CHAMELEON

FOR OUR SMARTER WORLD

Cross-Site Scripting

Adam Sarin
217342706

Contents

Executive summary:

This report presents the findings of a Cross-Site Scripting (XSS) vulnerability assessment, which was conducted on the Chameleon website, this report intends to repeat the tests a Junior did last trimester on the City of Melbourne site but aimed at the Chameleon site with the intention of identifying potential XSS vulnerabilities.

Introduction:

Now what is Cross Site Scripting? Well according to the OWASP community XSS attacks are a type of injection, where malicious scripts are injected into otherwise trusted sites, which is possible when an attacker uses a web application to send malicious code, usually in the form of a browser side script to a different end user. This can occur due to any flaw in a web application where user input is translated into an output and is not validated or encoded. With XSS an attacker can direct a malicious script to a user whose unsuspecting browser has no way to identify that the script can't be trusted and will be ran as it thinks the script is from a trusted origin.

The 3 types of XSS attacks are defined bellow as provided by the OWASP foundation site:

Reflected XSS [AKA Non-Persistent (Type I)]
"where user input is instantly returned by a web application in an error message, search result or any other response that includes some or all of the input provided by the user, without that data being made safe to render in the browser, and without permanently storing the user provided data."

Stored XSS [AKA Persistent (Type II)]
"generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc. And then a victim is able to retrieve the stored data from the web application without that data being made safe to render in the browser."

DOM Based XSS [AKA (Type 0)]
"DOM Based XSS (or as it is called in some texts, "type-0 XSS") is an XSS attack wherein the attack payload is executed as a result of modifying the DOM "environment" in the victim's browser used by the original client side script, so that the client side code runs in an "unexpected" manner. That is, the page itself (the HTTP response that is) does not change, but the client side code contained in the page executes differently due to the malicious modifications that have occurred in the DOM environment."

Tools used:

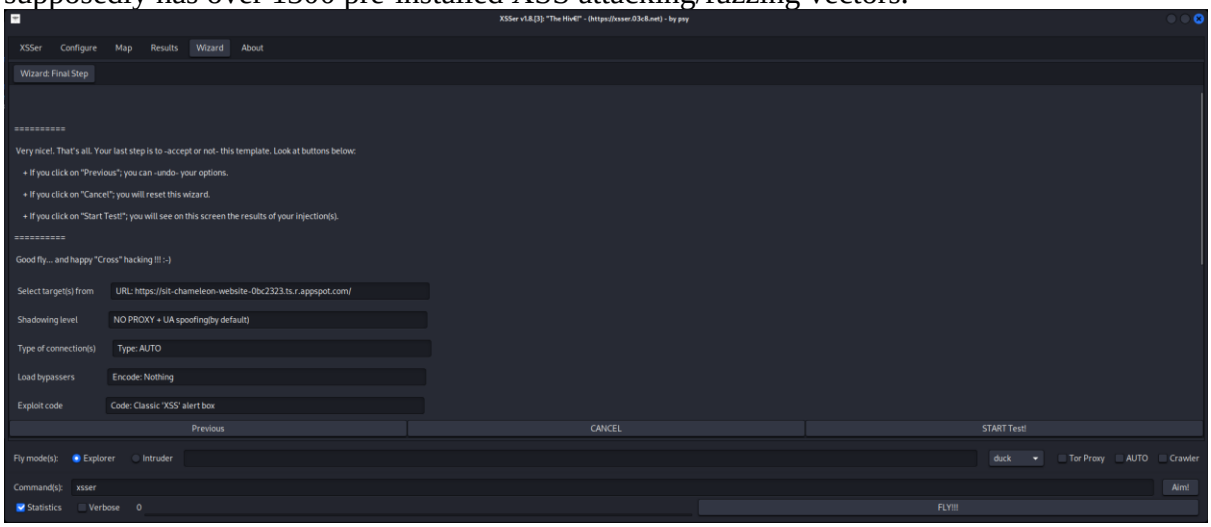
- Parrot OS
- Kali Linux
- XSSer
- OWASP ZAP
- XSS FREAK
- XSS Scanner
- XSS Strike

Scope

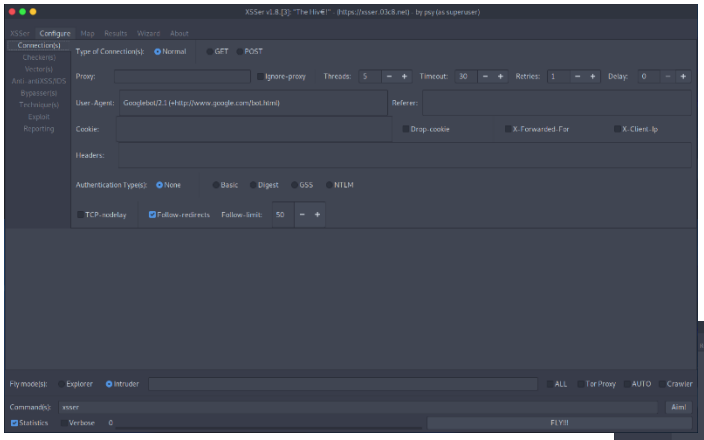
The scope of our testing will be exclusively the Chameleon Site, the tests will be done with no permissions or existing information known about how the site operates, no help from administrators or other users of the site and tests done purely as any other standard user, blind on the innerworkings.
Site: https://sit-chameleon-website-0bc2323.ts.r.appspot.com/

Methodology

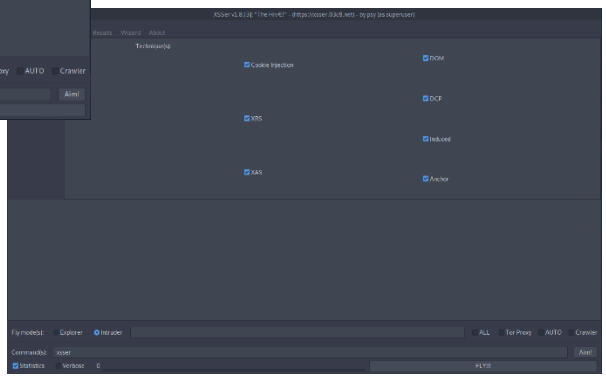
Firstly we are going to use a program called XSSer, which is their documentation states “is an automatic framework that is used to detect, exploit and report XSS vulnerabilities in web-based applications” allowing us to automatically throw every test in its database, which it supposedly has over 1300 pre-installed XSS attacking/fuzzing vectors.



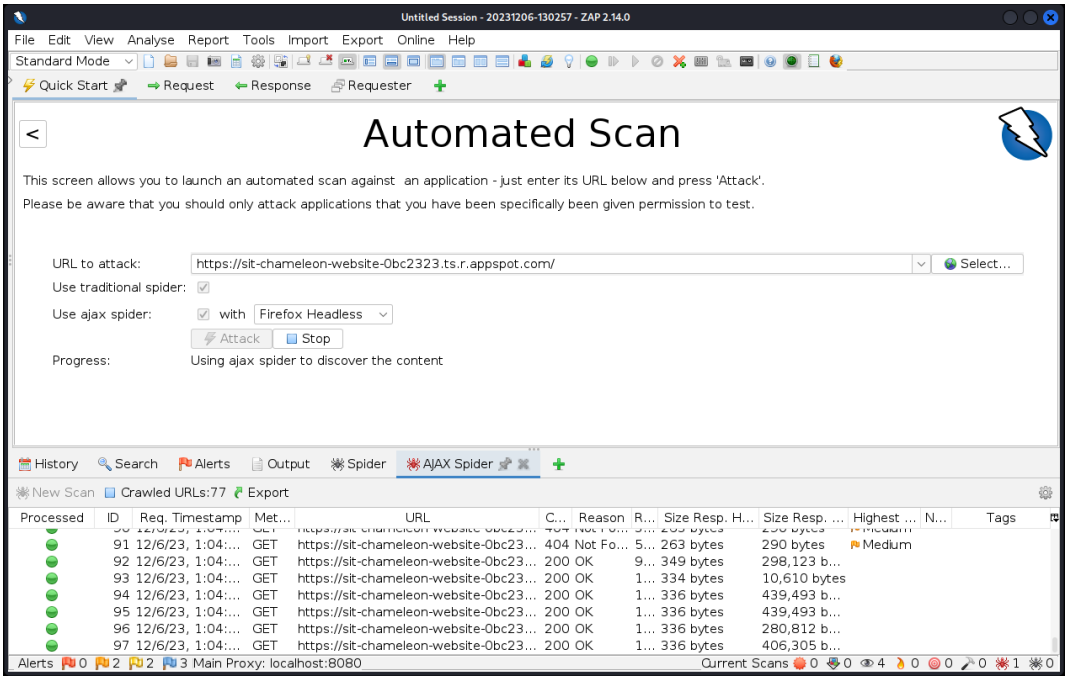
It also includes an easy-to-use wizard that automatically configures the testing based on our wishes, that of which since we don’t actually intend to inject any malicious code, but instead test for vulnerabilities and see if its capable of withstanding such an attack and therefore we will focus on identifying the vulnerabilities with a mass number of tests.



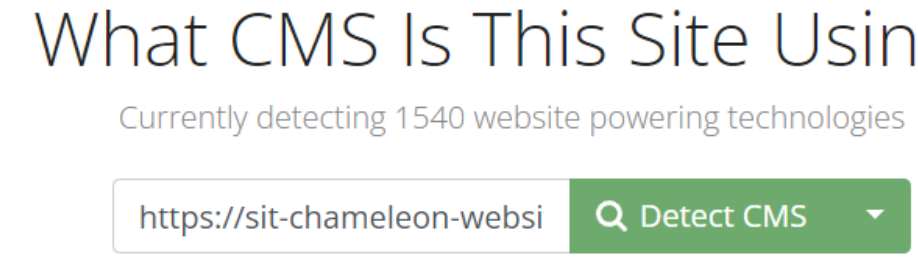
To make sure we are really getting the most out of the tool we will also manually do some tests, firstly we keep everything default on the connection menu except allowing the crawler to follow-redirects, but only 50 deep. We also specify we want to use all the different types of attacks.

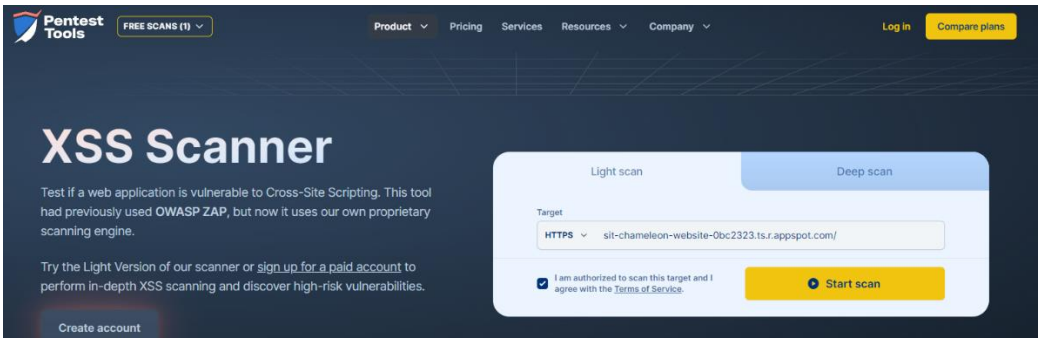


The next tool we will be using is the OWASP ZAP tool, to use its automated scan in an attempt to find vulnerabilities relating to XSS, additionally the tool also will do a spider and AJAX spider crawl, identifying resources that may be kept hidden on the site, but the results of those scans is addressed in my other report “Discovering Hidden Files & Vulnerabilities”.



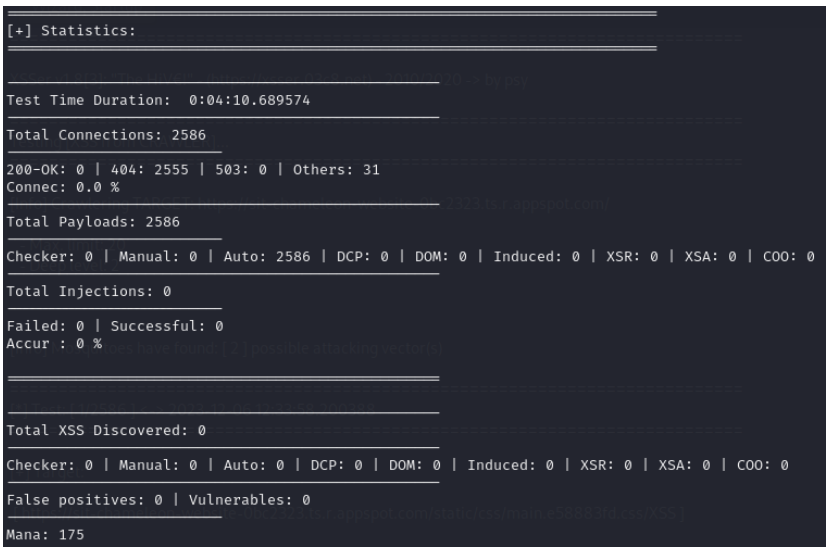
After that scan we will be using another tool previously used in the past trimester, a CMS scan site (https://whatcms.org/) attempting to identify what Content Management System the site uses which can help us narrow down any potential CMS scanners, depending on the technology found, such as using WPScan to hunt down vulnerabilities with WordPress websites or JoomScan if the site was using Joomla CMS. Which we will then follow up with another online tool, provided by pentest-tools.com, which is their website-vulnerability scanner, just to confirm our findings, lastly followed by a few more XSS tools.

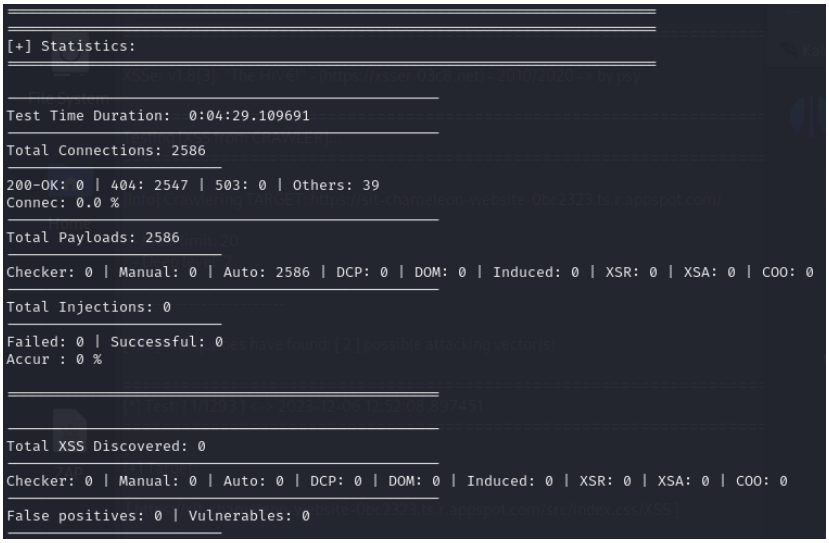




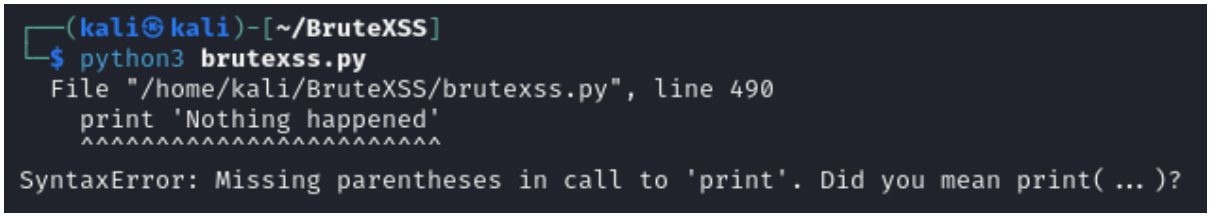
Results

So, after all these tests what was revealed? Well firstly the automated scans as well as the manually configured scans run by XSSer unfortunately bore no fruit, With both scans unable to identify any vulnerabilities in-regards to Cross-Site Scripting, I even did multiple tests with different configs as well as using third-party XSS payloads, but there was no change.





To pursue every avenue I also repeated the tests in other known XSS tools, such as XSS Freak, BruteXSS, xsscrapy and XSStrike, all after configuring them as well as attempting to use their default payloads list as well as the third-party list I have used before, but unfortunately majority of them were too old and outdated they didn't work on my virtual machines, probably also due to incorrect dependency versions, and the ones that did work didn't find anything.



```
┌──(kali㉿kali)-[~/xsscrapy]
└─$ ./xsscrapy.py -u https://sit-chameleon-website-0bc2323.ts.r.appspot.com
Traceback (most recent call last):
  File "/home/kali/xsscrapy/./xsscrapy.py", line 5, in <module>
    from xsscrapy.spiders.xss_spider import XSSspider
  File "/home/kali/xsscrapy/xsscrapy/spiders/xss_spider.py", line 9, in <module>
    from urlparse import urlparse, parse_qsl, urljoin, urlunparse, urlunsplit
ModuleNotFoundError: No module named 'urlparse'
```

```
┌──(kali㉿kali)-[~/XSStrike]
└─$ python xsstrike.py -u "https://sit-chameleon-website-0bc2323.ts.r.appspot.com" --crawl

        XSStrike v3.1.5

[~] Crawling the target
 !] Progress: 1/1
```

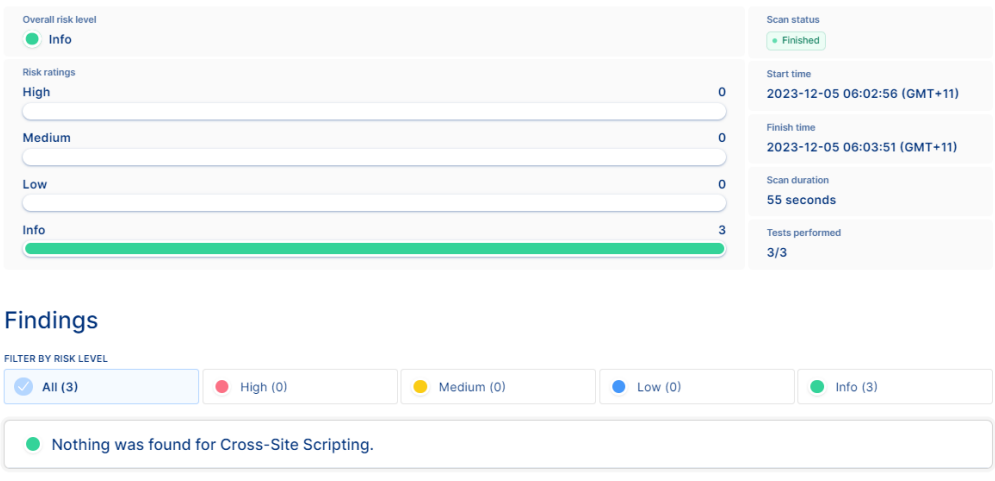
```
>> [+] Give Me A Target To Destroy
>> [?] Enter Target: https://sit-chameleon-website-0bc2323.ts.r.appspot.com/
>> [*] Enter File Containing XSS Payloads to Try: payloads.txt
>> [*] Searching Target For Possible Links And Directories

┌──(kali㉿kali)-[~/XSS-Freak]
└─$
```

Following the lack of results with the XSS programs I was hoping for some better results with the online services, but just like the rest there was a lack of results there too, with the XSS scanner returning with nothing and the CMS scanner failing to detect a CMS.

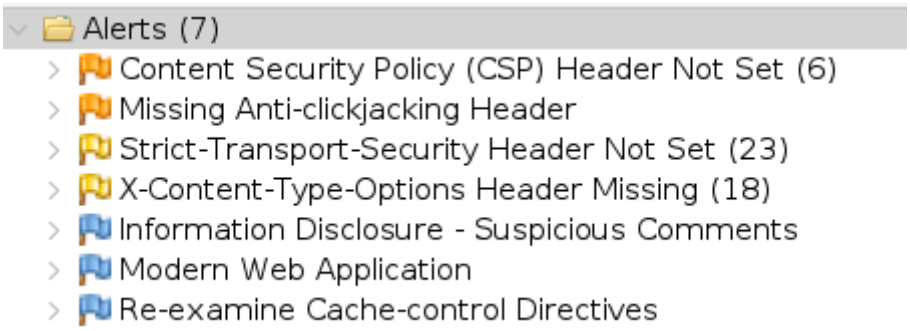
Sorry JSON

We couldn't detect a CMS at sit-chameleon-website-0bc2323.ts.r.appspot.com

Help us improve these results



Lastly came the OWASP ZAP tools report, which after running their active scan revealed what had practically been confirmed by the onslaught of tests run against the site, which is that there isn't any XSS vulnerabilities detected on the Chameleon Site.



While several other issues are present, which can result in more attack vectors for us as penetration testers to take advantage of, that is for a different report.

Conclusion & Recommendations

After the countless tests one thing has been made clear, that the Chameleon site, does not seem to be remotely vulnerable to Cross-Site Scripting attacks, which is great news, however it is not without its flaws as is seen in the ZAP report with other vulnerabilities existing instead and for that I would recommend looking towards my other report "Discovering Hidden Files & Vulnerabilities" where I address these alerts and what needs to be done about them.

References:

Cross site scripting (XSS) (no date) *Cross Site Scripting (XSS) | OWASP Foundation*. Available at: https://owasp.org/www-community/attacks/xss/ (Accessed: 06 December 2023).

Cross-site scripting (XSS) explained (2020) *YouTube*. Available at: https://youtu.be/EoaDgUgS6QA (Accessed: 06 December 2023).

DanMcInerney (no date) *Danmcinerney/xsscrapy: XSS spider - 66/66 WAVSEP XSS detected, GitHub*. Available at: https://github.com/DanMcInerney/xsscrapy (Accessed: 06 December 2023).

Payloadbox (no date) *Payloadbox/XSS-payload-list: Cross site scripting (XSS) vulnerability payload list, GitHub*. Available at: https://github.com/payloadbox/xss-payload-list?tab=readme-ov-file (Accessed: 06 December 2023).

Rajeshmajumdar (no date) *Rajeshmajumdar/BruteXSS, GitHub*. Available at: https://github.com/rajeshmajumdar/BruteXSS (Accessed: 06 December 2023).

s0md3v (no date) *S0MD3V/xsstrike: Most advanced XSS scanner., GitHub*. Available at: https://github.com/s0md3v/XSStrike (Accessed: 06 December 2023).

Types of XSS (no date) *Types of XSS | OWASP Foundation*. Available at: https://owasp.org/www-community/Types_of_Cross-Site_Scripting (Accessed: 06 December 2023).

Web app penetration testing - #10 - XSS(reflected, stored & dom) (2018) *YouTube*. Available at: https://youtu.be/SHmQ3sQFeLE (Accessed: 06 December 2023).

Web application ethical hacking - penetration testing course for beginners (2020) *YouTube*. Available at: https://youtu.be/X4eRbHgRawI (Accessed: 06 December 2023).

What CMS is this site using? (no date) *Detect which CMS a site is using - What CMS?* Available at: https://whatcms.org/ (Accessed: 06 December 2023).

XSS scanner - online scan for cross-site scripting vulnerabilities (no date) *Pentest*. Available at: https://pentest-tools.com/website-vulnerability-scanning/xss-scanner-online (Accessed: 06 December 2023).

Xsser: Kali linux tools (2022) *Kali Linux*. Available at: https://www.kali.org/tools/xsser/ (Accessed: 06 December 2023).