


XSSStrike and Cypress: Finding XSS Vulnerabilities, Testing, and Safe Your Web Apps.

 medium.com/@elyassinisafouat2/xsstrike-and-cypress-finding-xss-vulnerabilities-testing-and-safe-your-web-apps-84e0cdc51afc

Safouat El Yassini

October 16, 2023



Whether you're a developer or a QA tester, ensuring your website's safety is crucial. Even if you use security measures like a Content Security Policy, you must actively test and protect your site from XSS attacks. In this article, I'll introduce you to some tools that can help you discover malicious payloads. You'll also learn how to use these payloads with Cypress to figure out which one is harmful and where it can sneak into your website.

What is XSS:

XSS (Cross-Site Scripting) is when bad actors inject harmful javascript code into a website. The tricky part is, the website's server can't always tell if the code is from a trusted or untrusted source. With XSS, they can insert code that can swipe data from cookies and sessions, or even plant keyloggers to capture keystrokes. It's like inviting a spy into your web world without realizing it. there is two type of XSS attack :

- In this type of XSS , the injected malicious script is permanently stored on the web server, such as in a database, and it is then retrieved and executed when a user visits a specific page that displays the stored content. This means that any user who accesses the page is at risk of running the malicious script.

- In this type of XSS attack, the malicious script is embedded in a URL or a web form input, and it is reflected off a web server to the victim's browser. The victim clicks on a link or submits a form with the injected script, and the server sends the script back as part of the response.

Stored XSS is often more dangerous than reflected XSS because it can affect a broader range of users and persists even after the attacker has injected the script.

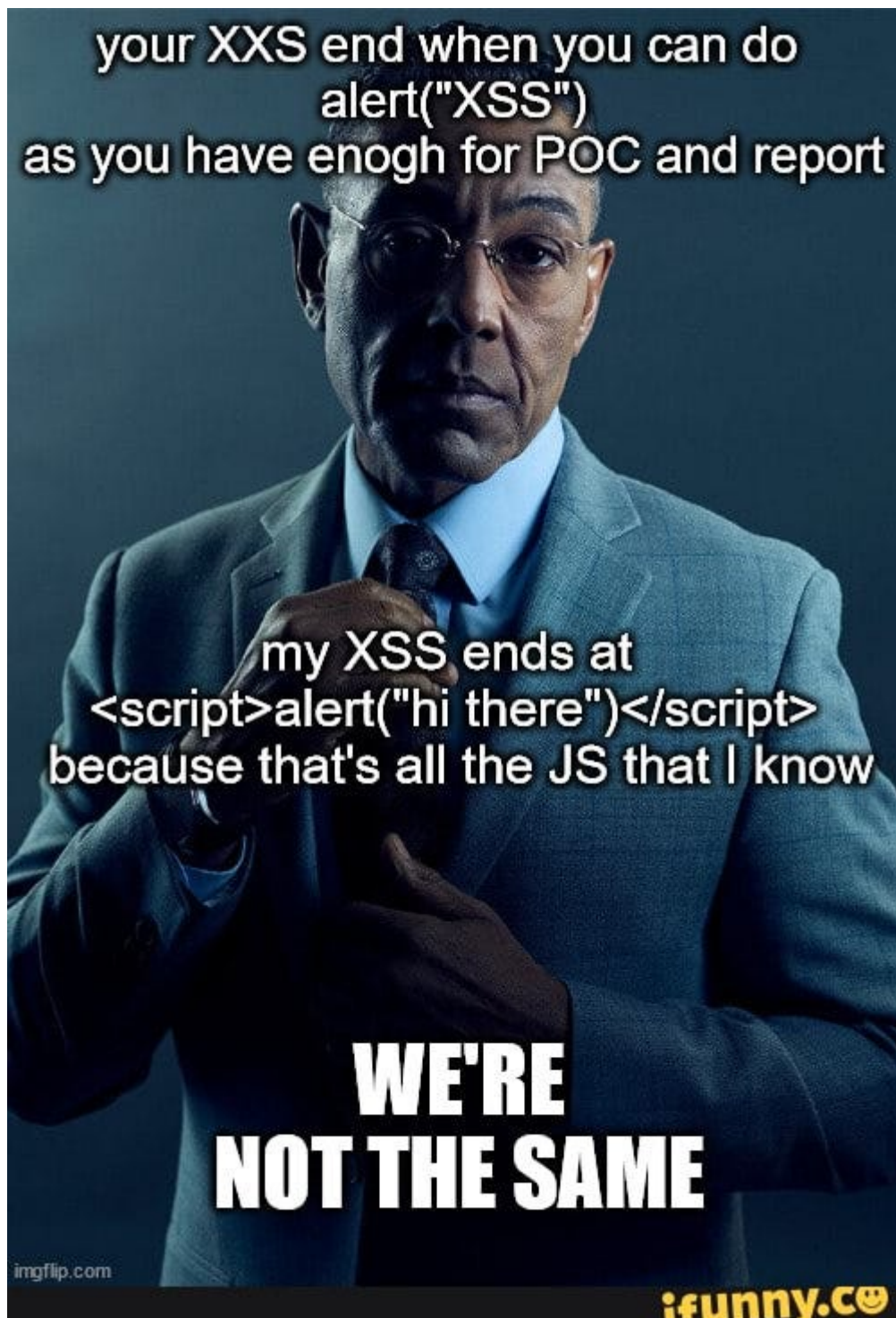
A simple example of Stored XSS a payload :

how can i find Payloads:

You can discover XSS payloads using different methods:

The traditional approach involves manually inspecting your browser and experimenting to see if you can inject a payload into the HTML or JavaScript code of a web page.

However, this method can be time-consuming, may not cover all possible vulnerabilities, and often requires a solid understanding of JavaScript and web development to be effective.



A more efficient way is to use a specialized scanner tool designed to detect XSS vulnerabilities by automatically injecting various payloads and analyzing the application's responses. These tools can help you identify potential weaknesses in your web application more quickly and comprehensively, reducing the manual effort required for testing. for example **XSSStrike**

What is XSSStrike:



is an open-source advanced XSS (Cross-Site Scripting) detection and exploitation framework. It is designed to help security professionals, ethical hackers, and penetration testers identify and exploit XSS vulnerabilities in web applications.

Key features of XSStrike include:

Payload Generation: XS-Strike provides a range of payload templates and encoding techniques, making it easier to craft payloads to test for XSS vulnerabilities.

Detection: The tool automates the process of detecting and identifying XSS vulnerabilities in web applications by injecting various payloads and analyzing the responses.

Exploitation: XS-Strike can be used to confirm the presence of an XSS vulnerability and even exploit it, allowing testers to demonstrate the impact of the vulnerability.

Advanced Techniques: XS-Strike includes support for a variety of advanced techniques, such as DOM-based and blind XSS detection, which can be particularly challenging to identify.

Reporting: The tool generates detailed reports that can be used to document and communicate the results of XSS testing to development and security teams.

Join Medium for free to get updates from this writer.

Setup and installation:

```
git https://github.com/s0md3v/XSStrike.git XSStrikepip install -r requirements.txtpython3 xsstrike.py -u
```

Let's try using XS-Strike on a website, but not on an official website; I don't like lentils.

i will use this [website sudo.co.il](https://www.sudo.co.il) for finding the payloads :

```

safouat@safouat-HP-ProBook-x360-435-G8:~$ cd XSSStrike
safouat@safouat-HP-ProBook-x360-435-G8:~/XSSStrike$ python3 xssstrike.py -u "http://www.sudo.co.il/xss/level6.php?p=your_payload"

XSSStrike v3.1.5

[~] Checking for DOM vulnerabilities
[+] WAF Status: Offline
[!] Testing parameter: p
[!] Reflections found: 1
[~] Analysing reflections
[~] Generating payloads
[!] Payloads generated: 5

-----
[+] Payload: \';(prompt)``//
[!] Efficiency: 96
[!] Confidence: 6
[?] Would you like to continue scanning? [y/N] y
-----
[+] Payload: \';[8].find(confirm)//
[!] Efficiency: 97
[!] Confidence: 6
[?] Would you like to continue scanning? [y/N] y
-----
[+] Payload: \';confirm()//
[!] Efficiency: 96
[!] Confidence: 6
[?] Would you like to continue scanning? [y/N] y
-----
[+] Payload: \';(a=prompt,a())//
[!] Efficiency: 96
[!] Confidence: 6
[?] Would you like to continue scanning? [y/N] y
-----
[+] Payload: \';(confirm())//
[!] Efficiency: 96
[!] Confidence: 6
[?] Would you like to continue scanning? [y/N] y

safouat@safouat-HP-ProBook-x360-435-G8:~/XSSStrike$

```

1-Payloads of our Target

As we can see, **XSSStrike** provides us with possible payloads to test on the website. Now, we should proceed to test these payloads in **Cypress**, especially when dealing with numerous inputs, as it's essential to assess all the payloads in various DOM elements.

Be carefulBe carefull

XSSStrike may not only provide payloads but also identify potential vulnerabilities in your **JavaScript code** that should be avoided. In such cases, the tool might help you recognize weak points in your code or risky practices that could lead to XSS vulnerabilities, even if it doesn't generate payloads for you. It's essential to address these vulnerabilities and strengthen the security of your web application to prevent potential attacks.

For example, on this official website, I discovered a vulnerable script that could potentially be exploited by a malicious attacker.


```
XSStrike v3.1.5

[~] Checking for DOM vulnerabilities
[+] Potentially vulnerable objects found

-----
39         eval(onload_functions[i]);
47         eval(onunload_functions[i]);
-----

[-] No parameters to test.
```

Eval is evil. — Some frustrated JavaScript engineer



Cypress is an open-source end-to-end testing framework for web applications. It is designed to simplify and improve the process of testing web applications by providing a powerful and developer-friendly testing environment.

setup and instalation:

```
npm install cypress --save-dev npx cypress open
```

in your `todo.file.js` and by using the 'payloads for our target' on the website, we don't have any input fields, which means that the only possible payload insertion point is in the URL.

```
(, {   targetUrl = ;   (, {   cy.(targetUrl);   payloads = [   ,
];   payloads.( {   xssUrl = ;   cy.(xssUrl);
cy.(, {   (confirmText) {   (confirmText)..();   }
});   });   });});
```

spec cy.js 00:13

```

1  visit http://www.sudo.co.il/xss/level6.php?
    p=your_payload

2  visit http://www.sudo.co.il/xss/level6.php?
    p=your_payload?param=';confirm()//

    (xhr) ● POST 200 https://www.google-
    analytics.com/j/collect?
    v=1&_v=j101&a=716930964&t=pageview&_s=1&dl=htt
    p%3A%2F%2Fwww.sudo.co.il%2F%2F%2Flevel6.php%3F
    p%3Dyour_payload&ul=en-us&de=UTF-8&sd=24-
    bit&sr=1920x1080&vp=1000x660&je=0&_u=IEBAAEABA
    AAAACAAI~&jid=1621986464&gjid=1434885182&cid=2
    051097459.1697493829&tid=UA-66662547-
    1&_gid=448807772.1697493829&_r=1&_slc=1&z=1058
    273874

    (confirm) http://www.sudo.co.il/xss/level6.ph
    p?p=your_payload?param=';confirm()//

3  visit http://www.sudo.co.il/xss/level6.php?
    p=your_payload?param=';(confirm)()//

    (xhr) ● POST 200 https://www.google-
    analytics.com/j/collect?
    v=1&_v=j101&a=546559311&t=pageview&_s=1&dl=htt
    p%3A%2F%2Fwww.sudo.co.il%2F%2F%2Flevel6.php%3F
    p%3Dyour_payload%3Fparam%3D%255C%2527%3Bconfir
    m()%2F%2F&ul=en-us&de=UTF-8&sd=24-
    bit&sr=1920x1080&vp=1000x660&je=0&_u=AACAAEABA
    AAAACAAI~&jid=&gjid=&cid=2051097459.1697493829
    &tid=UA-66662547-
    1&_gid=448807772.1697493829&_slc=1&z=297083236

    (confirm) http://www.sudo.co.il/xss/level6.ph
    p?p=your_payload?param=';(confirm)()//

    (xhr) ● POST 200 https://www.google-
    analytics.com/j/collect?
    v=1&_v=j101&a=1011787642&t=pageview&_s=1&dl=ht
    tp%3A%2F%2Fwww.sudo.co.il%2F%2F%2Flevel6.php%3
    Fp%3Dyour_payload%3Fparam%3D%255C%2527%3B(conf
    irm)()%2F%2F&ul=en-us&de=UTF-8&sd=24-
    bit&sr=1920x1080&vp=1000x660&je=0&_u=AACAAEABA
    AAAACAAI~&jid=&gjid=&cid=2051097459.1697493829
    &tid=UA-66662547-
    1&_gid=448807772.1697493829&_slc=1&z=122852594
    3
```

As we can see all of the payloads are being executed on the target page, and the results of these executions are being captured by the test script for further analysis.

- `(confirm)http://www.sudo.co.il/xss/level6.php?p=your_payload?param=';confirm()//:`
- This line indicates that the payload was injected into the URL, and it triggered a JavaScript `confirm()` dialog.

in case of having multiple inputs:

```
(, { targetUrl = ; (, { cy(targetUrl); payloads = [ , ,  
, ]; payloads.( { cy().( { cy(input).  
(payload); cy.(, { (alertText)..  
()); }); }); }); });
```

you can find all of this codes in my [GitHub](#).

Conclusion:

the combination of XSSStrike and Cypress empowers web developers and security professionals to conduct thorough and systematic XSS testing, allowing for the timely discovery and mitigation of vulnerabilities before they can be exploited by malicious actors.