Secure Coding - Team 7- Phase 5

Magnus Jahnen, Thomas Krex, Elias Tatros

January 22, 2015

Part I

Executive Summary

Contents

Part II

Time Tracking Table

Chapter 1

Time Tracking Table

Time Tracking		
Name	**Time**	**Description**
Magnus Jahnen	15h	Reverse engineer batch parser (team 8)
	2h	Reverse engineer batch parser (own)
	5h	Reverse engineer Java-SCS (team 8)
	2h	Reverse engineer Java-SCS (own)
	10h	Meetings
	5h	Report & Presentation
Thomas Krex	3h	Self introduction into test environment and target application
	5h	Searching for Vulnerabilities in PHP&JavaScript Code and of Team 7
	11h	Testing own app according owasp checklist
	6h	Exploiting Proccessing Time and Account guessing vulnerabilities
	10h	Meetings
	6h	Documentation
Elias Tatros	2h	Static Analysis decompiled Java and PHP
	4h	Finding Encryption Flaws in Java/PHP
	5h	Analysis of application memory
	15h	Planning, Implementation and Testing of Memory Scanner
	6h	Working on Report (Sections on Key Weakness, Memory Scanner, Static IV)
	10h	Meetings

Part III

Application Architecture

Part IV

Security Measures

Chapter 2

Security Measures

2.1 PDF Password Protecion

2.2 HTTPS

2.3 Strong password policy

2.4 Secure Session Management

2.5 title

Part V

Fixes

Chapter 3

Fix of HTTP Strict Transport Security

3.1 Affected Files

/etc/apache2/httpd.conf: line 15

3.2 Description

We added an HTTP Scrict Transport Security Header to teh config of our web server. Therefore the server notifies the client's browser that all traffic has to be exchanged via HTTPS. To do so the added the following line to the Virtual Host Config:

```
Header add Strict-Transport-Security "max-age=15768000"
```

Chapter 4

Fix for Bypassing Session Management Schema and Cookies attributes

4.1 Affected Files

- /etc/apache2/httpd.conf lines: 2++

4.2 description

In order to protect the session cookie and its attributes against attackers the added the following settings to httpd.conf:

```
php_value session.cookie_httponly true
php_value session.cookie_secure true
php_value session.cookie_lifetime 1800
```

The httponly flag was already set in phase 3 and prevents that the cookie can be acessed by javascript. The http_secure flag ensures that cookies are only sent via HTTPS.
Futhermore the attribute cookie_lifetime defines the expire data for a cookie. A short lifetime decreases the chances for an attacker to sucessfully use a foreign session cookie to authenticate with the web service.

Chapter 5

Fix Process Timing

5.1 Affected Files

- webroot/pw_recovery.php: 137++

5.2 Description

Our App was leaking information by its process timing. To be more precise, the vulnerability was placed in the password recovery service. The User can enter his/her email address and will retrieve an email with an new password. The problem was the time, the system took to send that email. If entering an email which isn't registered in the system, the process takes round about 30 ms instead of 3 seconds. This means that an attacker can guess regitered email addresses my observing the process time. To fix this problem there is more than one solution. You could start an asynchronous task that is sending the mail while the web service reponds to the client. For that you have to use external libraries that would increase the complexity of the system. Another way would be to send a curl request to an internal web service. This basically would work but it requires a static host name to do so. So we decided to chose a third solution. We inserted a sleep command that will wait a randomly picked interval between two and four seconds if the email address is not known to the system. This prevents an attacker of easily guessing registered email adresses.

Listing 5.1: Random Timeout in pw_recovery.php

```
if($user->email) {
// if we found the mail it is a valid user
$user->sendPwRecoveryMail();
}
else {
// timeout , so it's not that easy to guess existing accounts
    via the processing time.
$timeout = rand(2,4);
sleep($timeout);
}
```