ulm university universität ulm

Institute of Distributed Systems
Ulm University

Preparation Document

Cross-Site-Request-Forgery

Practical IT-Security

SS 2014

Michael Müller
michael-4.mueller@uni-ulm.de

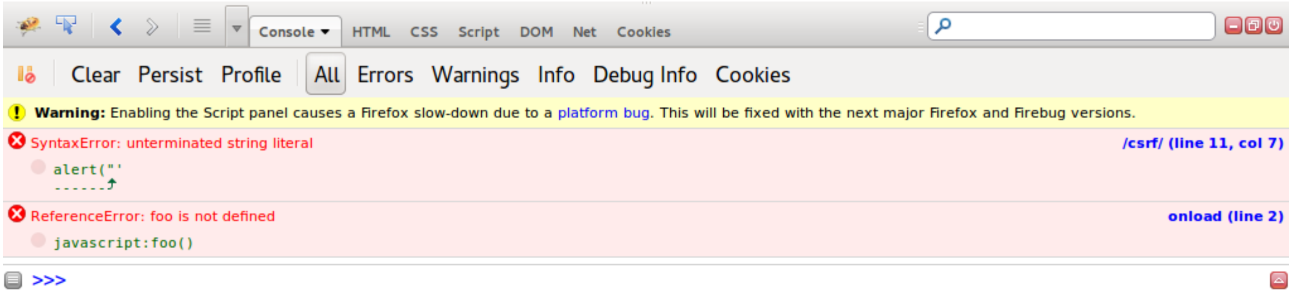
1 Introduction

Cross-Site-Request-Forgery (CSRF) is an attack on web-applications which will be further explained within the presentation. This document is meant as a preparation document in order to get you ready to understand the presentation and conduct the assignments.

1.1 Web Developer Tool

We aim to exploit several web-services in the assignments. Most of the exploits will be written in HTML in conjuction with JavaScript. For this purpose it comes handy if you are familiar with the developer tool of your favorite webbrowser. Most modern browsers come with some kind of developer tool. Check the corresponding manual or install a fitting plugin like Firebug (for Firefox). For Chromium the developer tools are accessible via "Tools > Developer Tools" (or via pressing F12). The main features of the developer tools, which will come handy for us, are:

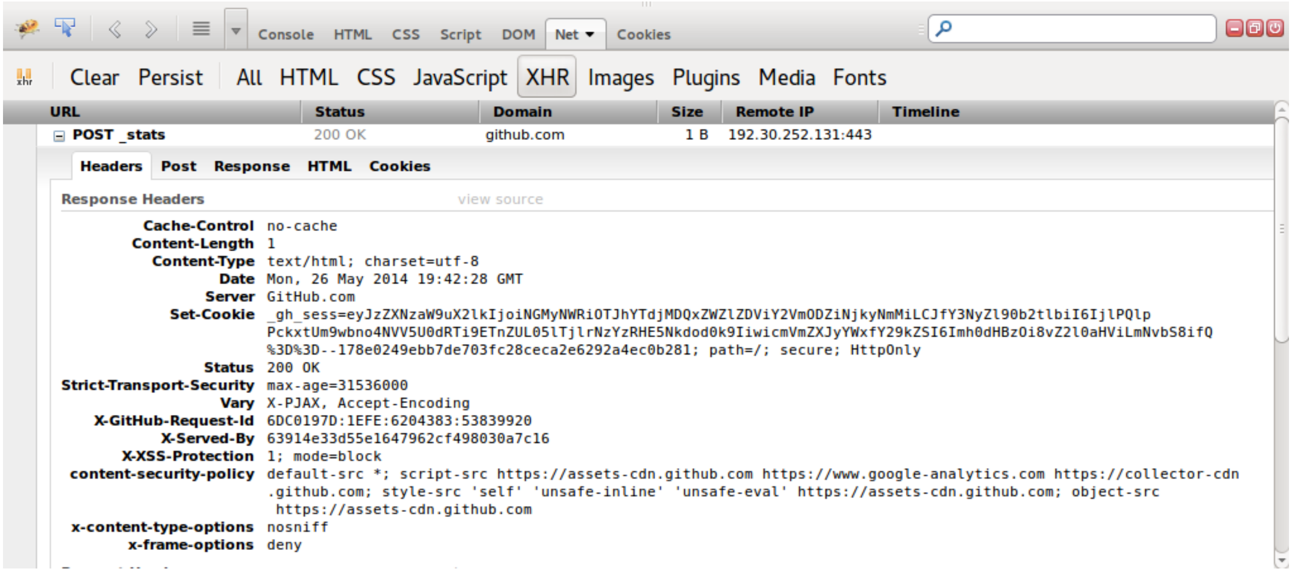
The JavaScript Console Comes handy when debugging JavaScript code, since execution errors are displayed and you have the ability to interactively execute code on the webpage.



```
! Warning: Enabling the Script panel causes a Firefox slow-down due to a platform bug. This will be fixed with the next major Firefox and Firebug versions.
✕ SyntaxError: unterminated string literal                                      /csrf/ (line 11, col 7)
    alert("'
    ------↑
✕ ReferenceError: foo is not defined                                                   onload (line 2)
    javascript:foo()
>>>
```

Figure 1: Firebug JavaScript console.

The Network Capturer Enables you to analyze and debug HTTP requests/replies which are send/received by the browser. Of course, you can also use Wireshark or similar tools as means to capture HTTP traffic. From my experience, however, it comes handy to use the builtin browser tools, since they oftentimes are nicely integrated and tie in well with the HTML/JavaScript coding workflow.



URL	Status	Domain	Size	Remote IP	Timeline
⊟ POST _stats	200 OK	github.com	1 B	192.30.252.131:443	

Response Headers

```
           Cache-Control no-cache
          Content-Length 1
            Content-Type text/html; charset=utf-8
                    Date Mon, 26 May 2014 19:42:28 GMT
                  Server GitHub.com
              Set-Cookie _gh_sess=eyJzZXNzaW9uX2lkIjoiNGMyNWRiOTJhYTdjMDQxZWZlZDViY2VmODZiNjkyNmMiLCJfY3NyZl90b2tlbiI6IjlPQlp
                         PckxtUm9wbno4NVV5U0dRTi9ETnZUL05lTjlrNzYzRHE5Nkdod0k9IiwicmVmZXJyYWxfY29kZSI6Imh0dHBzOi8vZ2l0aHViLmNvbS8ifQ
                         %3D%3D--178e0249ebb7de703fc28ceca2e6292a4ec0b281; path=/; secure; HttpOnly
                  Status 200 OK
Strict-Transport-Security max-age=31536000
                    Vary X-PJAX, Accept-Encoding
     X-GitHub-Request-Id 6DC0197D:1EFE:6204383:53839920
            X-Served-By 63914e33d55e1647962cf498030a7c16
       X-XSS-Protection 1; mode=block
 content-security-policy default-src *; script-src https://assets-cdn.github.com https://www.google-analytics.com https://collector-cdn
                         .github.com; style-src 'self' 'unsafe-inline' 'unsafe-eval' https://assets-cdn.github.com; object-src
                          https://assets-cdn.github.com
  x-content-type-options nosniff
         x-frame-options deny
```

Figure 2: Captured HTTP traffic within Firebug.

2 Assignments

To carry out the assignments you need to bring your own laptop configured to the run the services described in this section.

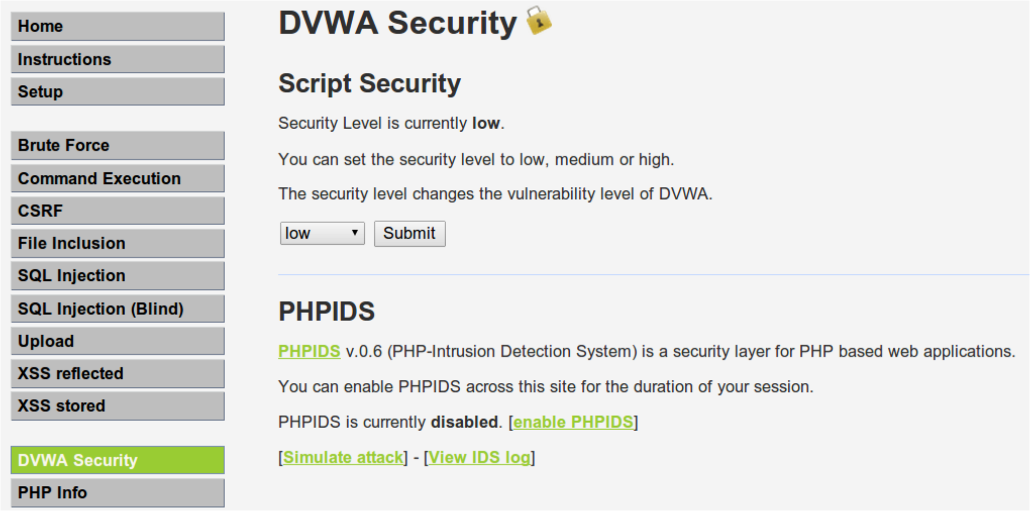
2.1 Metasploitable

We will continue to use the Metasploitable image, which has been provided in the warm-up phase of the course. If you no longer have it at hand you could download it from the Moodle platform again.
Use a virtualisiation software to get the image running as a virtual machine (vm). The VirtualBox software works well, for example. Configure the vm in a way that you have access to it over network (e.g. by configuring a Bridged Adapter within VirtualBox). If network access does not work out of the box you may have to log into Metasploitable (User: `msfadmin`, Password: `msfadmin`) and execute "`$ sudo dhclient etho`" in order to obtain an IP address.
If everything worked out, you should be able to access the Metasploitable webserver from your host machine. If the vm e.g. runs on 192.168.1.103, the address would be http://192.168.1.103/.

2.1.1 Damn Vulnerable Web App

If you have not already done so in the warm-up phase of the course, you should now log into the web application in order to get a feeling for the interface. Open up http://metasploitable-vm-ip/dvwa and log-in with the credentials User: `admin`, Password: `password`. Open the navigation tab "DVWA Security" and configure "Security Level: Low" (see Figure 3).
During the assignments we will work our way through the security levels low and medium. The level high is meant to show the state of art of a best-practice solution. It is not meant to be cracked.

Note: The security level which you configure is saved as a Cookie value. The default value is high. This might lead to problems while debugging: if you e.g. log into DVWA using Chromium and configure low you should also run your exploits in Chromium and not in e.g. Firefox. Firefox would still be configured for the security level high.



DVWA Security

Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

PHPIDS

PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [enable PHPIDS]

[Simulate attack] - [View IDS log]

Figure 3: Beware! The security level is only set temporarily as a Cookie!

2.1.2 TWiki

The other Metasploitable service which we will examine is TWiki. If the Metasploitable image is running you can access the application via http://metasploitable-vm-ip/twiki/.

2.2 advanced-csrf

The advanced-csrf package is a simple web application designed to be exploitable for CSRF/XSS attacks. You will obtain a link to an archive with the PHP code and a MySQL dump at the start of the assignments. The application requires a WebServer with PHP v.5.x support and a MySQL v.5.x database in order to run.

The easiest way to run the application is to set up a local webserver. A comfortable way of doing so is to use XAMPP, an environment which comes with a webserver and provides pre-configured PHP and MySQL support.

3 Quick Checklist

If you have encountered errors within the previous steps or have any questions regarding the assignments you can contact me via michael-4.mueller@uni-ulm.de. To give a quick, summarized overview of the requirements for the assignments:

You should bring a basic understanding of
HTTP, HTML, the DOM, PHP, JavaScript and JSON. We won't dive deep into markup or language subtleties, but you should have a basic understanding. It would be a pity if you would spent a lot of effort into fiddling on syntax particularities, so don't be afraid to ask me if you get stuck. The DVWA offers the possibility of displaying the (PHP) code of the web page which you are trying to exploit. A basic understanding of PHP may come in handy here.

You should bring
A laptop. Configured to run the Metasploitable virtual machine. The advanced-csrf web application should be able to run as well (requires PHP v.5.x and MySQL v.5.x).

You should have played with
A tool to capture HTTP requests/replies of your webbrowser and a JavaScript console, which displays execution errors.

You should be willing
To enjoy learning something new and nibble on puzzles.

```
$ ./happy_hacking --from 1401865200000 --to 1401876000000 --at "O27 / 341" --mood ":)"
```