CHAMELEON

FOR OUR SMARTER WORLD

Cross-Site Scripting (MOP)[Failure]

Adam Sarin
217342706

Contents

Executive summary:

This report presents the findings of a Cross-Site Scripting (XSS) vulnerability assessment, in which last trimester as a junior I conducted on the Chameleon site and am now applying my methods on the MOP site, this report will reincluded my introduction into XSS from my previous report as well as replicate many of the methodologies used last report, but this time aimed at the MOP site.

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."

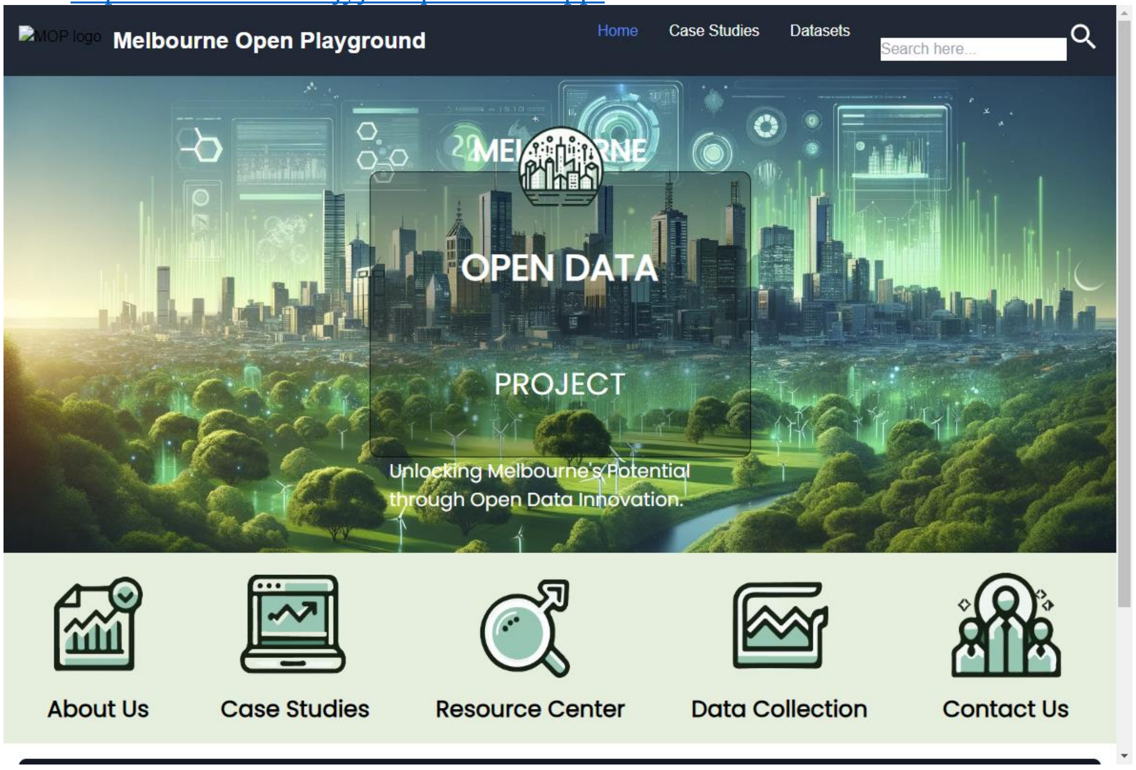
Tools used:

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

Scope

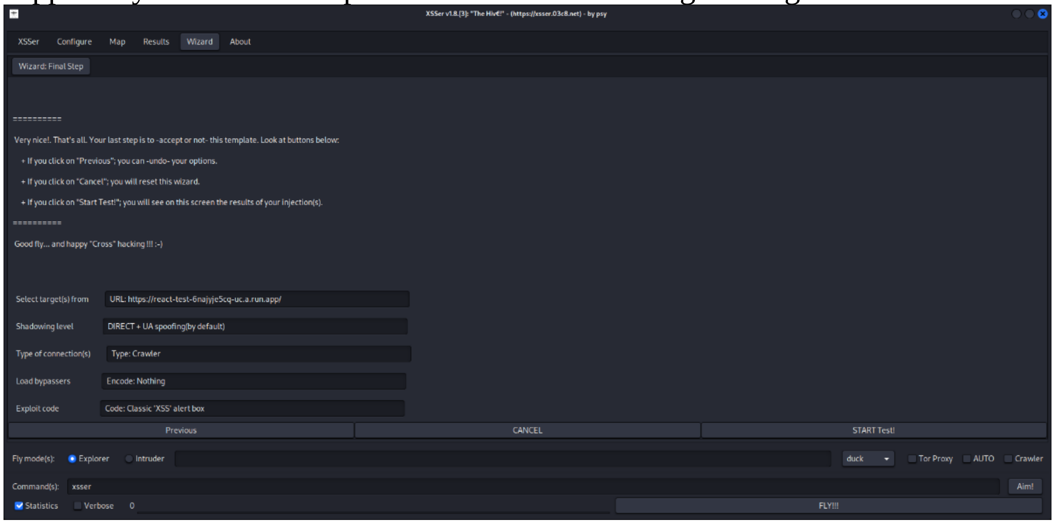
In-regards to the scope of our tests, we will be working exclusively with just the MOP site, with no GCP access I am unsure of the innerworkings of the sites security, as well as no information provided by any site administrators or others who might have access to the back-end. These tests will all be blind on what's happening under the hood and just be performed in a penetration testing manner, an image of the current state of MOP currently will be included below.

Site: https://react-test-6najyje5cq-uc.a.run.app/



Methodology

First we are going to use a program called XSSer, which in 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.

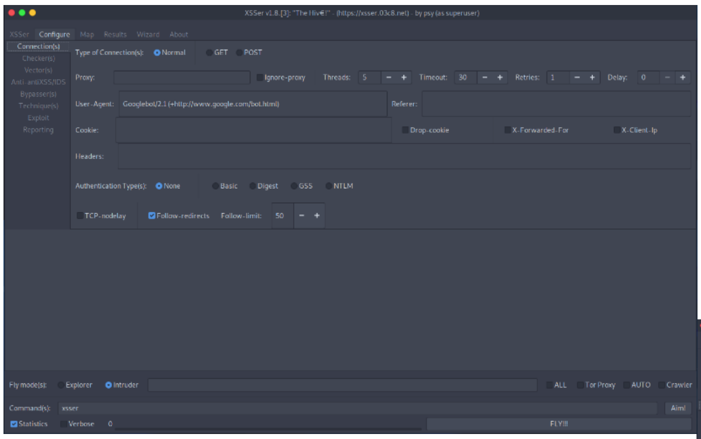


```
==========
Very nice!. That's all. Your last step is to -accept or not- this template. Look at buttons below:
  + If you click on "Previous"; you can -undo- your options.
  + If you click on "Cancel"; you will reset this wizard.
  + If you click on "Start Test!"; you will see on this screen the results of your injection(s).
==========
Good fly... and happy "Cross" hacking !!! :-)
```

Select target(s) from: URL: https://react-test-6najyje5cq-uc.a.run.app/
Shadowing level: DIRECT + UA spoofing(by default)
Type of connection(s): Type: Crawler
Load bypassers: Encode: Nothing
Exploit code: Code: Classic 'XSS' alert box

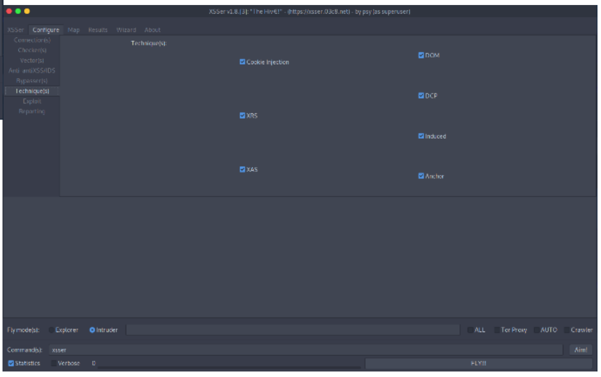
Fly mode(s): [✓ Explorer] Intruder; duck; ☐ Tor Proxy ☐ AUTO ☐ Crawler
Command(s): xsser
☑ Statistics ☐ Verbose 0

It also includes an easy-to-use wizard that automatically configures the testing based on our wishes, that of which since we don't 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.


XSSer v1.8.[3]: "The Hiv€!" - (https://xsser.03c8.net) - by psy (as superuser)

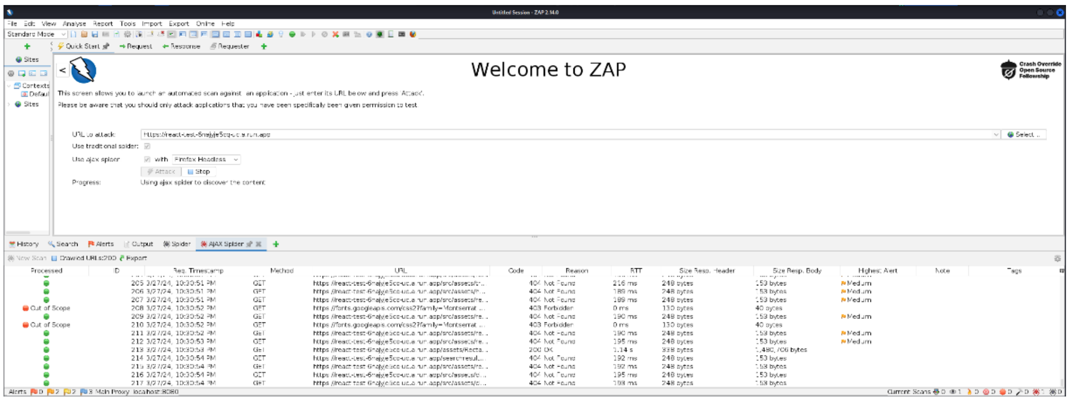
Type of Connection(s): Normal
Threads: 5 Timeout: 30 Retries: 1 Delay: 0
User-Agent: Googlebot/2.1 (+http://www.google.com/bot.html)
Authentication Type(s): None
Follow-redirects ✓ Follow-limit: 50
Fly mode(s): Intruder
Command(s): xsser
Statistics ✓ Verbose 0

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.


XSSer v1.8.[3]: "The Hiv€!" - (https://xsser.03c8.net) - by psy (as superuser)

Technique(s):
Cookie Injection ✓
DOM ✓
DCP ✓
XRS ✓
Induced ✓
XAS ✓
Anchor ✓
Fly mode(s): Intruder
Command(s): xsser
Statistics ✓ Verbose 0

The next tool we will be using is the OWASP ZAP tool, to use its automated scan 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 some of those scans are not relevant for our cross-site scripting attempts and will be addressed in another report.


Untitled Session - ZAP 2.14.0

Welcome to ZAP

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.
Please be aware that you should only attack applications that you have been specifically been given permission to test

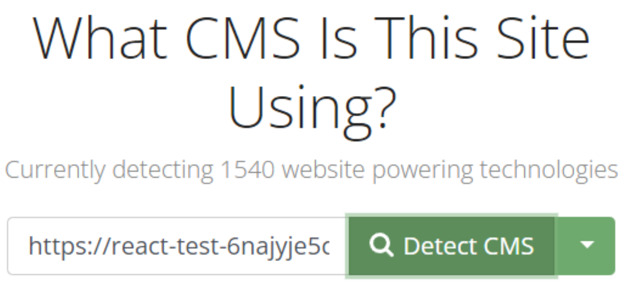
URL to attack: https://react-test-6najyje5cq-uc.a.run.app
Use traditional spider: ✓
Use ajax spider: ✓ with Firefox Headless
Progress: Using ajax spider to discover the content

Crawled URLs:200

```
Processed     ID  Req. Timestamp         Method  URL                                                   Code  Reason     RTT     Size Resp. Header  Size Resp. Body   Highest Alert
              205 3/27/24, 10:30:51 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/r...  404  Not Found  216 ms  248 bytes  153 bytes        Medium
              206 3/27/24, 10:30:51 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/r...  404  Not Found  189 ms  248 bytes  153 bytes        Medium
              207 3/27/24, 10:30:51 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/re...  404  Not Found  189 ms  248 bytes  153 bytes        Medium
Out of Scope  208 3/27/24, 10:30:52 PM   GET     https://fonts.googleapis.com/css2?family=Montserrat ...     403  Forbidden  0 ms    130 bytes  40 bytes
              209 3/27/24, 10:30:52 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/re...  404  Not Found  190 ms  248 bytes  153 bytes        Medium
Out of Scope  210 3/27/24, 10:30:52 PM   GET     https://fonts.googleapis.com/css2?family=Montserrat ...     403  Forbidden  0 ms    130 bytes  40 bytes
              211 3/27/24, 10:30:52 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/re...  404  Not Found  190 ms  248 bytes  153 bytes        Medium
              212 3/27/24, 10:30:53 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/re...  404  Not Found  195 ms  248 bytes  153 bytes        Medium
              213 3/27/24, 10:30:53 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/assets/React...   200  OK         1.14 s  338 bytes  1,480,706 bytes
              214 3/27/24, 10:30:54 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/searchresul...   404  Not Found  192 ms  248 bytes  153 bytes
              215 3/27/24, 10:30:54 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/re...  404  Not Found  192 ms  248 bytes  153 bytes
              216 3/27/24, 10:30:54 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/c...   404  Not Found  195 ms  248 bytes  153 bytes
              217 3/27/24, 10:30:54 PM   GET     https://react-test-6najyje5cq-uc.a.run.app/src/assets/c...   404  Not Found  193 ms  248 bytes  153 bytes
```

Main Proxy: localhost:8080

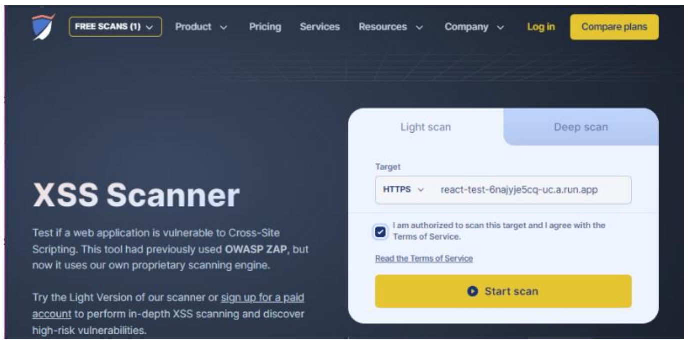
After that scan we will be using another tool previously used in the last trimester that I discovered from a previous report written by a capstone member, a CMS scan site (https://whatcms.org/) attempting to identify what Content Management System the site uses which can help us narrow down any potential 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.


What CMS Is This Site Using?

Currently detecting 1540 website powering technologies

https://react-test-6najyje5c

Detect CMS


FREE SCANS (1) Product Pricing Services Resources Company Log in Compare plans

XSS Scanner

Test if a web application is vulnerable to Cross-Site Scripting. This tool had previously used **OWASP ZAP**, but now it uses our own proprietary scanning engine.

Try the Light Version of our scanner or sign up for a paid account to perform in-depth XSS scanning and discover high-risk vulnerabilities.

Light scan | Deep scan

Target: HTTPS react-test-6najyje5cq-uc.a.run.app

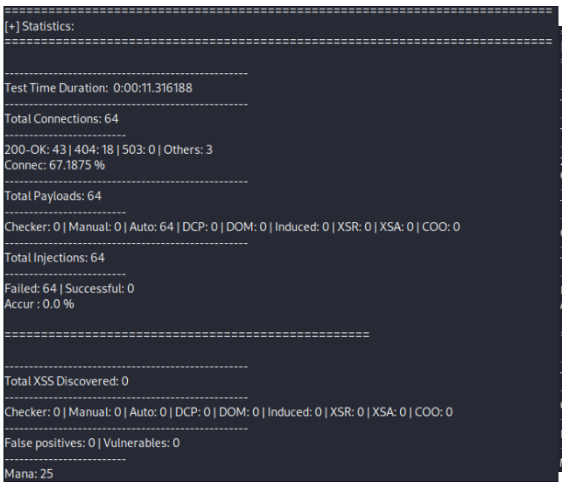
✓ I am authorized to scan this target and I agree with the Terms of Service.

Read the Terms of Service

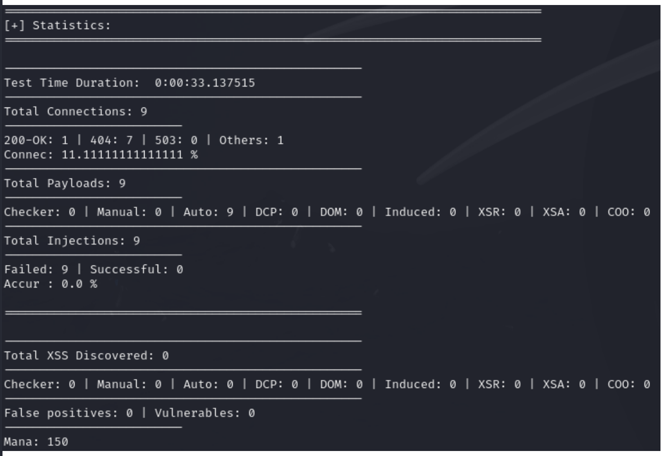
Start scan

Results

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

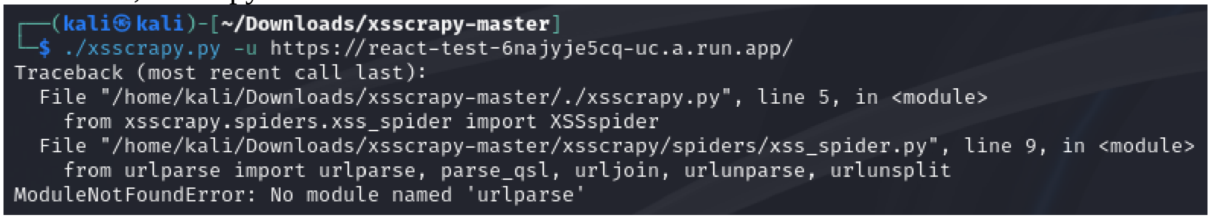


```
==========================================================================
[+] Statistics:
==========================================================================

-----------------------------------------------
Test Time Duration:  0:00:11.316188
-----------------------------------------------
Total Connections: 64
-----------------------
200-OK: 43 | 404: 18 | 503: 0 | Others: 3
Connec: 67.1875 %
-----------------------------------------------
Total Payloads: 64
-----------------------
Checker: 0 | Manual: 0 | Auto: 64 | DCP: 0 | DOM: 0 | Induced: 0 | XSR: 0 | XSA: 0 | COO: 0
-----------------------------------------------
Total Injections: 64
-----------------------
Failed: 64 | Successful: 0
Accur : 0.0 %

================================================

-----------------------------------------------
Total XSS Discovered: 0
-----------------------------------------------
Checker: 0 | Manual: 0 | Auto: 0 | DCP: 0 | DOM: 0 | Induced: 0 | XSR: 0 | XSA: 0 | COO: 0
-----------------------------------------------
False positives: 0 | Vulnerables: 0
-----------------------
Mana: 25
```

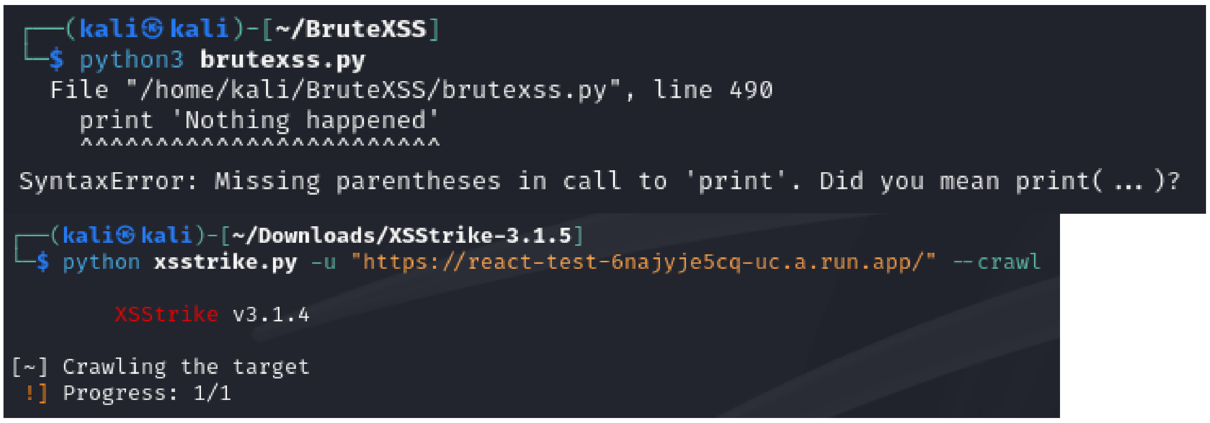


```
[+] Statistics:

Test Time Duration:  0:00:33.137515

Total Connections: 9

200-OK: 1 | 404: 7 | 503: 0 | Others: 1
Connec: 11.11111111111111 %

Total Payloads: 9

Checker: 0 | Manual: 0 | Auto: 9 | DCP: 0 | DOM: 0 | Induced: 0 | XSR: 0 | XSA: 0 | COO: 0

Total Injections: 9

Failed: 9 | Successful: 0
Accur : 0.0 %


Total XSS Discovered: 0

Checker: 0 | Manual: 0 | Auto: 0 | DCP: 0 | DOM: 0 | Induced: 0 | XSR: 0 | XSA: 0 | COO: 0

False positives: 0 | Vulnerables: 0

Mana: 150
```

To attempt every method, I also repeated the tests I used with Chameleon, using other known XSS tools, however either they found nothing, or like my previous report, I was unable to get them working, mostly due to their age and missing dependencies as well as a lack of documentation of people solving the issues with these programs, the tools used were BruteXSS, xsscrapy and XSStrike.



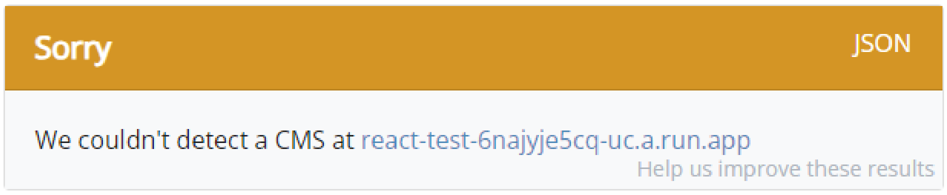
```
┌──(kali㉿kali)-[~/Downloads/xsscrapy-master]
└─$ ./xsscrapy.py -u https://react-test-6najyje5cq-uc.a.run.app/
Traceback (most recent call last):
  File "/home/kali/Downloads/xsscrapy-master/./xsscrapy.py", line 5, in <module>
    from xsscrapy.spiders.xss_spider import XSSspider
  File "/home/kali/Downloads/xsscrapy-master/xsscrapy/spiders/xss_spider.py", line 9, in <module>
    from urlparse import urlparse, parse_qsl, urljoin, urlunparse, urlunsplit
ModuleNotFoundError: No module named 'urlparse'
```



```
┌──(kali㉿kali)-[~/BruteXSS]
└─$ python3 brutexss.py
  File "/home/kali/BruteXSS/brutexss.py", line 490
    print 'Nothing happened'
    ^^^^^^^^^^^^^^^^^^^^^^^^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
```

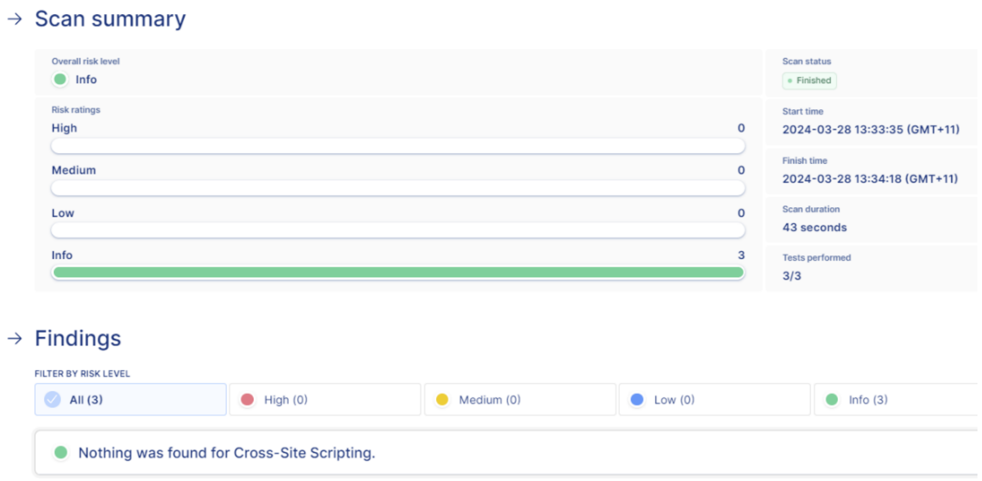
```
┌──(kali㉿kali)-[~/Downloads/XSStrike-3.1.5]
└─$ python xsstrike.py -u "https://react-test-6najyje5cq-uc.a.run.app/" --crawl

        XSStrike v3.1.4

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

After lacking results with the Cross Site Scripting programs, I had hoped for some better news using the online services, however those services could not find anything that the other programs might of missed, providing a lack of results as the CMS scan was unable to detect a CMS and the XSS scanner provided by pentest-tools.com returning no findings.



Sorry JSON

We couldn't detect a CMS at react-test-6najyje5cq-uc.a.run.app
Help us improve these results



→ Scan summary

Overall risk level
● Info

Risk ratings

High	0
Medium	0
Low	0
Info	3

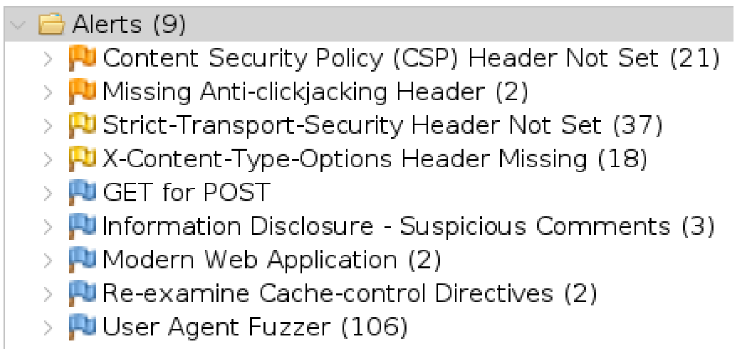
Scan status	Finished
Start time	2024-03-28 13:33:35 (GMT+11)
Finish time	2024-03-28 13:34:18 (GMT+11)
Scan duration	43 seconds
Tests performed	3/3

→ Findings

FILTER BY RISK LEVEL
All (3) | High (0) | Medium (0) | Low (0) | Info (3)

● Nothing was found for Cross-Site Scripting.

Then at last came the OWASP ZAP tools report, which showed using their active scan what had been seen with the previous tests run against MOP, which is that there aren't any obvious XSS vulnerabilities that we can take advantage of. While several other minor issues are present, which could result in more attack paths for a bad actor to take advantage of, that is for a different report as they are not helping us in-regards to XSS


Alerts (9)
- Content Security Policy (CSP) Header Not Set (21)
- Missing Anti-clickjacking Header (2)
- Strict-Transport-Security Header Not Set (37)
- X-Content-Type-Options Header Missing (18)
- GET for POST
- Information Disclosure - Suspicious Comments (3)
- Modern Web Application (2)
- Re-examine Cache-control Directives (2)
- User Agent Fuzzer (106)

Conclusion & Recommendations

Similar to my report on the Chameleon site, the MOP site does not seem to have any Cross-Site Scripting vulnerabilities from my own testing, however unlike Chameleon it actually may be more vulnerable to different types of attacks as shown with the number of flaws detected by OWASP, The Chameleon site did have a few issues flagged here but not as many as MOP, however at least most of them are minor issues that can be easily addressed and don't result in large attack vectors at all.

References:

Cross-site scripting (FAILED) (no date) | *Adam Sarin*.

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).