Secure Coding - Team 7- Phase 5

Magnus Jahnen, Thomas Krex, Elias Tatros

January 26, 2015

Part I

Executive Summary

This is the final report for the Secure Coding bank web application project of Team 7. In the semester we had to develop a common bank web software with different user roles, transactions of money with different TAN methods and batch transaction via file upload.

The bank application was tested by two other teams. The aim of the tests was to find security vulnerabilities. After each test we fixed all vulnerabilities the team or we ourselves have found. As a plus we implemented new features or fixed other non security related bugs.

This final report first describes the architecture of our application. After that there is a brief overview of the security measures our system offers. The last part describes the recent fixes we have implemented.

Contents

Part II

Time Tracking Table

Chapter 1

Time Tracking Table

Time Tracking		
Name	**Time**	**Description**
Magnus Jahnen	2h	Lock out mechanism
	3h	Security Question and PW Recovery fix
	2h	Java-SCS sha256 and ProGuard obfuscation
	1h	C Parser sha256 and -fstack-protector-all and -fPIE
	4h	Trailer
	2h	Meetings
	4h	Report & Presentation
Thomas Krex	3h	Fixes for TAN Generation
	2h	Adjust session cookie settings
	3h	Trailer
	2h	Meetings
	5h	Documentation
Elias Tatros	5h	Presentation Layer Overhaul (CSS/HTML/JS)
	3h	Implementation of new functionality (user details, header renderer, bootstrap, JQuery)
	2h	Fixing bugs (information leaks, session management rework)
	3h	Analysis of existing system architecture (PHP/MySQL)
	2h	Creation of diagrams for architecture overhaul
	5h	Implementation and testing of restructured system
	2h	Documentation and writing report (architecture)

Part III

Application Architecture

Chapter 2

Architecture

In this section we take a look at the relationships and organizational structure of the internal and external components that make up the *myBank* banking application. The architecture description of our application defines several separate components and sub-components, each having clearly designed responsibilities. Interaction between components is controlled through the specification of interfaces. The goal of the design was to structure the application in such a way, that it would meet all requirements of the specification, while being well documented, maintainable and easily extensible in the future.

2.1 Three Layer Structure

Figure 2.1 shows the main structure of the myBank application. The *Smart-Card-Simulator* (*SCS*) was not included in this figure, since it is regarded as an independent external component, which does not interact with the main myBank system. From a high abstraction level view, the application is structured into a classic three layers architecture. The fundamental layer at the bottom of the hierarchy is the *Data Layer*. The main responsibility of this layer is to store and retrieve data. The *Data Layer* offers an interface for data storage and retrieval functions, which is used by the *Logic Layer*. This layer contains the "meat" of the banking application (*business logic*), meaning that it contains the central components of the system, which are handling all logical tasks and are not part of the supporting infrastructure. This includes components responsible for Transactions, User management, Tan generation, Email dispatch and so on. The logic layer in turn offers functionality through an interface, that is used by the *Presentation Layer* to display and manipulate information on the user level. In general we made sure, that components of each layer are only allowed to use the next lower layer through a specified interface, although our application does not strictly adhere to this in all cases.

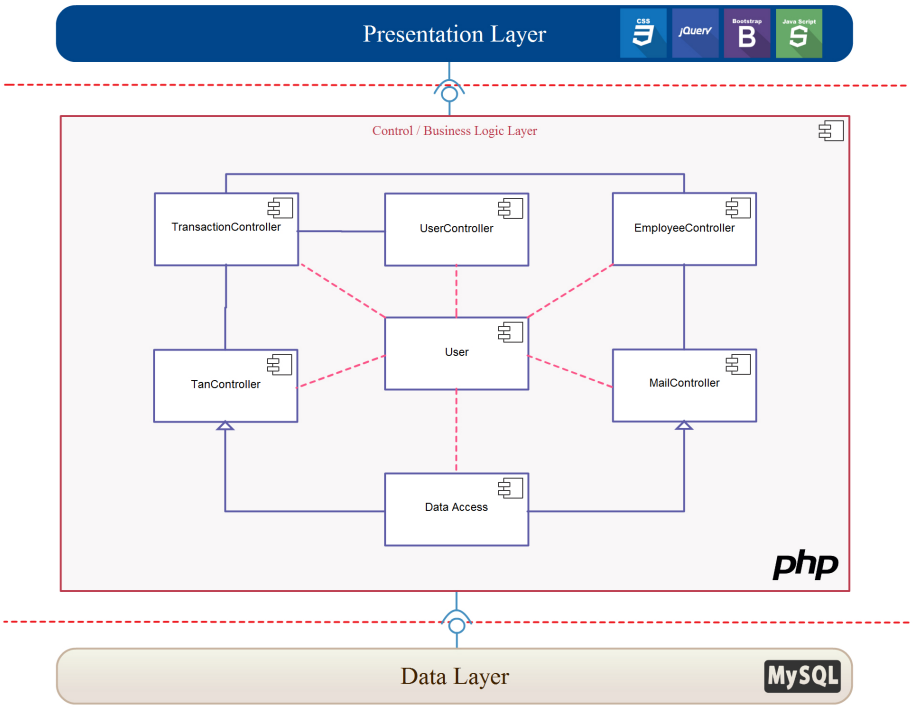

Figure 2.1: myBank Component Overview

2.2 Logic Layer Components: Separation of Concerns

The sub-components of the logic layer clearly encapsulate related functionality and are designed to function as independently as possible. Moreover, components and modules are clearly named according to their responsibilities. This makes a complex system easier to understand and limits the possibility for confusion, due to inappropriately named modules. The important components of the logic layer are:

- Data Access
- User
- User Controller
- Tan Controller
- Transaction Controller
- Employee Controller
- Mail Controller

The flow of data is generally designed to be from bottom to top, as indicated by the arrows in figure 2.1. Dependencies between components are designed to be minimal. Most components depend on a maximum of one or two other components, which also limits their communication channels, improving maintainability and ease of use. The following list gives some insight into the functionality that is encapsulated by each of the main logic layer components. We would like to note, that we realize, the architecture is not perfect and there is still room for improving the independence of components and their modules.

Data Access
: Abstracts access to the data layer. Main task is insertion and retrieval of data through communication with the mysql database.

User
: Abstracts a user or employee of the System. The myBank application is a user centric system, meaning that the user (customer or employee) is the central component of the system and is involved in almost all operations performed at the logic layer. Since the User class is used by nearly all sub-components, it is designed to be very concise and lightweight.

User Controller
: Offers functionality to verify user and employee credentials, as well as control over the lockout mechanisms.

Tan Controller
: Responsible for generation, verification, selection and maintenance of transaction numbers.

Transaction Controller

Implements functionality for credit transfers between users. Each transaction involves a large number of steps, including the validity of transaction data, verification of credentials, checking of available funds, selection of TANs and so on. These steps are abstracted into an easy to use interface by the Transaction Controller.

Employee Controller

Offers employee functionality, such as approval of users and transactions over the limit of 10000 Credits.

Mail Controller

As the name suggests, this component is tasked with sending out Emails to customers.

Figure 2.2 shows a more granular view of some system modules on the function level. The figure shows a functional hierarchy for parts of the Mail Controller, User Controller and Data Access. In the body of each function dependencies are listed, some of which are external and not included in this view to abstract from the complexity.

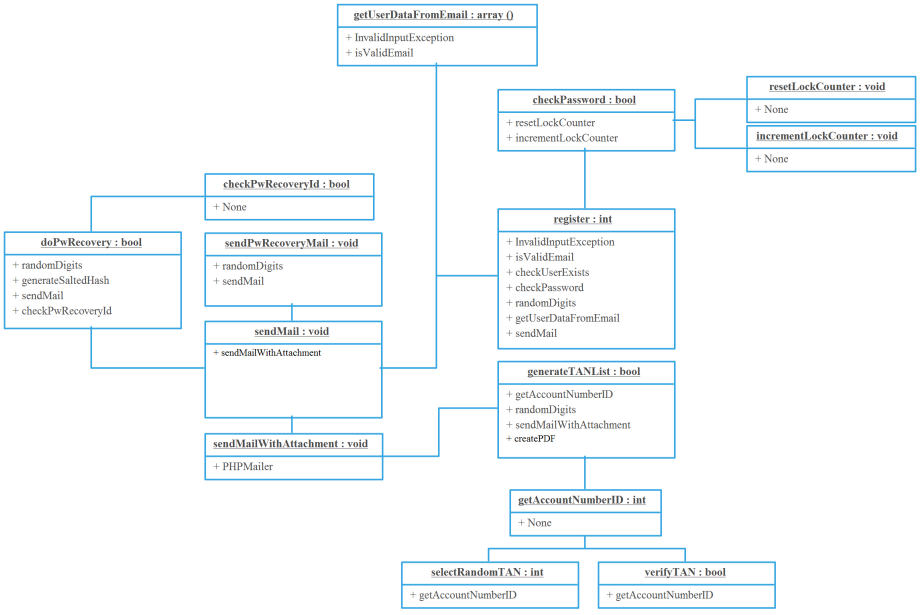

Figure 2.2: myBank Function Hierarchy Excerpt

2.3 Technologies & Languages

The myBank application makes use of several different technologies on each layer. The following is a list of technologies and programming/scripting languages utilized by layer.

Presentation Layer
HTML5, CSS3, JavaScript, Twitter Bootstrap (front-end framework), Yahoo! pure

Logic Layer
PHP 5.3/4, Java, C

Data Layer
MySQL

Part IV

Security Measures

Chapter 3

Security Measures

3.1 PDF Password Protection

The TAN List of our customers are password protected. That ensures that only customers themselves can access the transaction codes. To protect the pdfs we used the third party library FDPI.

3.2 HTTPS

Our webservice uses the HTTPS protocoll to encrypt all transmitted data to and from our clients. This prevents attackers from reading sensitive data via man-in-the-middle attacks.

3.3 Strong password policy

We enforce the user to choose a strong password to minimize the possible threat outgoing from brute force attacks.

3.4 Secure Session Management

The session cookie of an user is only transmitted via HTTPS, can not be accessed via javascript and expires after 30 minutes. So attackers cannot bypass the authentication schema by stealing cookies of logged in users.

3.5 Prepared Statements for SQL database

We protected our web service against sql injections by using prepared statements for database requests.

3.6 CSRF Token

3.7 Protection of C Parser against Heap/Buffer Overflows

To our knowledge there exist no possibilities for heap or stack overflows, since we tried to avoid allocating space on the heap and used secure functions for copying buffers. Also we add the compiler flag *-fPIE* and *-fstack-protector-all* in gcc to randomize address locations and a canary, making it harder for an attacker to cause buffer overflows.

3.8 Secure TAN Generation

To generate secure transaction codes for single or batch transfers, we use the hash algorithm sha256 in combination with a current timestamp as seeed. Time syncrhonization is done via a NTP server. The timestamp is concatenated with information of one transfer, including the destination account, the amount that will be transferred and the PIN of the user. A hash of this information is than trimmed to 15 digits. Since the timestamp is part of the TAN, one transaction code is only valid for at maximum 2 minutes, because the webservice only generates two tans for verifcation. One with the current timestamp and one with a timestamp one minute in future. Therefore transaction codes which an attacker stole, are useless in less than 2 minutes. That minimizes the risk of unauthorized transactions.

3.9 Secure Password Recovery

The Password Recovery for users is done in 3 steps. The user requests a new password at the website. Then he receives an email with a personalized URL. By clicking on it , the user gets to the page where he have to select one of the predefined security question and the corresponding answer that he specified at the registration. This procedure ensures that only the owner of the account can change the password.

3.10 Obfuscation of SCS

The Byte Code of the Java application SCS.jar was obfuscated with ProGuard in order to complicate the reverse engineering. In Listing 3.1, the corresponding ProGuard config is shown.

Listing 3.1: Configuration File of ProGuard

```
-injars /Volumes/MacintoshHD/Users/mep/Downloads/scs/scs.jar
-outjars /Volumes/MacintoshHD/Users/mep/Downloads/scs/scs_out.jar
```

```
-libraryjars
    /Library/Java/JavaVirtualMachines/jdk1.7.0_71.jdk/Contents/Home/jre/lib/rt.jar

-libraryjars
    /Library/Java/JavaVirtualMachines/jdk1.7.0_71.jdk/Contents/Home/jre/lib/jfxrt.jar

-dontshrink
-dontwarn com.javafx.**
-dontwarn org.apache.**

-keepattributes '*Annotation*'

-adaptresourcefilecontents **.fxml
-keepclassmembernames class * {
       @javafx.fxml.FXML *;
}

-keepclasseswithmembers public class com.javafx.main.Main, scs.Main {
       public *; public static *;
}


# Keep - Applications. Keep all application classes, along with their
    'main'
# methods.
-keepclasseswithmembers public class * {
       public static void main(java.lang.String[]);
}
```

3.11 Secure Handling of uploaded files

Uploaded Files for batch transaction are handled in a secure manner to avoid damage done by uploaded malicious code. This includes the usage of the default PHP temporary folder. All files are stored in /tmp and get a randomly chosen name. This has two effects. First of all files are not stored at the webroot and can not be access via an unsecure webserver. Secondly, renaming files before passing them to the c parser, prevents malicous code in the filename to be executed.
Another precoution we took, is to delete this files right after parsing them, so that malicous code can't be executed on our server later on.

3.12 Lock Out Mechanism

To avoid Brute Force attacks, we implemented an lock out mechanism at login and when entering TANs.

Hence the a user will be blocked after a certain count of invalid inputs. After that no further login is possible for the user and he must be approved again by an employee.

Part V

Fixes

Chapter 4

Fix of HTTP Strict Transport Security

4.1 Affected Files

/etc/apache2/httpd.conf: line 15

4.2 Description

We added an HTTP Scrict Transport Security Header to the 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:

Listing 4.1: Command for enabling HSTS

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

The directive "max-age" indicates the number of seconds that the browser should automatically convert all HTTP requests to HTTPS.

Chapter 5

Fix for Bypassing Session Management Schema and Cookies attributes

5.1 Affected Files

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

5.2 description

In order to protect the session cookie and its attributes against attackers the following settings were added 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 session cookie can be acessed by javascript. The http_secure flag ensures that session cookies are only sent via HTTPS.
Futhermore the attribute session cookie_lifetime defines the expire data for a session cookie. A short lifetime decreases the chances for an attacker to sucessfully use a foreign session cookie to authenticate with the web service.

Chapter 6

Fix Process Timing

6.1 Affected Files

- webroot/pw_recovery.php: 137++

6.2 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 instruction on how to change his 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 is responding 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 choose 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 6.1: 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);
}
```

Chapter 7

Fix of TAN Generation

7.1 Affected Files

- c_user.php : line 324++
- helper.php : line 216

7.2 Description

Our app offers the possibility to use a SmartCardSimulator to generate transaction codes for single or batch transactions. This code can then be entered at our website or in the batch transaction batch file. To verify the TAN, the webservice is using the same computations as the SCS. This includes the hashing of a string containing the current time as seed, the destination account, the amount and the pin of the user. This String was hashed with md5. This was a vulnerability because md5 is not secure in terms of collision attacks. So we changed the hashing algorithm to sha256 which generates a 256 bit long hash instead of 128 bit. The following transformations of the hashed string are required, because the c and java hash function are creating byte arrays. When converting this byte order to decimal numbers negative number can occur and will be converted to their absolute values. To get the same result in php the generated string, containing an hexadecimal number, has to be converted pairwisely and 127 has to substracted if the resulting number is greater tan 256.

Listing 7.1: Generation of a TAN using sha256

```
function generateTanWithSeed($seed,$pin,$destination,$amount){

$plaintext = $seed.$pin.$destination.$amount.$seed;
//$hash = $this->generateMD5Hash($plaintext);
```

```
$hash = hash('sha256',$plaintext);
$hash_array = array();
for ($i=0;$i<16;$i += 2) {
$tmp = substr($hash, $i,1).substr($hash, $i+1,1);
array_push($hash_array, hexdec($tmp));
}
$hash_string = "";
for ($j=0;$j<16;$j++) {
$tmp = $hash_array[$j];
if ($tmp > 127) {
$tmp -= 256;
}
$hash_string .= strval(abs($tmp));
}
return substr($hash_string,0,15);
}
```

The next problem was the unavailbility of the time server which was used to synchronize the SCS and the webservice. Also it ensures that transaction codes for an transaction with the same amount and destinations differs over the time. This results in a stronger protecion against brutforcing the transaction codes. To fix this problem we simply changed the time server from that the time is requested, shown in Listing **??**

Listing 7.2: New TimeServer IP

```
function getUTCTime(){
$timeserver = "129.6.15.28";
```

Chapter 8

Fix for Upload of Unexpected File Types

Our app allows the user to upload any kind file for the batch transaction. But in our point of view that is not a vulnerability. Uploaded Files are not stored in the webroot of the server and are renamed before they are stored. Furthermore uploaded files are directly deleted after being parsed by the c program. If an attacker woud have the possibility to execute his code, the extension of the file would not make any difference. To sum it up, allowing uploading any file extension increases the usabilty and has no influence on the security of our web service.

Chapter 9

Lock out Mechanism

To avoid brute forcing attacks when logging in and transferring money (TANs) we implemented a mechanism which blocks the user after 5 attempts with invalid input. Thus we created a new database field in the *users* table (lock_counter, int).

9.1 Affected Files

- c_user.php : line 898 new methods
- c_user.php : lines 455, 469, 877 method call resetLockCounter()
- c_user.php : lines 459, 472, 883 method call incrementLockCounter()

9.2 Additions

Listing 9.1: New methods

```
function incrementLockCounter() {
       $connection = new PDO( DB_NAME, DB_USER, DB_PASS );
       $sql = "SELECT lock_counter FROM users WHERE email = :email
           LIMIT 1";
       $stmt = $connection->prepare( $sql );
       $stmt->bindValue( "email", $this->email, PDO::PARAM_STR );
       $stmt->execute();
       $result = $stmt->fetch();
       if ($result['lock_counter'] < 5) {
              $sql = "update users set lock_counter = lock_counter + 1
                  where email = :email";
              $stmt = $connection->prepare( $sql );
              $stmt->bindValue( "email", $this->email, PDO::PARAM_STR );
              $stmt->execute();
```

```
                return false;
        } else {
                $sql = "update users set is_active = 0, lock_counter = 0
                    where email = :email";
                $stmt = $connection->prepare( $sql );
                $stmt->bindValue( "email", $this->email, PDO::PARAM_STR );
                $stmt->execute();
                return true;
        }

        $connection = null;
}
function resetLockCounter() {
        $connection = new PDO( DB_NAME, DB_USER, DB_PASS );
        $sql = "update users set lock_counter = 0 where email = :email";
        $stmt = $connection->prepare( $sql );
        $stmt->bindValue( "email", $this->email, PDO::PARAM_STR );
        $stmt->execute();
        $connection = null;
}
```

Chapter 10

Security Question for Password Recovery

In Phase 3 we developed a password recovery functionality, which sends the user a generated new password via email. Because the user had no opportunity to change the password it would be easy for an attacker to log into any account if he somehow could get the user's emails.

Therefore we included a security question in the registration process, the user must answer correctly when recovering his password.

We added two database fields into the *users* table. The number of the security question the user chose, can be between 0 and 4 and the answer of the question are saved in them.

10.1 Diff

Listing 10.1: c_user.php

```
line 26
+      public $securityQuestionNumber = null;
+      public $securityQuestionAnswer = null;
line 550
+
+             if( isset( $data['sec_q_number'] ) ) {
+                    $this->securityQuestionNumber = stripslashes(
    strip_tags( $data['sec_q_number'] ) );
+             } else {
+                    throw new InvalidInputException("Security Question
    not specified.");
+             }
+
+             if( isset( $data['sec_q_answer'] ) ) {
```

```
+                   $this->securityQuestionAnswer = stripslashes(
    strip_tags( $data['sec_q_answer'] ) );
+           } else {
+                   throw new InvalidInputException("Security Question
    Answer not specified.");
+           }

line 571
-                   $sql = "INSERT INTO users
    (email,name,passwd,is_employee,is_active,pin,use_scs) VALUES
    (:email,:name,:password,:isEmployee,:isActive,:pin,:use_scs)";
+                   $sql = "INSERT INTO users
    (email,name,passwd,is_employee,is_active,pin,use_scs,security_question_number,security_question_a
    VALUES
    (:email,:name,:password,:isEmployee,:isActive,:pin,:use_scs,:security_question_number,:security_qu

line 592
+                   $stmt->bindValue( "use_scs", $this->useScs,
    PDO::PARAM_STR );
+                   $stmt->bindValue( "security_question_number",
    $this->securityQuestionNumber, PDO::PARAM_STR );
+                   $stmt->bindValue( "security_question_answer",
    $this->securityQuestionAnswer, PDO::PARAM_STR );

line 745
-                   $sql = "SELECT id, name, use_scs, email, passwd,
    pin, BIN(`is_employee` + 0) AS `is_employee`, BIN(`is_active` + 0)
    AS `is_active`, pw_recover_id FROM users WHERE email = :email LIMIT
    1";
+                   $sql = "SELECT id, name, use_scs, email, passwd,
    pin, BIN(`is_employee` + 0) AS `is_employee`, BIN(`is_active` + 0)
    AS `is_active`, pw_recover_id, security_question_number,
    security_question_answer FROM users WHERE email = :email LIMIT 1";

 line 764
 +                   $this->securityQuestionNumber =
     $result['security_question_number'];
 +                   $this->securityQuestionAnswer =
     $result['security_question_answer'];

 line 790
 -                   $sql = "SELECT id, name, email, passwd, use_scs,
     pin, BIN(`is_employee` + 0) AS `is_employee`, BIN(`is_active` + 0)
     AS `is_active`, pw_recover_id FROM users WHERE id = :id LIMIT 1";
 +                   $sql = "SELECT id, name, email, passwd, use_scs,
     pin, BIN(`is_employee` + 0) AS `is_employee`, BIN(`is_active` + 0)
     AS `is_active`, pw_recover_id, security_question_number,
     security_question_answer FROM users WHERE id = :id LIMIT 1";

line 808
```

```
+                       $this->securityQuestionNumber =
    $result['security_question_number'];
+                       $this->securityQuestionAnswer =
    $result['security_question_answer'];

line 1195
-       public function doPwRecovery($id) {
+       public function checkPwRecoveryId($id) {

                if(!is_numeric($id))
                        return false;
-               if(strcmp($this->pwRecoverId, $id) == 0) {
-                       $newPassword = randomDigits(8);

-                       try {
-                               $connection = new PDO( DB_NAME, DB_USER,
    DB_PASS );
-                               $sql = "UPDATE users set passwd =
    :password, pw_recover_id='NULL' WHERE id = :id";
-
-                               $stmt = $connection->prepare( $sql );
-                               $stmt->bindValue( "password",
    generateSaltedHash($newPassword), PDO::PARAM_STR );
-                               $stmt->bindValue( "id", $this->id,
    PDO::PARAM_STR );
-                               $stmt->execute();
-
-                               $connection = null;
+               return strcmp($this->pwRecoverId, $id) == 0;
+       }
+
+       public function doPwRecovery($id, $postArray) {
+               if(!$this->checkPwRecoveryId($id))
+                       return false;
-                               // Send the mail
+               if(strcmp($this->securityQuestionAnswer,
    $postArray['sec_q_answer']) != 0) {
+                       return false;
+               }
+
+               try {
+                       $connection = new PDO( DB_NAME, DB_USER, DB_PASS );
+                       $connection->setAttribute( PDO::ATTR_ERRMODE,
    PDO::ERRMODE_EXCEPTION );
-                               $message= "your new Password is:
    $newPassword";
-
-                               $this->sendMail($this->email, $message,
    "Your new Password");
-
```

```
-                           return true;
+                   $sql = "update users set passwd = :password,
    pw_recover_id = NULL where id = :id";
} catch ( PDOException $e ) {
-                           echo "<br />Connect Error: ".
    $e->getMessage();
-                           return false;
-                   }
+                   $stmt = $connection->prepare( $sql );
+                   $stmt->bindValue( "password",
    generateSaltedHash($postArray['password']), PDO::PARAM_STR );
+                   $stmt->bindValue( "id", $this->id, PDO::PARAM_STR
    );
+                   $stmt->execute();

-           } else {
+                   return true;
+           } catch ( PDOException $e ) {
+                   echo "<br />Connect Error: ". $e->getMessage();
                    return false;
            }
-
       }
```

Listing 10.2: conf.php

```
line 9
$SECURITY_QUESTIONS = array("What is the first name of the person you
    first kissed?",
+                          "What is the last name of the teacher who
    gave you your first failing grade?",
+                          "What is the name of the place your wedding
    reception was held?",
+                          "In what city or town did you meet your
    spouse/partner?",
+                          "What was the make and model of your first
    car?" );
```

Listing 10.3: register.php

```
line 74
+                   <div class="pure-control-group">
+                           <label for="sec_q_number">Security Question
    (Needed for Password recovery)</label>
+                   <select id="sec_q_number" name="sec_q_number"
    size="1">
+                           <?php
+                                   $counter = 0;
+                                   foreach($SECURITY_QUESTIONS as
```

```
    $question) {
+                                       echo "<option
    value=\"$counter\">$question</option>";
+                                       $counter = $counter +1;
+                                   }
+                               ?>
+                           </select>
+                       </div>
+
+
+                       <div class="pure-control-group">
+                           <label for="sec_q_answer">Answer</label>
+                           <input name="sec_q_answer" id="sec_q_answer"
    type="text" placeholder="Answer" onkeyup="check_answer()" required>
+                           <b id=answer_info></b>
+                       </div>
+

line 175
+       function check_answer() {
+               var an_field = document.getElementById("sec_q_answer")
+               var an_info = document.getElementById("answer_info")
+
+               if (an_field.value.length >= 4) {
+                       submit_button.disabled = false;
+                       an_info.textContent = ""
+               } else {
+                       submit_button.disabled = true;
+                       an_info.textContent = "Answer must be at least 4
    characters"
+               }
+       }
+
```

Listing 10.4: pw_recovery.php

```
line 22
+
        $user = new User();
        $user->getUserDataFromEmail($_GET['email']);
+
+
+       if( isset($_POST['password'])) {
+               if($user->doPwRecovery($_GET['id'], $_POST))
+                       echo "Success, your password was set to the newly
    entered one!";
+               else
+                       echo "Something went wrong, please try again
    later!";
```

```
+
+               die();
+       }
+
        $success = false;
        if($user->email) {
                // if we found the mail it is a valid user
-               $success = $user->doPwRecovery($_GET['id']);
+               $success = $user->checkPwRecoveryId($_GET['id']);
        }?>

line 65
<p>
                        <?php
                        if($success) {
-                               echo "Your new password has been sent to
    you via email!";
+                       ?>
+                       <form method="post" action="" class="pure-form
    pure-form-aligned">
+                   <fieldset>
+                               <div class="pure-control-group">
+                               <label for="sec_q_number">Security
    Question:</label>
+                       <label name="sec_q_number" id="sec_q_number"><?php
    echo $SECURITY_QUESTIONS[$user->securityQuestionNumber] ?></label>
+                       </div>
+
+
+                       <div class="pure-control-group">
+                           <label for="sec_q_answer">Answer</label>
+                           <input name="sec_q_answer" id="sec_q_answer"
    type="text" placeholder="Answer" required>
+                       </div>
+
+                       <div class="pure-control-group">
+                           <label for="password">New Password</label>
+                           <input name="password" id="password"
    type="password" placeholder="**********" onkeyup="check_pw()"
    required>
+                           <b id=password_info></b>
+                       </div>
+
+                       <div class="pure-control-group">
+                           <label for="confirm_password">Confirm
    Password</label>
+                           <input name="confirm_password"
    id="confirm_password" type="password" placeholder="**********"
    onkeyup="check_pw()" required>
+                           <b id=confirm_password_info></b>
```

```
+                    </div>
+
+                    <div class="pure-controls">
+                        <button id="SendButton" type="submit"
    name="recoverPassword" class="pure-button pure-button-primary"
    onclick="setTimeout(disableFunction, 1);">Submit</button>
+                    </div>
+
+                </fieldset>
+                    </form>
+                    <?php

line 153
-      <script>
-              function disableFunction() {
-                  document.getElementById("SignInButton").disabled =
    'true';
-              }
-      </script>

line 213
+
+      <script>
+              function disableFunction() {
+                  document.getElementById("SignInButton").disabled =
    'true';
+              }
+
+              var pw_info = document.getElementById("password_info")
+      var confirm_pw_info =
    document.getElementById("confirm_password_info")
+
+      var pw_field = document.getElementById("password")
+      var confirm_pw_field =
    document.getElementById("confirm_password")
+
+      var submit_button = document.getElementById("SendButton")
+      submit_button.disabled = true
+
+      function check_pw() {
+              // lets check if the pw is strong enough
+              var pw = pw_field.value
+              var lowercase = pw.search("[A-Z]")
+              var uppercase = pw.search("[a-z]")
+              var number = pw.search("[0-9]")
+
+              if(pw.length < 8 || lowercase == -1 || uppercase == -1 ||
    number == -1) {
+                      pw_info.textContent = "Password must have at least
    eight characters, one upper case, one lower case letter and one
```

```
    number"
+                       submit_button.disabled = true
+               } else {
+                       pw_info.textContent = ""
+               }
+
+               if(confirm_pw_field.value != pw_field.value) {
+                       confirm_pw_info.textContent = "The two passwords
    do not match!"
+                       submit_button.disabled = true
+               } else {
+                       confirm_pw_info.textContent = ""
+                       submit_button.disabled = false
+               }
+       }
+
+       </script>
```