Cyber Sapiens Internship Task 9

Cross Site Scripting

What is Cross-site scripting?

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

Types of Cross-site Scripting attacks

1. **Reflected XSS**
2. **Stored XSS**
3. **DOM-based XSS**
4. **Blind XSS**

1. Reflected XSS

Reflected XSS attacks, also known as non-persistent attacks, occur when a malicious script is reflected off of a web application to the victim's browser. The script is activated through a link, which sends a request to a website with a vulnerability that enables execution of malicious scripts.

2. Stored XSS

Stored XSS, also known as persistent XSS, is the more damaging of the two. It occurs when a malicious script is injected directly into a vulnerable web application. Reflected XSS involves the reflecting of a malicious script off of a web application, onto a user's browser.

3. DOM-based XSS

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.

4. Blind XSS

Blind XSS vulnerabilities are a variant of persistent XSS vulnerabilities. They occur when the attacker input is saved by the web server and executed as a malicious script in another part of the application or in another application.

How to Find this Bug?

The vast majority of XSS vulnerabilities can be found quickly and reliably using Burp Suite's web vulnerability scanner.

Manually testing for reflected and stored XSS normally involves submitting some simple payload into every parameter, identifying every location where the submitted input is returned in HTTP responses, and testing each location individually to determine whether suitably crafted input can be used to execute arbitrary JavaScript.

Manually testing for DOM-based XSS arising from URL parameters involves a similar process. placing some simple unique input in the parameter, using the browser's developer tools to search the DOM for this input, and testing each location to determine whether it is exploitable.

Impact of cross-site scripting

An XSS attack can probably lead to

- Cookie Stealing
- Account Hijacking
- Data Leakage
- Session Hijacking
- Browser Hooking

How to prevent cross-site scripting attacks

1. Sanitizing Inputs
2. Use HTTPOnly cookie flag
3. X-XSS-Protection Header
4. Implement Content Security Policy

References

https://owasp.org/www-community/attacks/xss/
https://portswigger.net/web-security/cross-site-scripting
https://www.acunetix.com/websitesecurity/cross-site-scripting/
https://developer.mozilla.org/en-US/docs/Glossary/Cross-site_scripting
https://www.techtarget.com/searchsecurity/definition/cross-site-scripting
https://www.synopsys.com/glossary/what-is-cross-site-scripting.html
https://www.acunetix.com/websitesecurity/detecting-blind-xss-vulnerabilities/
https://xsshunter.com/
https://r0x4r.medium.com/introduction-to-blind-xss-417dcf9c842c