Chameleon Website Security Audit



Contents

[Chameleon Website 2](#_Toc2110863708)

[Vulnerability Assessment 3](#_Toc1189182111)

[Tools used: 3](#_Toc2095252016)

[Results 3](#_Toc670419019)

[Content Security Policy Header 4](#_Toc366976022)

[Missing Anti-clickjacking Header 4](#_Toc1603037405)

[Strict-Transport Security Header Not Set 5](#_Toc1190290040)

[X-Content-Type-Options Header Missing 5](#_Toc1858056594)

[Encryption 6](#_Toc448488220)

[Tools Used 7](#_Toc651886232)

[Security Headers 10](#_Toc880794397)

[Tools used 10](#_Toc1454285817)

[The Positives 12](#_Toc1798454188)

[Improvements that can be made 13](#_Toc1689179974)

[Method Validation 13](#_Toc1069437288)

[Tools used 13](#_Toc983522586)

[Error Handling 14](#_Toc147057996)

[Tools used 14](#_Toc1025912639)

[References 16](#_Toc247681519)

# Chameleon Website

The Chameleon website aims to be a hub that showcases the resources of Chameleon as a company and the projects that it is committing to, within this page we can see the goals of the company, the service we provide and the different projects.

## Vulnerability Assessment

The vulnerability Assessment was the first test conducted for this audit and it highlights a few issues that can be further researched to learn how vulnerable or in this case how secure the website is.

### Tools used:

ParrotOS

OwaspZAP

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### Results

### Content Security Policy Header

A Content Security Policy is a HTTP response header that is in modern browsers that can allow or restrict resources and where they come from.

A CSP header is a measure that prevents external resources from being loaded and executed this can help to reduce the possibility of cross site scripting attacks. Within the scan that was ran there are several pages with the same flag simply mentioning that the security policy isn’t as secure as what it should be. Below are a few images indicating what the error looks like and below there will an improved version that could be implemented.

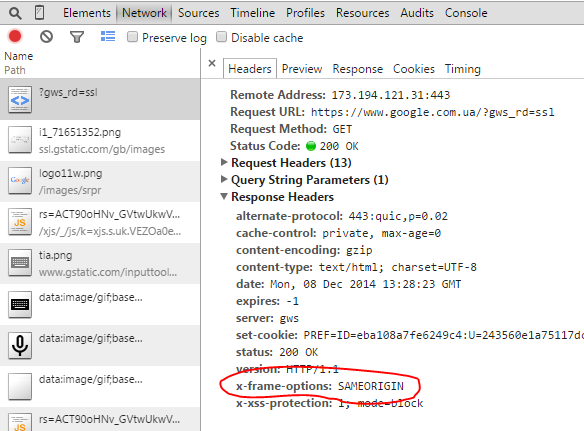
### Missing Anti-clickjacking Header

Clickjacking is a type of cyber-attack where an attacker can trick a user into clicking on something that may not be what they are expecting, instead of legitimate content that may be on the website they are visiting. This can result in users performing actions such as revealing sensitive information or installing malware. Below is an example of a packet that gets received by the user showing the lack of anti-clickjacking method(s)

A screenshot of a computer program

Description automatically generated

In the above example we can see everything received by the user and below is the example of a packet that has the anti-clickjacking method implemented.



By having the response header set as sameorigin it would prevent anyone from running any malicious code that was not originating from the website.

### Strict-Transport Security Header Not Set

Strict-Transport Security is a feature that forces browsers to always use an encrypted connection. Not having this set within the website can leave the users of the website vulnerable to different attacks such as man-in-the-middle attacks, session hijacking and data interception. Being vulnerable to these attacks is obvious that it can potentially compromise the user privacy and data integrity.

A screenshot of a computer

Description automatically generated

Above is an example of a packet received by the user when loading into the chameleon website

Below is a screenshot of an example of a packet that would contain Strict transport security with proper implementation, as you can see in the example packet below the max age of the packet is extremely high can provide long lasting protection against connections that aren’t secure.

A screenshot of a computer

Description automatically generated

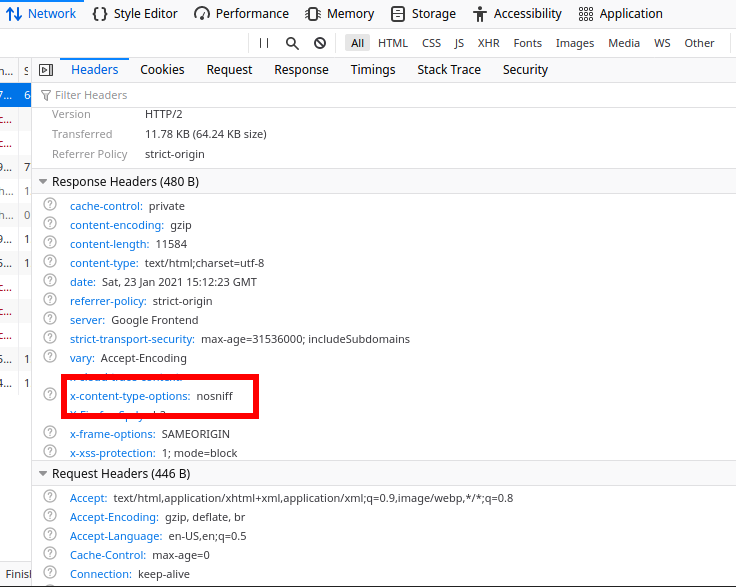
### X-Content-Type-Options Header Missing

The X-Content-Type-Options header is responsible for preventing MIME sniffing attacks. A MIME sniffing attack will attempt to disguise one file type as another to attack a user without them knowing. An attack that can take advantage of this lack of security can include Cross Site Scripting attack(s) that execute code pretending to be a different file format. Below is an example of a packet received while scanning the website.

A screenshot of a computer

Description automatically generated

To make this packet more secure we would need to set the X-Content-Type-Options Header to nosniff like below.



Setting that header as nosniff means that the website will become more secure to Cross Site Scripting attacks since if the website is unsure of what the content is it will not guess, preventing malicious code from being ran if it were to happen.

## Encryption

### Tools Used

* Chromium
* ParrotOS
* Qualys SSL Labs

A screenshot of a certificate

Description automatically generated

A screenshot of a certificate viewer

Description automatically generated

The provided screenshot shows the encryption method implemented to secure the website's traffic. To provide further context, it's worth noting that RSA encryption with a robust key size of 2048 bits is in use. This level of security effectively safeguards against a range of potential threats, including brute force attacks.

**A screenshot of a computer

Description automatically generated**

In order to further test the encryption, I decided to use a third party application dedicated to testing SSL (Qualys) , to ensure that the encryption is doing what should and as we can see above the encryption is passing standards and below we can see why it has a received a grade of B

**A screenshot of a web page

Description automatically generated**

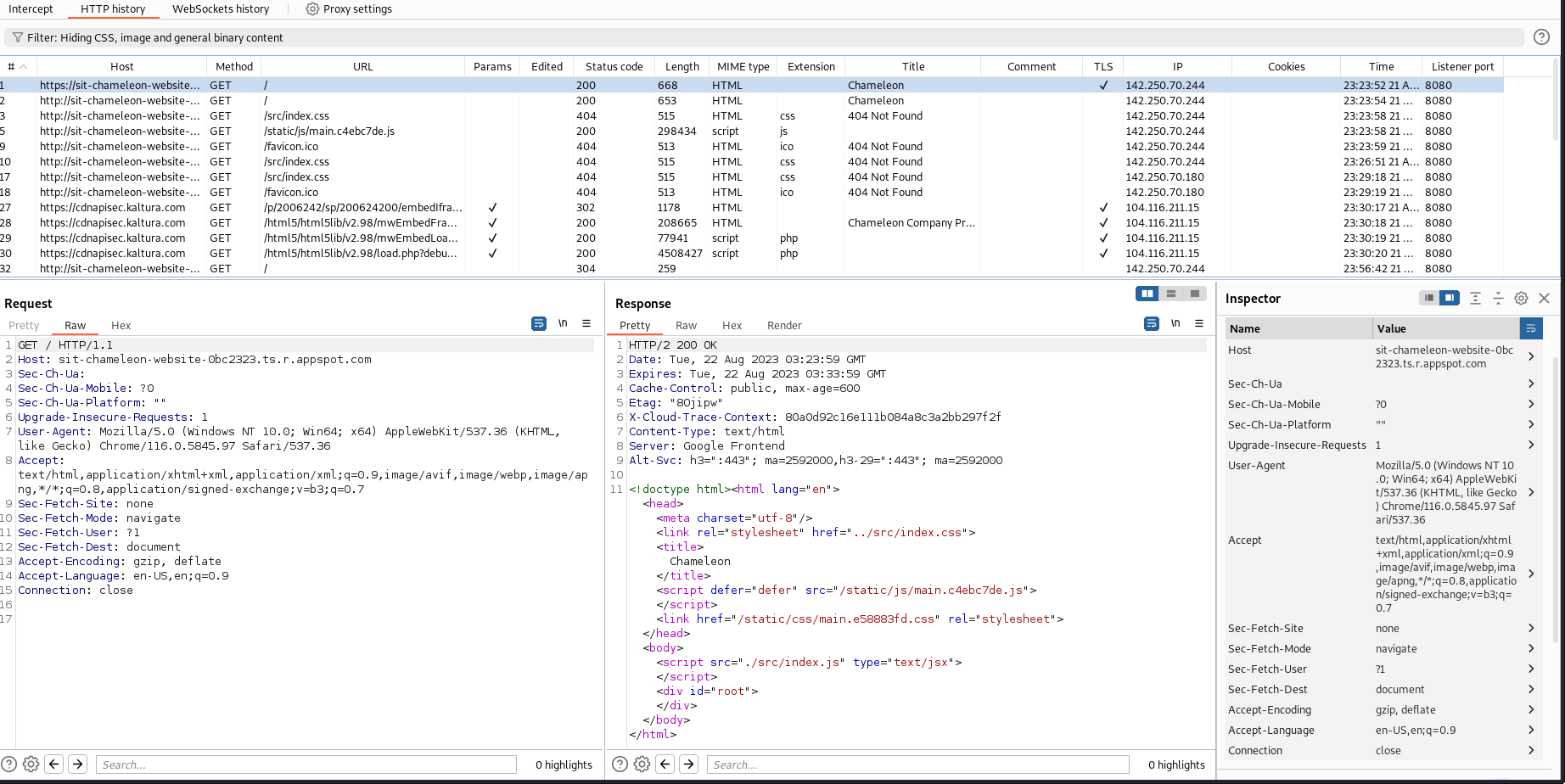
As we can see it is a simple change that can be made to receive an “A” when it comes to security standards and that change is removing support for TLS 1.0 and 1.1, setting a boundary of a minimum TLS 1.2 would increase security however that comes with certain risks as well, since that would lock out users who only have older devices with browsers that can’t be updated but since the world has been moving from TLS 1.0 and 1.1 Krynitsky (2023) perhaps it is time for the Chameleon website to do the same. Not only is TLS 1.0 and 1.1 becoming obsolete it is becoming a regulation to have a minimum of TLS 1.2, not to mention the advantage that can be given with performance and security benefits (Stronger Algorithms, better certificate handling to name a few) (A10networks,

## Security Headers

### Tools used

* Burp Suite
* Kali
* Chromium

This section of the audit will detail the purpose of various headers that are present in the loading of the Chameleon web page. There are many headers that are present so this section will only detail those that more relevant to security.

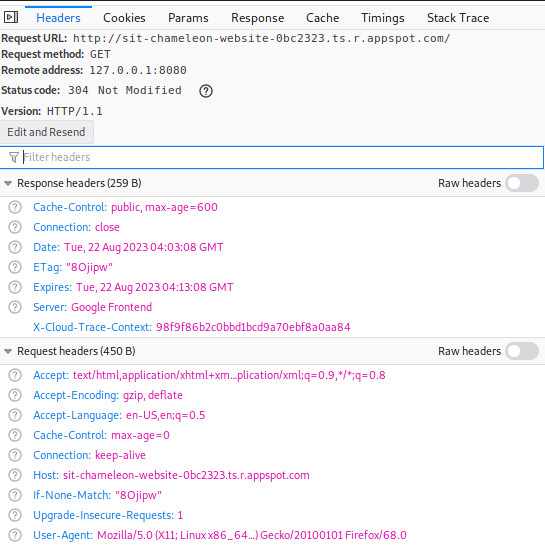


The headers that were present from the interception:

A screenshot of a computer

Description automatically generated

Headers found through dev tools:



The list of headers from each source and their overlaps:

|  |  |  |
| --- | --- | --- |
| Overlap in Security Headers: | Only in Intercept | Only in Dev Tools |
| Accept  Accept-Encoding  Accept-Language  Connection  Host  Upgrade-Insecure-Requests  User-Agent | Sec-Ch-Ua  Sec-Ch-Ua-Mobile  Sec-Ch-Ua-Platform  Sec-Fetch-Site  Sec-Fetch-Mode  Sec-Fetch-User  Sec-Fetch-Dest | Cache-Control  If-None-Match |

**Accept-Encoding:** This header tells the server what compression algorithms the client can understand and use. It helps the server compress the response in a format that the client can decode.

**Cache-Control:** This header details how the client and intermediate caches should handle the response. It can control caching behaviour to improve performance and reduce the load on the web server.

**Connection:** This header controls whether the connection should be kept alive after the current request or if it should be closed. This can help improve performance for the web server by reducing the communication setup overhead.

**If-None-Match:** This header is used in conditional requests. It contains an ETag that the server compares to its current version of the resource. If they match, the server can respond with a 304 Not Modified status instead of the full resource, which in turn saves bandwidth for the server.

**Sec-Ch-Ua, Sec-Ch-Ua-Mobile, Sec-Ch-Ua-Platform:** These headers are used to give information about the user agent's characteristics. This includes whether it is a mobile device, and the platform it is running on. While not specifically security-related, this can contribute to security by allowing the server to optimize content delivery based on the device characteristics that is loading the website.

**Sec-Fetch-Site, Sec-Fetch-Mode, Sec-Fetch-User, Sec-Fetch-Dest:** These headers are used to provide the server with information about the context of fetching resources. They are used to help prevent certain types of cross-site request forgery (CSRF) attacks.

**Upgrade-Insecure-Requests:** This header is used by browsers to request that resources be upgraded to use secure connections (HTTPS) if they are currently loaded over an insecure connection (HTTP). It helps mitigate potential security risks associated with mixed content.

**User-Agent:** This header contains information about the user agent making the request, which is typically the browser. It helps servers give certain responses based on the capabilities of the user agent.

### The Positives

* **Sec-Fetch Headers:** prevent cross-site forgery attacks
* **Upgrade-Insecure-****Requests:** mitigates mixed content security risk
* **Cache-****Control:** improves the performance of the website

### Improvements that can be made

* Lack of XSS prevention headers

## Method Validation

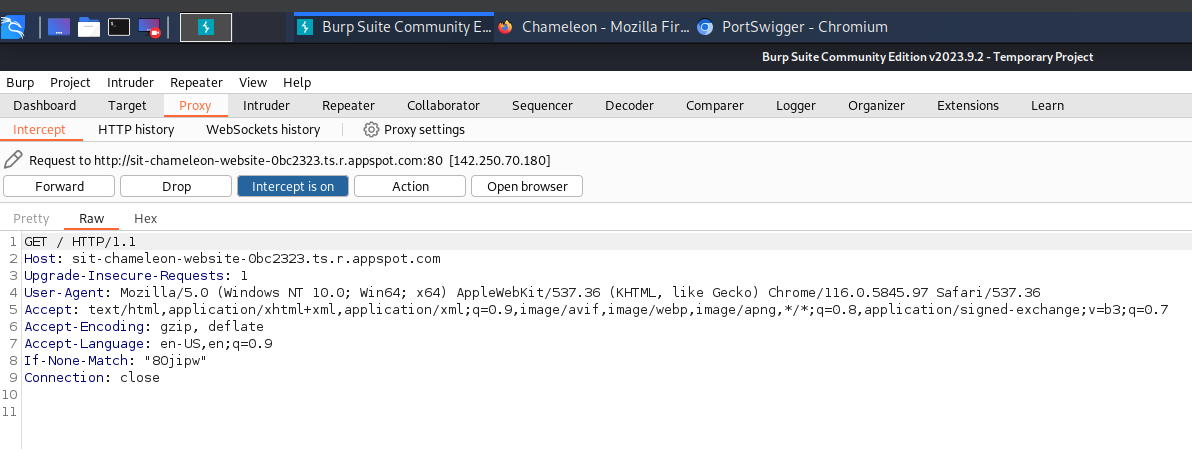
### Tools used

* Burp Suite
* Kali
* Chromium

To test method validation, we have set up a proxy with Burp Suite to intercept the web packets that are being transferred to edit them before we can receive them. When loading a website, burp suite will give us the opportunity to edit a packet before forwarding it to our browser, outputting the result.

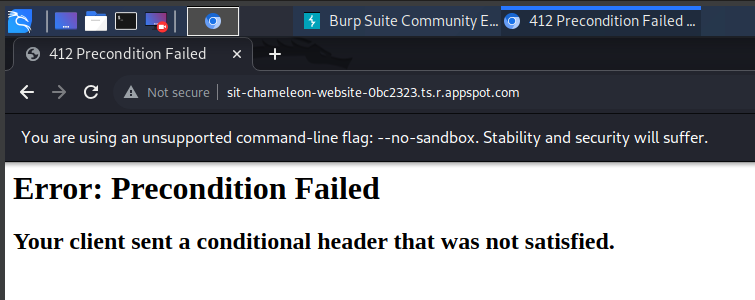
I edited various requests and changed the methods to something different to test what would be outputted to the browser upon forwarding the packet. I tried different requests such as GET, DELETE, and POST requests to examine the different effects.

If the server is not set up properly this will result in requests properly going through and make changes to the website, however we are expecting to be given an error page depending on the request we send to see whether it can deal with a change in request.



This is how the packet we receive looks like upon attempting to load the Chameleon website on Chromium. By editing the GET request on line 1, we can test the method validation.

Changing the GET to DELETE results in this error page:



Changing the GET to POST results in this error page:

A screenshot of a computer

Description automatically generated

As we have run into various errors upon changing the requests, it demonstrates that the requests were not permitted as it errored out rather than proceeding to the Chameleon page. This validation test ensures that the Chameleon website’s security measures are up to date in preventing changes to packet requests.

## Error Handling

### Tools used

* Burp Suite
* Kali
* Chromium

In this section we will explore how the Chameleon website can handle various payloads being sent and whether it will receive the attack and shut down or deny the payloads.

The main set of payloads that I attempted to use were User Agent related attacks as there were a wide variety that I could use to test the aptitude of the web security. After receiving the GET request on Burp Suite, I modified the payload position to enter via the user agent header.

A screenshot of a computer

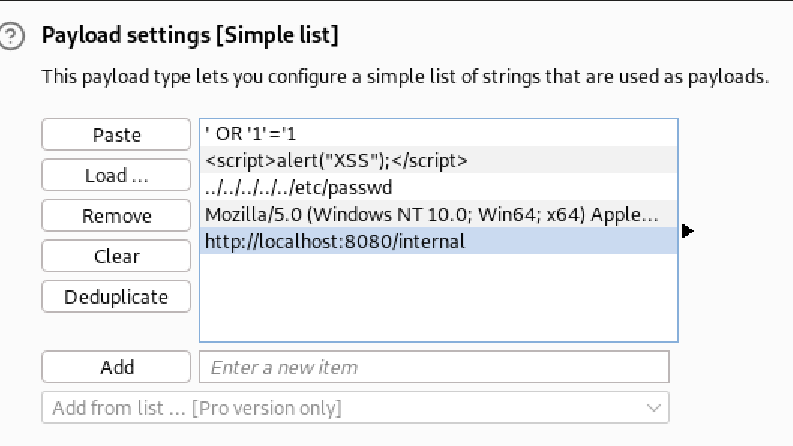
Description automatically generated

I then created a payload list that would attempt to enter into the website via our GET request to see whether Chameleon could handle it.

A screenshot of a computer program

Description automatically generated

Starting the attack resulted in status code 200 for every payload attempt. This means that the request was received successfully by the website, however this payload attack did not affect the website in any way. This is likely due to the website being hosted on GCP, creating a firewall issue that is preventing the payloads from successfully hurting the website.



I tried another attempt with a traversal attack to see if any other kind of payload attack could make it through but it gave the same result.

A screenshot of a computer

Description automatically generated

Showing status code 200 is great for Chameleon’s web security as it is able to detect issues that are present and still persist with loading the website normally. There are no suggestions that can be made to improve this attack currently due to the GCP firewall making payload attacks like these null.

# 

# References

A10networks (n.d.). Key differences Between TLS 1.2 and TLS 1.3 | Glossary. [online] A10 Networks. Available at: <https://www.a10networks.com/glossary/key-differences-between-tls-1-2-and-tls-1-3/#:~:text=TLS%201.2%20vs%20TLS%201.3>.

Krynitsky, J. (2023). TLS 1.0 and TLS 1.1 soon to be disabled in Windows. [online] TECHCOMMUNITY.MICROSOFT.COM. Available at: https://techcommunity.microsoft.com/t5/windows-it-pro-blog/tls-1-0-and-tls-1-1-soon-to-be-disabled-in-windows/ba-p/3887947 [Accessed 3 Aug. 2023].

OWASP (n.d.). HTTP Headers - OWASP Cheat Sheet Series. [online] Available at: https://cheatsheetseries.owasp.org/cheatsheets/HTTP\_Headers\_Cheat\_Sheet.html.

Veracode (n.d.). Error Handling Flaws - Information and How to Fix Tutorial. [online] Veracode. Available at: https://www.veracode.com/security/error-handling-flaws-information-and-how-fix-tutorial.

W3C (2021). Fetch Metadata Request Headers. [online] Available at: https://www.w3.org/TR/fetch-metadata/ [Accessed 27 Aug. 2023].