Ultimate XSS Beginner guide

BY UNCLE RAT

Agenda

- What is XSS?
- XSS Contexts
  - Javascript context
  - HTML tag attribute
  - HTML tag
- Types of XSS
  - Reflected XSS
  - Stored XSS
  - DOM XSS

Agenda

- How to test for XSS
  - Reflected
  - Stored
  - DOM XSS
- Getting around Filters
- Raising our impact

What is XSS

What is XSS

- Allows attacker to inject client-side script
- Happens because developer does not sanitize input
- XSS attacks are possible in VBScript, ActiveX, Flash, and even CSS
- Most common in javascript

XSS Contexts

XSS Contexts
- For stored and reflected XSS

- XSS into JS
- XSS in HTML tag attributes
- XSS between HTML tags
- XSS in JavaScript template literals
- XSS in the context of the AngularJS sandbox
- XSS in CSS
- XSS in headers
- ...

XSS Contexts – Into JS

XSS Contexts – Into JS

- `const poem = "The Wide Ocean";`
- `const author =` prompt("Please enter your name", "Harry Potter");
-
- `const favePoem = \`My favorite poem is ${poem} by ${author}.\`;`

XSS Contexts – Into JS

- Since we can control author
  - We can insert `
  - We can break out of function
  - We can insert our javascript

  - `alert()/*
    - ` to break out of JS function
    - Alert() to pop an alert
    - /* to put the rest of the JS in comment so it doesn't break

XSS Contexts – XSS in HTML tag attributes

XSS Contexts – XSS in HTML tag attributes

- `<script>var name= prompt("Please enter your name", "Harry Potter");</script>`
- `...`
- `<input onload="this.value=name;" style="margin-top: 3%">`

- We can
  - Break out of the attribute if our input is not sanitized
  - Insert our own arbitrary JS code
  - Execute XSS
  - `"><script>confirm()</script>`

XSS Contexts – XSS in HTML tag

XSS Contexts – XSS in HTML tag attributes

```
<head>
    <title>Test</title>
</head>

<body>
    <h1>Hi there<span id="username"></span>!</h1>
    <script>
        let userName = prompt("What is your name?");
        document.getElementById('username').innerHTML = userName;
    </script>
</body>
```

XSS Contexts – XSS in HTML tag attributes

- We can
  - Insert our own HTML tags if developer did not sanitize properly
  - Make that a javascript executing tag
  - Execute arbitrary javascript code
  - Execute a XSS attack

XSS Contexts – XSS in HTML tag attributes

- Input = <script>confirm(document.cookie)
- Input = <img src=x onerror=confirm()>
- ...

Types of XSS

Types of XSS

- Three main attack types
  - Reflected XSS
  - Stored XSS, this includes blind XSS
  - DOM XSS
- Many more smaller ones

Types of XSS – Reflected

Types of XSS - Reflected

- User input gets reflected
  - Into an attribute of an html tag
  - Into the HTML page
  - Into the javascript context
  - ...
- User input **does not** gets stored into database
- User input is not properly sanitized
- User input can contain javascript code

Types of XSS – Stored

Types of XSS - Stored

- User input gets stored into database
- Database value gets reflected
  - Into HTML page
  - Into HTML tag attribute
  - Into the javascript context
  - …

Types of XSS - Stored

- User input is not sanitized properly
  - At input
  - **And** at write to the database
  - **And** at read from database
- User input can contain malicious javascript code

Types of XSS – DOM

Types of XSS - Stored

- var search = document.getElementById('search').value;
- var results = document.getElementById('results');
- results.innerHTML = 'You searched for: ' + search;
  - Results.innerHTML <<< DOM Sink

Types of XSS - Stored

- DOM Sinks
  - Where user input enters the Document Object Model
    - Eval()
    - InnerHTML
    - …

Types of XSS - Stored

- Most common sources for DOM XSS
  - Arises from windows.location
  - Usually in query string (?)
  - **OR** fragment portion of URL(#)

How to test for XSS – General

How to test for XSS - General

- Attack vectors
  - Depend on the context
    - JS: "'`…
      - Single quote, double quote, backtick... to break out of JS function
    - HTML: <img src=x>
      - First test for HTML injection, then expand to XSS
    - HTML attribute: '>">`>…
      - Single quote, double quote, backtick... to break out of html tag attribute
  - All are simple, if they break the page or insert image, look deeper
  - Replace the <img src> tag with your own tag like <script>

How to test for XSS - General

- Passive method
  - As soon as you register, enter attack vector into every input field you see
  - Hope XSS pops somewhere down the line

How to test for XSS – Passive method

Positives

Low effort

Easier to test hidden features by just clicking around

Negatives

Not very effective

How to test for XSS – Reflected

How to test for XSS - Reflected

- Test every entry point
- Submit a random value (ex. Sdfs8f453168)
- Determine the reflection context
  - JS/HTML/Attribute/...
- Test a payload based on the location the value is reflected in
- Test alternative payloads

How to test for XSS – Stored

How to test for XSS - Stored

- Test every location that stores user input
- Investigate the context in which it is rendered onto the page
  - JS/HTML/Attribute/...
- Craft an exploit based on context
- Craft an alternative exploit if the first one doesn't work

How to test for XSS – DOM

How to test for XSS - DOM

- Testing HTML sinks
  - Place random value into source
    - I.e. random value into location.search
      - https://www.google.com/submit.htm?email=dsf dfssdfsdfsdfsdfsdfsdfds
      - Location.search would return ?email=dsfdfssdfsdfsdfsdfsdfsdfds
  - Inspect the HTML ***using DEVELOPER TOOLS only***
    - View source doesn't take DOM into account

How to test for XSS - DOM

- Look for your random value in the DOM
- If you string appears in the DOM you need to
  - Identify context
  - Craft exploit based on context
    - I.e. if string enters in double qoute, we might break out by using double quote
  - If your data gets URL-encoded before being processed
    - an XSS attack is unlikely to work.

Getting around filters

Getting around filters

- Many different filters out there
- Blacklist based
  - Try fuzzing all possible HTML tags and javascript handlers
  - Try URL encoding XSS attack vector
  - Try double or triple encoding it
  - Try putting blacklisted word into other blacklisted word as filtering might only happen one time leaving our original blacklisted word intact

Getting around filters

- Whitelist based
  - Very hard to get around
  - Only allow certain words
  - Block the rest
- Pattern based
  - Tries to look for things that look like a XSS attack
  - Depends on the configuration

Raising our impact

Raising our impact

- Steal cookies
  - Very hard recently
  - Httponly flag is gaining traction
- Execute a keylogger
  - Getting harder to smuggle out data
- Steal data from the page
  - Same as keylogger
- Execute JS functions on page