web application's answers to see if there are any hints about internal IP addresses. Error messages, headers, and other response data that inadvertently reveal internal network information may fall under this category.
5. Check for Weaknesses: Make an effort to introduce mistakes or certain actions into the web application that can cause the internal IP addresses to become visible. This might entail taking advantage of known application vulnerabilities or purposefully causing issues.
6. Monitor Traffic: To record and examine the communication between the attacker's machine and the intended web application, use network monitoring tools. Look for any instances in which the answers reveal internal IP addresses.
7. Document Findings: Keep track of any occurrences in which error messages or answer texts disclose internal IP addresses. Take note of the details around the disclosure, such as the page being visited or the particular request that was made.

d**)Proposed Mitigations:**

1. Sanitizing Error Messages: Verify that the web application's error messages don't include sensitive information or internal IP addresses. Adapt error messages so that they give general answers without disclosing internal network information.
2. Web Server settings: Modify the web server's settings to stop internal IP addresses from accidentally leaking in HTTP headers or server replies. This might entail hiding internal infrastructure details using reverse proxies or altering server settings.
3. Network Address Translation (NAT): Prior to sending replies to external users, implement NAT to convert internal IP addresses into ones that are visible to the public. This aids in hiding the topology of the internal network from outside enemies.
4. Content Security Policies (CSP): Use CSP headers to limit the sources from which content may be loaded. This will stop internal IP addresses from accidentally being revealed by scripts or other resources.
5. Frequent Security Assessments: To find and fix vulnerabilities, including the possible leaking of internal IP addresses, conduct regular security assessments and penetration tests. By taking a proactive stance, security flaws are found and fixed before an attacker may take use of them.