ulm university universität
uulm

Institute of Distributed Systems
Ulm University

Cross Site Request Forgery

Michael Müller | Practical IT-Security | ss 2014

Structure

The following assignments have been written short on hints. This is due to the fact that I don't know on which level your skills are. If you are stuck on something, please don't hesitate to ask me. It would be a pity, if you would spend half the time on figuring out JavaScript syntax subtleties. It may be helpful for the assignments to revisit the presentation slides. I have made them available at http://share.creal.de/csrf/talk/.

To give a rough structure of the practical assignments:

Part 1	ca. 30 min
Presentation Solution	ca. 10 min
Part 2	ca. 60 min
Presentation Solution	ca. 20 min

Exercise 1: Metasploitable

ca. 30 minutes

These tasks will be executed using Metasploitable. Please start the application and make sure you have access to the webserver of the virtual machine.

a) **Damn Vulnerable Web App** Direct your browser to the IP-address of the Metasploitable image, e.g. http://192.168.1.103/dvwa. Configure Security Level `low` as described in the preparation document.

   Try to exploit the CSRF vulnerability available via the navigation bar on "/dvwa/vulnerabilities/csrf/". The button "`View Source`" might provide additional information. Your exploit should be a HTML page containing only HTML markup code. Once this markup is interpreted by a browser, a HTTP request should be send by the browser in order to set a new password. Think about which typical HTML elements execute HTTP requests in order to load a further resource.

b) **Damn Vulnerable Web App** Configure Security Level `medium` and try to exploit the hardened CSRF vulnerability. Your exploit should consist of HTML markup and JavaScript code this time.

c) **TWiki** Write an exploit which creates a new wiki page. To achieve this, prepare a HTML page with the embedded exploit. The exploit should not rely on a user being logged in. Feel free to use HTML and/or JavaScript.

Exercise 2: Advanced CSRF

ca. 60 minutes

These tasks will be done using the advanced-csrf web application. Please see the Moodle platform for the advanced-csrf.zip. Please unzip the archive to your web server executable directory.

You will then need to create a database "advanced-csrf" within MySQL and import the dump.sql into your MySQL database. You might then need to adapt the MySQL credentials within the "advanced-csrf/mysql.inc.php".

a) The site "/ex2-1/leaking.php" provides a way of leaking textual information. Suppose somebody wants to frame a certain person of leaking information. They craft a mail and use social engineering mechanisms as a mean to get a victim to visit a website which contains a malicious CSRF attack.

   How would the exploit on such a website need to be written as a mean to submit the form with textual information, without the user noticing? Use JavaScript and HTML to assemble such a website.

b) The site "/ex2-2/index.php" contains a form which the developers tried to make safe against CSRF attacks using the Token Synchronizer Pattern.

   Try to exploit the application in a way that when a visitor visits "/ex2-2/index.php" he automatically votes for banana without noticing it.

c) The site "/ex2-3/index.html" contains a multi-step form for a subscription service. Your goal is to prepare an exploit which will lead the browser who interprets the code into submitting the completed survey with options of your choice.

d) The voting page has an administration interface. Assume the administrators are logged into it and have full access to the JSON API of the administration interface. For example, this command will reset the voting:

```
POST /ex2-3/api/results HTTP/1.1
Host: localhost

{
  "action": "reset-voting",
  "value": "all"
}
```

   The administrators are very careful and have turned off JavaScript within their browsers. However, they still get send a faked mail with a link to a pretty website – which they of course visit.

   How can you write a – purely HTML based – exploit, which will send manipulated calls to the API? Please write an exploit which makes a POST request with a JSON payload.

```
$ ./happy_hacking --now --mood ":)"
```