

**ASSESSMENT REPORT**

**Week 3 Penetration Testing Report**

October 27, 2021

YELI BUONYA

Penetration Testing Report

# 1. INTRODUCTION

This report document hereby describes the proceedings and results of a Black Box security assessment conducted against the *Week 3 Labs*. The report hereby lists the findings and corresponding best practice mitigation actions and recommendations.

# 2. OBJECTIVE

The objective of the assessment was to uncover vulnerabilities in the *Week 3 Labs*and provide a final security assessment report comprising vulnerabilities, remediation strategy and recommendation guidelines to help mitigate the identified vulnerabilities and risks during the activity.

# 3. ASSESSMENT SCOPE SUMMARY

Outlined is a Black Box Application Security Assessment for the *Week 3 Labs.*

**Engagement Timeframe**

10/22/2021 – 10/27/2021

This section defines the scope and boundaries of the project.

|  |  |
| --- | --- |
| **Application Name** | Cross-Site Scripting |

# 

**RISK CALCULATION AND CLASSIFICATION**

|  |  |  |
| --- | --- | --- |
| **P1** | **Critical** | Vulnerabilities that cause a privilege escalation from unprivileged to  admin or allow for remote code execution, financial theft, etc. |
| **P2** | **High** | Vulnerabilities that affect the security of the software and impact the processes it supports. |
| **P3** | **Medium** | Vulnerabilities that affect multiple users and require little or no user interaction to trigger. |
| **P4** | **Low** | Vulnerabilities that affect singular users and require interaction or significant prerequisites to trigger (MitM) to trigger. |
| **P5** | **Info** | Non-exploitable vulnerabilities in functionality. Vulnerabilities that are by design or are deemed acceptable business risk to the customer. |

Total number of labs: 11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Critical** | **High** | **Medium** | **Low** | **Info** |
| 1 | 1 | 9 | 0 | 0 |

## 

## 

## 

## 

Source: https://www.bugcrowd.com/glossary/vulnerability-priority/

## SERVICE DESCRIPTION

## 

Penetration Testing is the process of simulating real-world attacks by using the same techniques as malicious hackers.

##### Black Box Penetration Testing

Also referred to as a Dynamic Application Security Testing (DAST). Black Box Penetration Testing is a method to assess the security level of an organization by simulating an attack which a malicious threat actor might undertake to exploit weaknesses in the target network and applications.

Black Box Penetration Testing is performed without any prior knowledge of the organization system, network, or applications. Unlike an attacker, however, we stop our test before exposing sensitive data or doing harm to your environment.

## CAMPAIGN OBJECTIVES

## 

The objectives of Black Box Penetration Testing are:

To simulate a real-world hacking scenario to test the strength of current security defenses and countermeasures.

To harden the perimeter of your environment (firewall, VPN, etc.) as well as any external services that may be exposed to the internet (cloud infrastructure, DMZ, and email services).

Validate the configurations of Information Technology (IT) Assets and produce a list of known vulnerabilities present in systems and applications and mitigate before they can be exploited by adversaries.

Provide a detailed report on each security vulnerability and suggest remediation guidelines for each security issue.

# LABS

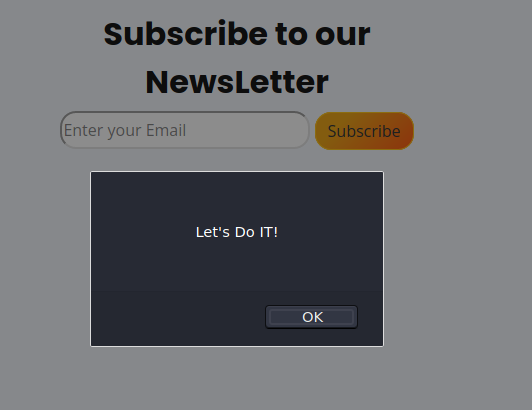
## 1. Cross-Site Scripting Labs

### 1.1. Let's Do IT!

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_basic\_injection\_01 | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| Reflected XSS found in the URL “email” parameter subscription input field. | |
| **Payload** | |
| <script>alert("Let's Do IT!")</script> | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_1/lab\_1.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers | |
| **References** | |
| https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### 

#### Proof of Concept



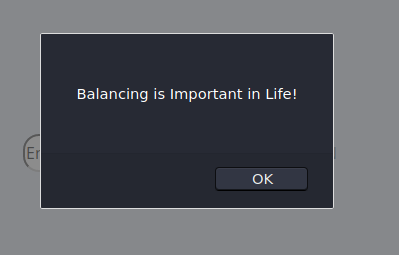
Reflected XSS Injection Payload

### 1.2. Balancing is Important in Life!

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xxs\_balanced\_payload\_01 | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| Reflected XSS found in the URL “email” parameter subscription input field. | |
| **Payload** | |
| ">N7RUZN<script>alert("Balancing is Important in Life!");</script> | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_2/lab\_2.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### 

#### Proof of Concept

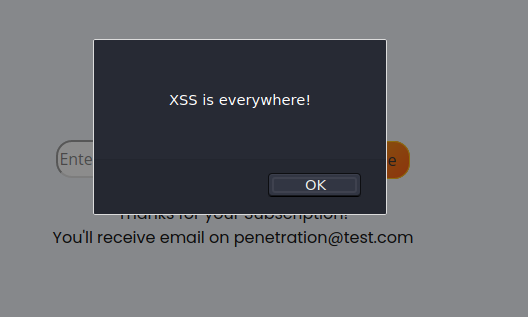


Reflected XSS using Balanced Payload

### 1.3. XSS is Everywhere!

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_email\_injection\_01 | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| Reflected XSS found in the URL “email” parameter subscription input field. | |
| **Payload** | |
| penetration@test.com<script>alert("XSS is everywhere!")</script> | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_3/lab\_3.php? | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### Proof of Concept



Reflected XSS in Email Parameter

### 1.4. Alternatives Are a Must!

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_alternatives\_01 | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| Reflected XSS found in the URL “email” parameter subscription input field. | |
| **Payload** | |
| ">penetration@test.com<script>prompt("Alternatives are must!")</script>"> | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_4/lab\_4.php? | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content to users and such as malware. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### Proof of Concept

# 

Alternate XSS Injection Payload using `Prompt`

### 1.5. Developers Hate Scripts!

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| weak\_xss\_sanitization\_01 | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| Weak sanitization mechanism that modifies the script tag from <script> to <scr\_ipt> can be bypassed by using the <img> tag. | |
| **Payload** | |
| ">penetration@test.com<img src/onerror=alert("Developers\_Hate\_Scripts")>"> | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_5/lab\_5.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content to users and such as malware. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### 

#### Proof of Concept

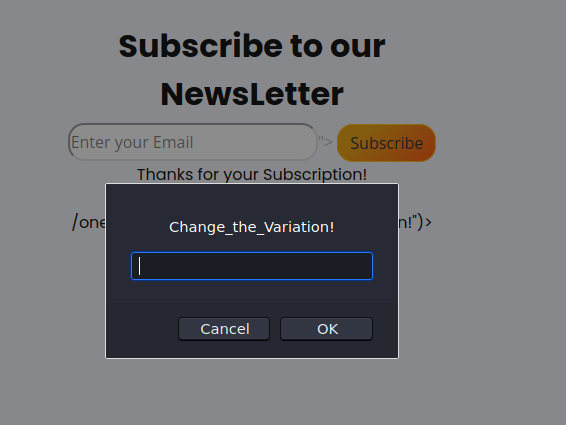
# 

Weak Input Filter Bypass using <img> Tag

### 1.6. Change the Variation!

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_script\_tag\_filter\_bypass\_01 | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| Bypass the `<script>` tag filter by using the `<img>` tag to perform a reflected XSS attack. | |
| **Payload** | |
| "><img/src/onerror=prompt("Change\_the\_Variation!")> | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/html\_lab/lab\_6/html\_injection\_6.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content to users and such as malware. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### Proof of Concept



URL Encoded Search Filter Bypass

### 

### 1.7. Encoding is the Key?

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_url\_encoded\_bypass\_01 | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| XSS Bypass using URL encoded payload. | |
| **Payload** | |
| %22%3E%3Cscript%3Ealert%28%22Encoding%20is%20Key%22%29%3C%2Fscript%3E | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_7/index.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content to users and such as malware. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| <https://portswigger.net/web-security/cross-site-scripting>  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### 

#### Proof of Concept

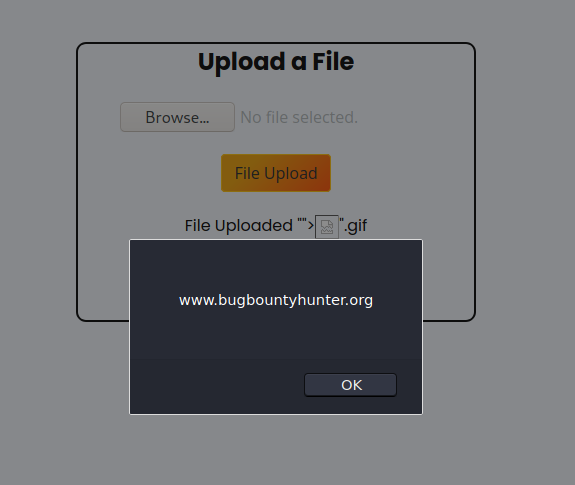
# 

URL Encoded Bypass

### 1.8. XSS With File Upload (File Name)

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_file\_name\_upload | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| XSS injection using filename upload | |
| **Payload** | |
| '""><img src=x onerror=alert(document.domain)>"'.gif | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_8/lab\_8.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content to users and such as malware. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| https://medium.com/@sarang6489/file-upload-xss-using-filename-f2f53e10033d  https://medium.com/@lucideus/xss-via-file-upload-lucideus-research-eee5526ec5e2  https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### Proof of Concept



XSS Filename Upload Execution

### 

### 1.9. XSS With File Upload (File Content)

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_file\_content\_injection\_01 | Medium - CVSS Score 5.8 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| XSS file content injection in subscription input field. | |
| **Payload** | |
| GIF89a/\*<svg/onload=alert("XSS\_with\_File\_Upload\_(File\_Content)")>\*/=alert("XSS")//; | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_9/index.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content to users and such as malware. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| https://medium.com/@sarang6489/file-upload-xss-using-filename-f2f53e10033d  https://medium.com/@lucideus/xss-via-file-upload-lucideus-research-eee5526ec5e2  https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### Proof of Concept

#### 

XSS File Content Injection

### 

### 1.10. Stored Everywhere!

|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_stored\_injection\_01 | High - CVSS score 7-8.9 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| XSS injection in the “email” parameter subscription input field. | |
| **Payload** | |
| ">N7RUZN<script>alert("Stored Everywhere!");</script> | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_10/lab\_10.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content to users and such as malware. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| https://portswigger.net/web-security/cross-site-scripting  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS | |

#### Proof of Concept

#### 

Stored XSS Injection

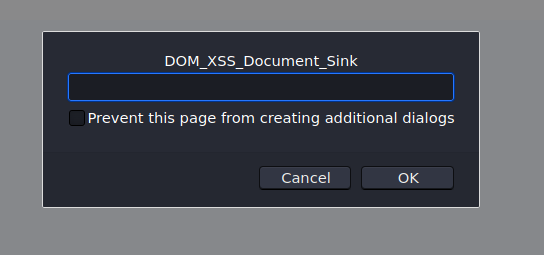
### 

### 1.11. DOMs Are Love!

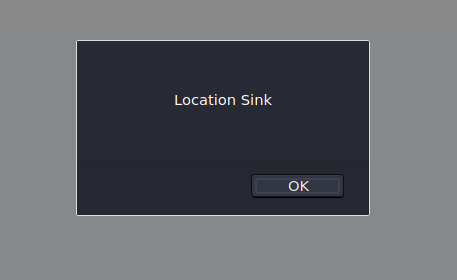
|  |  |
| --- | --- |
| **Reference** | **Severity** |
| xss\_dom\_injection\_01 | Critical - CVSS Score 10 |
| **Tools Used** | |
| Firefox Browser | |
| **Vulnerability Description** | |
| XSS injection in the “email” parameter subscription input field. | |
| **Payload** | |
| docSink"><img src =q onerror=prompt(1)>  -----------------------------------------------------------------------------------------------------------------------  javascript:alert("Location Sink")  -----------------------------------------------------------------------------------------------------------------------  btc"><iframe src=javascript:alert("Execution\_Sink")> | |
| **How It Was Discovered** | |
| Manual Analysis | |
| **Vulnerable URLs** | |
| https://www.bugbountyhunter.org/internship\_labs/HTML/xss\_lab/lab\_11/lab\_11.php | |
| **Consequences of not Fixing the Issue** | |
| * Attackers are able to read/modify/delete webpage content. * Steal cookies and hijack user sessions. * Serve malicious content to users and such as malware. | |
| **Suggested Countermeasures** | |
| * Validate to catch potentially malicious user-provided input. * Encode output to prevent potentially malicious user-provided data from triggering automatic load-and-execute behavior by a browser. * Implement Content Security Policy * Use appropriate response headers * Implement WAFs. | |
| **References** | |
| <https://portswigger.net/web-security/cross-site-scripting>  <https://owasp.org/www-project-top-ten/2017/A7_2017-Cross-Site_Scripting_(XSS)>  https://portswigger.net/web-security/cross-site-scripting  https://brutelogic.com.br/blog/dom-based-xss-the-3-sinks/  https://www.acunetix.com/blog/articles/finding-source-dom-based-xss-vulnerability-acunetix-wvs/  https://brutelogic.com.br/blog/location-based-payloads-part-i/  https://medium.com/@shilpybanerjee/dom-under-attack-i-followed-you-from-source-to-the-sink-5427adc04785  https://owasp.org/www-project-web-security-testing-guide/v41/4-Web\_Application\_Security\_Testing/07-Input\_Validation\_Testing/01-Testing\_for\_Reflected\_Cross\_Site\_Scripting.html  https://github.com/s0md3v/AwesomeXSS  <https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html> | |

#### 

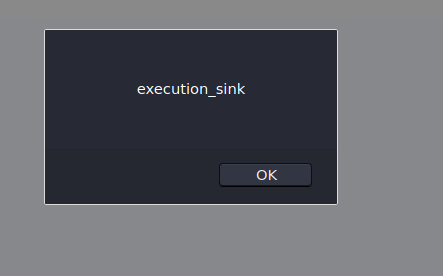
#### Proof of Concept



Document Sink XSS



Location Sink XSS



Execution Sink XSS