Secure Coding WS14/15

Team 20 Phase 5

Presented by

Michael Remmler

Bharti Munjal

Simon Barth

Executive Summary

In this report we first talk about the architecture of our application "Kangal Bank." Our application is based on the phpsec framework following its basic structure. The application uses the Object Oriented Paradigm and is ensured to follow general coding guidelines.

The report then describes the security measures we have taken in our application to protect it from attacks. Our application is tested to be safe from all security risks discussed in the OWASP testing guide. Security measures include strong session management (using phpsec) to CSRF protected forms and role based access control (RBAC) to name a few.

The report also talks about the fixes we did after phase 4. A total of 13 issues found in phase 4 have been fixed. The issues were minor and required little changes in the code.

The "Kangal Bank" is tested to be safe from vulnerabilities and users can safely trust the application. We concentrated on developing a sound solution of an online banking system with a focus on security and code quality, not only allowing easy security reviews but also to provide good maintainability.

Table of Content

Time Tracking Table

Task	Student	Workload
Project Management	Each	2h
Creating Presentation + Report + Video	Each	15h
Server configuration, Browser Cache, Change Password, Customized Error Page	Michael	5h + 0,5h + 4h + 2h = 11,5h
Session Management, HTTP Verb Tampering, PIN via unsecure channel (Email), try java obfuscating	Bharti	2h + 2h + 1h + 2h = 7h
HSTS, server configuration, Change Password, C++ obfuscator	Simon	1h + 1h + 2h + 3h = 8h

Application Architecture

Application Structure

Our application is present in "/var/www/https" directory. This directory has two sub-directories "framework" and "libs".

1. "`framework`": This folder contains the application code.
2. "`libs`": This folder has framework's basic code for "session", "authentication" etc.

This structure is given by the phpsec framework. Inside the "framework" subdirectory the structure is as follows:

`framework/control`	`Controllers according to the MVC pattern`
`framework/view/default`	`View according to the the MVC pattern`
`framework/model`	`Model according to the MVC pattern`
`framework/static`	`Static files like CSS, the batch template and pictures`
`framework/uploads`	`Temporary storage of uploaded batch files`
`framework/exec`	`Contains the batch transaction executable and template for SCS`
`framework/config`	`Configuration files`

Inside of the "`control`" and "`view/default`" directories there are separate subdirectories for employees, customers and users.

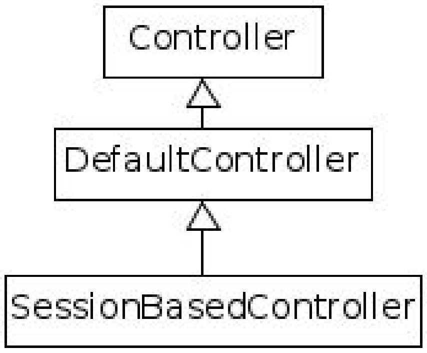
Application Design

MVC Pattern

The application follows the Model-View-Controller pattern. This pattern divides the responsibility of data storage, data mutation and data display into the domains of model(persistence), view(display) and controller(business logic). Our use of this pattern can be seen on the file system level(see section “Application Structure” above), as well as in the code. Database access is only done in the model, business logic is in the controllers and the views do nothing more than displaying that data. We departed from the pure MVC pattern at one point: we put some of the business logic into stored procedures within the database. This has the advantage that no other application that might be developed at a later time using the same database can reuse those stored procedures and code duplication is avoided.

Object Oriented Paradigm

The application follows the Object Oriented Paradigm. This can be observed very well at the example of controllers.



Controllers that are only available to employees or customers inherit from `SessionBasedController` whereas Controllers that are available to anybody inherit directly from `DefaultController`.
The application has been abstracted in well-defined classes with specific roles. A few examples are:

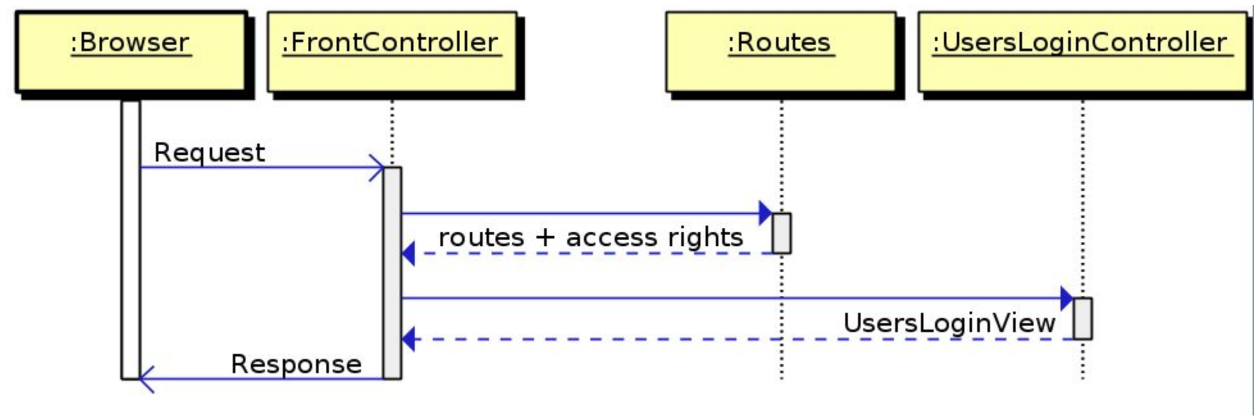
1. `NewPasswordController`
2. `LoginController`
3. `RegistrationController`

The design makes the code easy to understand and allows easy extension in future. The steps needed to add new pages are:

1. Add controller for new page
2. Add view for new page
3. Create route for new page and set access restrictions

The hierarchical design ensures the common functionality abstracted out in superclasses. There is no redundant code in the application.

Application Flow



The application has a single point of entry, which is called “`FrontController`”. This controller takes an HTTP request and checks if the user issuing this request is allowed to access the requested content. Afterwards the file “`routes.php`” is used to translate the requested URL into the responsible controller for this URL. As a last step control is given to this responsible controller.

The routes not only contain information about which controller is responsible for a given URL but also which URLs are accessible for the different roles in the system (user, customer and employee).

PHPSec Framework

Our application is built on the phpsec framework. The framework is part of the OWASP Security Project and provides tools and libraries to build secure applications. The framework provides features like better session management compared to the php built-in session management, basic user management including temporary password and activation link functionality, encrypted strings, database access layer and a class to protect against CSRF attacks. Even though phpsec provides lots of functionality we had to changes and correct certain parts of the framework. The following things had to be changed:

Changes in cookie attributes

The attributes of session cookie "SESSIONID" are hard-coded in session.php in function `updateUserCookies`(...). We changed the value of attribute "secure" to "true" for the session Cookie in this file. Ideally all these attributes should be configurable from a configuration file and no change should be made in the framework code.
The `setSessionData`(...) function is used to store and update key-value pairs in session of a user. But there was no facility to remove any key-value pair from the session. So we added a new `deleteSessionData`(...) function to allow removal of a given key.

Change in CSRF protection mechanism

The original implementation used the standard php session management so we changed it to use the phpsec session management.

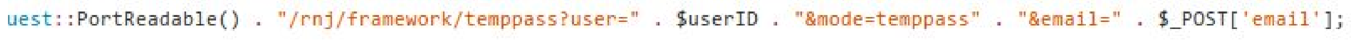
Addition in SQL statement handling

Using the phpsec framework to execute an SQL statement is done like this:
`\phpsec\SQL(“SELECT * WHERE ACCOUNTS USERID = ?”, $userid);`
The facility to execute these statements is located in the file “`libs/db/adapter/base.php`”. Particularly the function “`SQL`” in the class “`DatabaseModel`” is of interest here. It scans the query for keywords like “`SELECT`”, “`INSERT`” and “`DELETE`” and returns different values dependent on the keyword. The problem we faced however was that big parts of our logic is contained in stored procedures. The “`DatabaseModel`” class was not prepared to handle the keyword “`CALL`” though, so we couldn’t use it as is. We introduced handling for “`CALLS`” in the same manner “`SELECT`” statements are handled, by returning all results.

Security issue in built-in “Forgot Password” page

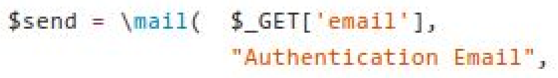
Phpsec contains an example page for “Forgot Password” functionality. However it contains a big security hole.

1. The user has to enter his email address
2. The system looks up the user using this email address
3. The system redirects the user to a new URL which is handled by the “`TemporaryPasswordController`” and passes the username and email as get parameters of this new URL:



```
uest::PortReadable() . "/rnj/framework/temppass?user=" . $userID . "&mode=temppass" . "&email=" . $_POST['email'];
```

4. The “`TemporaryPasswordController`” sends an email with a temporary password to the email address it reads from the URL:



```
$send = \mail(  $_GET['email'],
                "Authentication Email",
```

The problem is that an attacker could use this to acquire a temporary password for any account by using this redirected URL and inputting his mail address for the “`email`” parameter.

Third Party Libraries Used

fpdf_protection

This library was used to generate password protected pdfs for sending TAN list to the bank users. The library code is present at `/var/www/https/libs/fpdf`

phpmailer

This library was used to send mails to bank users. The code is present under `/var/www/https/libs/mail/`

Other Practices Followed

All the configurable variables like `DBNAME`, `DBUSER`, `EMAILFROM` etc. are taken from configuration file at `https/libs/config.php` and not hard-coded in the application anywhere. This file is not accessible via the internet so even if an attacker manages to download the source code of the application he cannot read those parameters.

Security Measures

Locking Mechanism

The phpsec framework provides a feature to detect brute force attacks. So if multiple unsuccessful login attempts are made, the user account would be locked and the user needs to perform a forgot password action. An attack is detected if one of the two conditions are true:
- 2 failed logins within 1 second
- 5 failed logins within 25 seconds

Session Timeout

If the user is inactive for a certain amount of time then he is logged out of the system by the application itself. Session timeout is provided by the phpsec framework and inactivity timeout has been configured to 10 minutes.

Strong Passwords

The application only accepts strong passwords from users during registration/change password. The framework assigns a score to the password. The password is allowed if the score calculated is above a threshold value. A good password score means

1. The password has a high entropy
2. The password should contain a number, lowercase character, uppercase character and a special character
3. There is no keyboard character sequence, no common key pattern nor lexicographical order in the password.
4. There is no date or phone number in the password.
5. There are non duplicated words

The characteristics which are used for the calculation already cover a wide range of possible weaknesses so users are protected from using weak passwords.

Strong Usernames

The application also ensures strong usernames. A strong username means:

1. The length is between 4 and 32 characters
2. It only contains “a-z0-9A-Z_@.-”

CSRF Tokens

The application generates a CSRF token for each session. This token is stored as hidden field in all the required forms and is sent within a POST requests. The token ensures extra security against Cross Site Request Forgery attacks. The token value for a session is changed once it is utilized.

Encrypted Transmission

The "kangal bank" sends data over an encrypted channel only. "https" is enforced by default and there is no option to switch to "http".

Cookies are Secure and cannot be accessed from JavaScript

The application uses one cookie "SESSIONID". The "`secure`" and "`httpOnly`" flags of this cookie are set to `TRUE`.

The site cannot be opened in IFrame

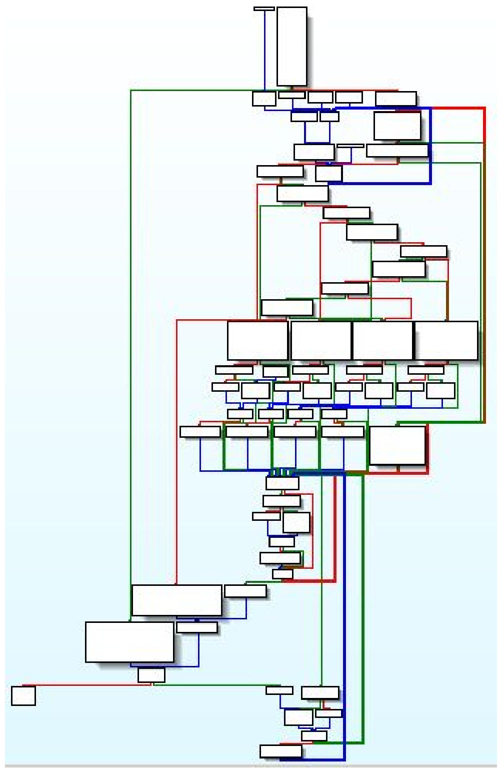
To prevent clickjacking the site is restricted to be opened in an Iframe.

Obfuscated Batch Transaction program

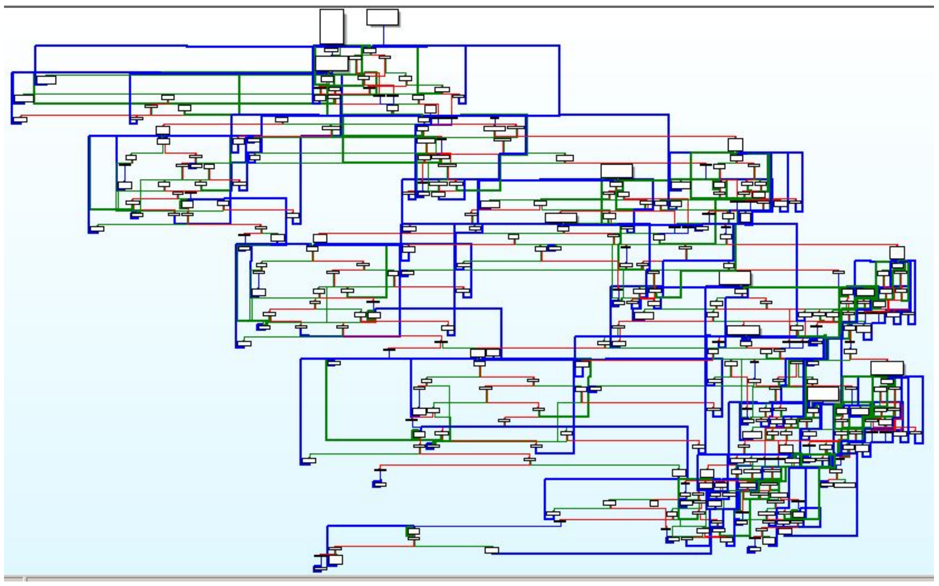
The “kangal bank” uses a C++ program to handle batch file transactions. We used “obfuscator-LLVM” to protect our application in case an attacker manages to download the binary file. We used the obfuscation options in the following order:

1. Control Flow Flattening (-mllvm -fla) to increase the number of blocks
2. Bogus Control Flow (-mllvm -bcf) to introduce fake connections between those blocks
3. Instructions Substitution(-mllvm -sub) to obfuscate the newly added code even more

The comparison of the IDA Pro graphs shows that the structure of the program is way more complex. The file size increased from 50 kB to 98 kB.



This picture shows the call graph before the obfuscations were applied.



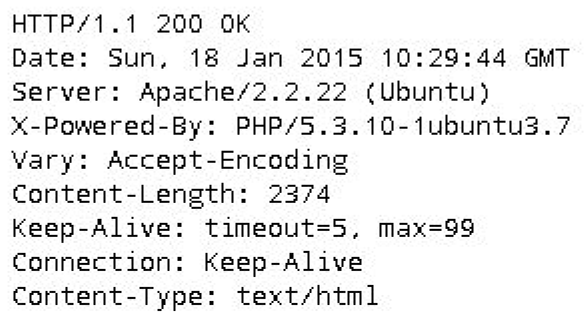
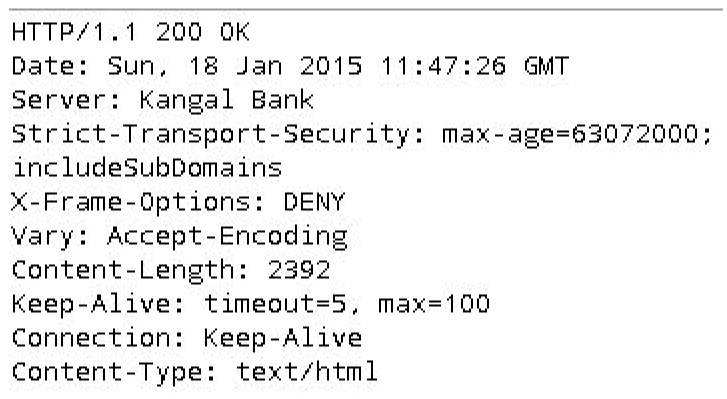
This picture shows the IDA Pro graph after the obfuscation techniques have been applied.

Fixes

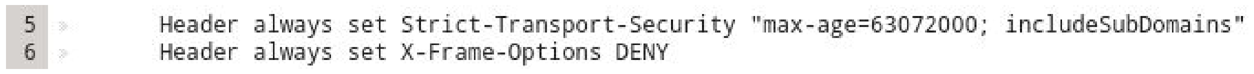
Configuration and Deploy Management Testing (4.3)

Enumerate Infrastructure and Application Admin Interfaces (OTG-CONFIG-005)

before	after

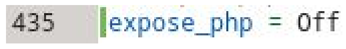

`HTTP/1.1 200 OK`
`Date: Sun, 18 Jan 2015 10:29:44 GMT`
`Server: Apache/2.2.22 (Ubuntu)`
`X-Powered-By: PHP/5.3.10-1ubuntu3.7`
`Vary: Accept-Encoding`
`Content-Length: 2374`
`Keep-Alive: timeout=5, max=99`
`Connection: Keep-Alive`
`Content-Type: text/html`
	
	`HTTP/1.1 200 OK`
	`Date: Sun, 18 Jan 2015 11:47:26 GMT`
	`Server: Kangal Bank`
	`Strict-Transport-Security: max-age=63072000;`
	`includeSubDomains`
	`X-Frame-Options: DENY`
	`Vary: Accept-Encoding`
	`Content-Length: 2392`
	`Keep-Alive: timeout=5, max=100`
	`Connection: Keep-Alive`
	`Content-Type: text/html`

File: `/etc/apache2/sites-enabled/default-ssl`
Line Number: 5 and 6
Description: In order to activate HTTP strict transport security (HSTS) we set the corresponding header option using mod_headers. This tells the browser to handle any communication with the server via secure channels. While at it we also activated the "`X-FRAME`" header option which works as ClickJacking protection on newer browsers. To also avoid these attacks on older browsers we already employ a frame braking script.



```
5       Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains"
6       Header always set X-Frame-Options DENY
```

File: `/etc/php5/apache2/php.ini`
Line Number: 435
Description: To give away less information about the technologies and versions used in our application removed the "`X-Powered-By`" header which is activated by php's standard configuration.



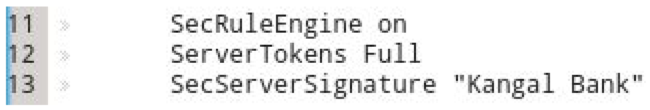
```
435   expose_php = Off
```

File: `/etc/apache2/mods-available/mod-security.conf`
Line Number: 11 to 13
Description: Another change to give away less information was to use mod-security in order to change the Server string contained in HTTP headers. The HTTP server is now called “Kangal Bank” giving away no information about which httpd, e.g. apache, nginx, IIS and neither which version is used by the application.

Firstly we need to install the required module with the following command:

```
      sudo apt-get install libapache2-modsecurity
```

Secondly we need to configure the module:



```
11 »      SecRuleEngine on
12 »      ServerTokens Full
13 »      SecServerSignature "Kangal Bank"
```

Identity Management Testing (4.4)

Testing for Account Enumeration and Guessable User Account (OTG-IDENT-004)

File: `libs/auth/user.php`
Line Number: 597
File: `framework/control/users/UsersLoginController.php`
Line Number: 39
Description: To show the same error message for correct username with wrong password and not existing username at the login screen, the message contained in the thrown exceptions have been changed to be the same. The `UsersLoginController` has been changed to use the error message contained in the exception rather than a custom error message. This still allows for internal distinction of the error message while there is no visible difference on the interface.

Authentication Testing (4.5)

Testing for default credentials (OTG-AUTHN-002)

Description: We changed the database password of the “root” user and created a new user with restricted rights only for the database access of the application.

Testing for Browser cache weakness (OTG-AUTHN-006)

File: `/var/www/https/framework/_core/base/control.php`
Line Number: 32 and 33
Description: To protect the secret pages we add the following header attributes to all pages by inserting these lines of code at a central point in the application.

```
header("Cache-Control: no-store, no-cache, must-revalidate, no-transform, max-age=0, post-check=0, pre-check=0");
header("Pragma: no-cache");
```

These attributes tell the browser not to cache the page and reload it every time when it is accessed.

Testing for weak password change or reset functionalities (OTG-AUTHN-009)

Our application provides a change password functionality where it is possible to change the password without providing the current password.

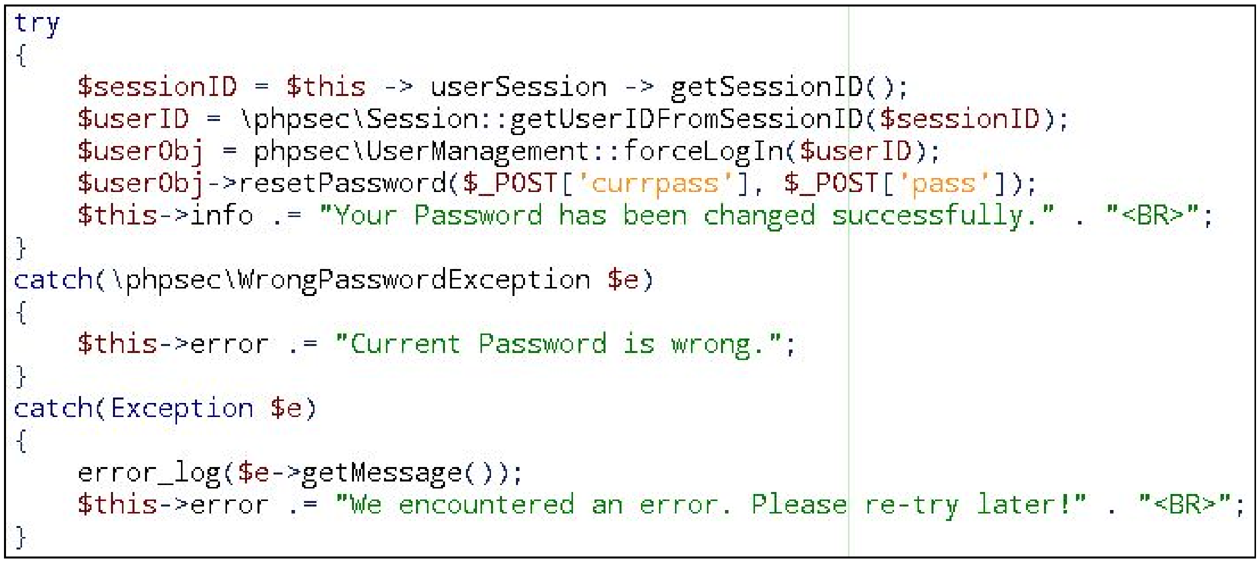
File: `/var/www/https/framework/view/default/users/ChangePassword.php`
Line Number: 28 to 31
Description: We added a new “`currpass`” field in the change password view.

```
<tr name="currpass-field" id="currpass-field">
    <td><label>Current Password:</label></td>
    <td><input type="password" name="currpass" id="currpass" maxlength="32"></td>
</tr>
```

File:
`/var/www/https/framework/control/users/ChangePasswordController.php`
Line Number: 46 to 62
Description: In the old version of the change password functionality we used the `forceResetPassword` function of the user class of phpsec framework. This function did not require the current password.

```
$userObj = phpsec\UserManagement::forceLogIn($userID);
if ($userObj->forceResetPassword($_POST['pass']))
    $this->info .= "Your Password has been changed successfully." . "<BR>";
else
    $this->error .= "We encountered an error. Please re-try later!" . "<BR>";
```

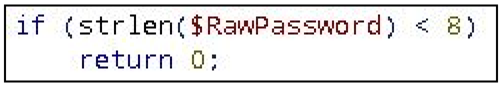
So we replaced the function call with a `resetPassword` function call. This call requires the current password. If the current password is wrong it throws a `WrongPasswordException`.



```
try
{
    $sessionID = $this -> userSession -> getSessionID();
    $userID = \phpsec\Session::getUserIDFromSessionID($sessionID);
    $userObj = phpsec\UserManagement::forceLogIn($userID);
    $userObj->resetPassword($_POST['currpass'], $_POST['pass']);
    $this->info .= "Your Password has been changed successfully." . "<BR>";
}
catch(\phpsec\WrongPasswordException $e)
{
    $this->error .= "Current Password is wrong.";
}
catch(Exception $e)
{
    error_log($e->getMessage());
    $this->error .= "We encountered an error. Please re-try later!" . "<BR>";
}
```

We also changed your reset password functionality. In the old version the users received an email with a URL that contains a security token. Using this URL he could login without providing username and password. The user will be redirected to the change password page. Since at this point the user has a valid session he can use other functionality of the application and is not bound to only change his password. This procedure was bad because the user wasn't forced to change the password. So we changed it. Now the user is only logged in until he successfully changed the password. We created a new view (`/var/www/https/framework/view/default/users/resetPassword.php`) and a new corresponding controller (`/var/www/https/framework/control/users/resetPasswordController`).

File: `/var/www/https/libs/auth/user.php`
Line Number: 307 to 308
Description: We also improve the password policy so that a password with at least 8 characters is required.



```
if (strlen($RawPassword) < 8)
    return 0;
```

Session Management Testing (4.7)

Testing for Bypassing Session Management Schema (OTG-SESS-001)

File: `/var/www/https/libs/session/session.php`
Line Number: 204 and 209
Description: The secure flag is set to TRUE in the `setCookie`(..) function. The lines 203 and 208 were commented and new function call were added.

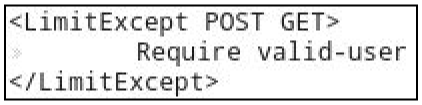
Testing for Cookies attributes (OTG-SESS-002)

File: `/var/www/https/libs/session/session.php`
Line Number: 204 and 209
Description: The secure flag is set to TRUE in the `setCookie`(..) function. The lines 203 and 208 were commented and new function call were added. Also the path variable is unchanged to "/" since our application is directly deployed under root directory.

Data Validation Testing (4.8)

Testing for HTTP Verb Tampering (OTG-INPVAL-003)

File: `/etc/apache2/sites-enabled/000-default`
Line Number: 13 to 15
File: `/etc/apache2/sites-enabled/default-ssl`
Line Number: 15 to 17
Description: In order to avoid HTTP verb tampering attacks we only accept HTTP methods POST and GET. We checked the supported HTTP methods using the command '`nikto -Plugins httpoptions -h <IP-ADDRESS> -port 443`' which didn't show any options. This is because the OPTIONS method is not available anymore. Before setting the '`LimitExcept`' option for apache, nikto outputted that the HTTP methods HEAD, OPTIONS, PUT and GET were available. From the output of nikto and because the website is still operating normal we deduct that only POST and GET methods are available.



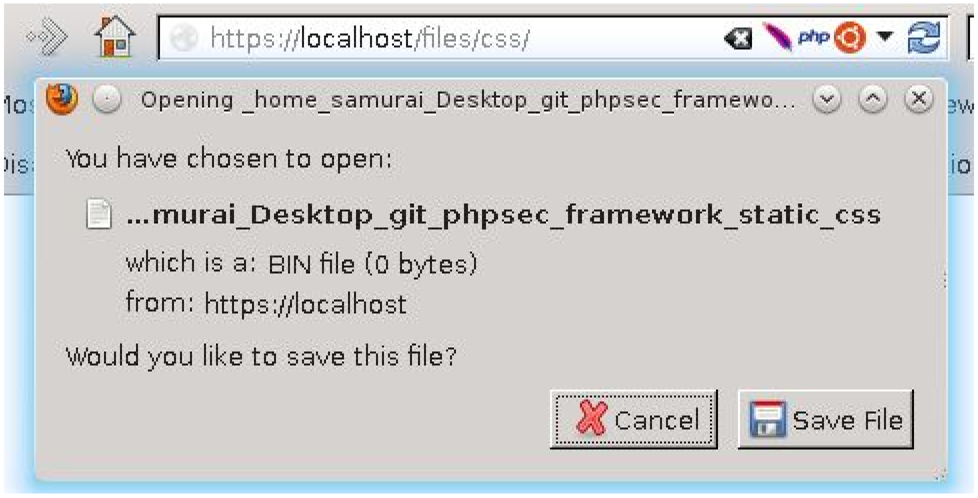
```
<LimitExcept POST GET>
»       Require valid-user
</LimitExcept>
```

Error Handling (4.9)

Analysis of Error Codes (OTG-ERR-001) and Analysis of Stack Traces (OTG-ERR-002)

File: `/var/www/https/framework/view/default/404.php`
File: `/var/www/https/framework/_core/front.php`
Line Number: 101
Description: In our last version of the application you got an empty page if you tried to access a non existing file/folder. So the page itself did not contain any sensitive data but its HTTP response attributes. It contains the following information: server os + version, web server used + version, web technology used (php) + version.

Another problem was that if you access an existing folder you could download an empty file. Its filename contained the physical path of the folder on the server.


```
https://localhost/files/css/
Opening _home_samurai_Desktop_git_phpsec_framewo...
You have chosen to open:
...murai_Desktop_git_phpsec_framework_static_css
which is a: BIN file (0 bytes)
from: https://localhost
Would you like to save this file?
```

We fixed this issue by providing a customized error page when you request an existing folder or a non existing file/folder. We also modify the HTTP attributes of the response (see above Enumerate Infrastructure and Application Admin Interfaces (OTG-CONFIG-005)) to avoid sensitive data exposure.

If the requested path isn't an existing file the Start()-function return false and the NotFound()-function will load the customized error page.

```
if (!is_file($path)) return false;
```
(front.php, line 101)

```
$FrontController=new FrontController();
if (!$FrontController->Start())
    $FrontController->NotFound();
```
(front.php, line 211 ff)

Cryptography (4.10)

Testing for Sensitive information sent via unencrypted channels (OTG-CRYPST-003)

Our application sends SCS PIN and Tanlist PIN via an unencypted email. This could be a security issue if an attacker manages to read the communication between the kangal bank server and the email server. In our case PGP could be used to encrypt the email, but PGP isn't very common and the users email program maybe doesn't support PGP. It is very unlikely that an attacker intercepts the communication or has access to the email account so we did not change our implementation. In a real life bank the PIN would be sent via Mail and used as a pre-shared secret between the bank and a customer.