Securing Web Applications with FFP (FAUST, flagbot, polygl0ts) Javascript and XSS

Moritz Schneider²

In collaboration with 1 FAUST (ctf@fsi.informatik.uni-erlangen.de), 2 flagbot (ctf@vis.ethz.ch) and 3 polygl0ts (ctf@epfl.ch)

October 9, 2021







Table of Contents

What is XSS?

Types of XSS

Mitigations

Workshop challenges

What is XSS?





What is XSS?

- > XSS stands for *cross-site scripting*.
 - type of attack where a malicious user can inject scripts into a web page.







October 9, 2021

What is XSS?

- ► XSS stands for *cross-site scripting*.
 - type of attack where a malicious user can inject scripts into a web page.
- ► Why is it dangerous?
 - Can steal session cookies (if cookie not httpOnly)
 - Change content on the web page
 - redirect user
 - Log every interaction on page
 - make requests











October 9, 2021

- Reflected XSS
 - Payload is store in a URL parameter and "reflected" on the page.





October 9, 2021

- Reflected XSS
 - Payload is store in a URL parameter and "reflected" on the page.

Demo

Simple reflected XSS







- Persisted XSS
 - Payload is stored on the server
 - Common examples: Comments, Avatar





- Persisted XSS
 - Payload is stored on the server
 - Common examples: Comments, Avatar

Demo

Guestbook





In both cases

Client or server did not properly sanitize the input.





Mitigations





October 9, 2021

Mitigation

- ► Sanitize user input
 - ▶ Use innerText instead of innerHTML
 - Escape template variables on the server (usually default behaviour)
 - don't just concatenate HTML
- ► Use CSP (Content Security Policy)





► The Content Security Policy specifies in a header (or rarely also a <meta> tag) what resources can be loaded and executed.

```
Content-Security-Policy: script-src self;
```

Will only allow scripts loaded from the same origin and disallow inline scripts like or

```
<script>alert("XSS")</script>
```



CSP

► The Content Security Policy specifies in a header (or rarely also a <meta> tag) what resources can be loaded and executed.

```
Content-Security-Policy: script-src self;
```

- Will only allow scripts loaded from the same origin and disallow inline scripts like or <script>alert("XSS")</script> .
- Can also be used to restrict style loading, fonts, images, etc.
- Can automatically report violations



CSP Bypasses

- ▶ Often CSP is too lax and can be bypassed.
 - ► CDNs
 - JSONP endpoints
 - Certain frameworks

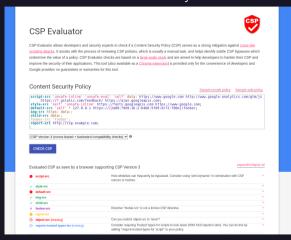






CSP Bypasses

► There is a tool to check your CSP¹:





Workshop challenges





Hands on challenges

- ► Challenge 1
 - Reflected XSS
 - Steal the admin session
 - Get the email
- ► Challenge 2
 - Persisted XSS
 - Steal the admin session
 - Read the guestbook entry
- ► Challenge 3
 - Exfiltrate the path of the image that only admin can view
 - Session cookie is httpOnly!
 - extract HTML-text
 - beware: innerHTML doesn't execute <script> Tags
 - but you can use event handlers





XSS Helper

- ► To have an admin visit a page, submit it on /xss-helper
- ► You can use the "Cookie Jar" for exfiltration
- ▶ Setting a cookie can be done via the dev tools





Let's gooooo

► Tasks: http://ffp-ctf.0x4141.net/





More resources

- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/script-src
- https://book.hacktricks.xyz/pentesting-web/content-security-policy-csp-bypass
- ► https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XSS



