Team 20

Presented by

Simon Barth
Bharti Munjal
Michael Remmler

Use Cases From Phase 3	Implemented?
Encrypted TAN Delivery by email	YES
Download of SCS	YES
Transfer Using SCS	YES
Customer/Employee Password Recovery	YES
Employee able to initialize balance	YES
Transactions to Existing Accounts	YES
Account Number and amount visible to customer	YES
Batch Transfer allow multiple transfer	YES
All Required fields shown in transaction history	YES

Additional Features

Migration to phpsec
Session Timeout
“Remember Me” option
Account validation mail
Locking of account if multiple unsuccessful attempts are made
Ensuring Strong password
Prompt user to change password (If not changed for long time)

Use Cases From Phase 1	Implemented?
Customer / Employee registration (including sending e-mail with TANs)	YES
Customer / Employee login	YES
Customer / Employee logout	YES
Customer / Employee views bank account details of Customer	YES
Customer / Employee views transaction history of Customer	YES
Customer money transfer via HTML form (using TAN)	YES
Customer money transfer via uploading transaction batch file (using TAN)	YES
Employee approves transfers larger than 10.000 EUR	YES
Employee approves registration of Customer or of other employee	YES
(optional) Customer / Employee downloads transaction history of Customer as PDF document	YES

Live Demo ...

Time Tracking

Task	Student	Workload
Project Management	each	12h
phpsec, framework exploring and migration (create Transaction single/batch Tan/SCS, Encrypted PDF, Fixing vulnerabilities, ...)	Michael	20 + 60 = 80h
SCS (Java-Single, Batch), SCS (php-Single, Batch), Encrypted PDF, Batch transaction accepting tan from GUI, SCS Download, SCS Pin, PPT. phpsec migration (SCS Download, SCS Pin), Forgot password , Remember me in php sec, TAN Preference, Fixing vulnerabilities-xss,csrf, improved design of code)	Bharti	6h + 14h + 3h + 3h + 3h + 3h+3h+45+10h= 90h
Fixing php vulnerabilities, C Code fixes, phpsec migration	Simon	30h + 25h + 25h = 80h

Login Credentials

Role	Username/Email	Password	Account No
Customer1 (Approved, TAN)	user1	p@ssw0rd	4652813414
Customer2 (Approved, SCS)	user2	p@ssw0rd	8190647560
Customer 3 (Approved, First Login)	user3	p@ssw0rd	1865036925
Customer4 (Not Approved)	user4	p@ssw0rd	
Employee1 (Approved)	emp1	p@ssw0rd	
Employee2 (Not Approved)	emp2	p@ssw0rd	
Admin	admin	p@ssw0rd	
URL	https://<IP>/login		
TAN Infos of Customers			/home/samurai/secode20-bank/phpsec/SQL/Infos.txt

UseCases

Name	Forgot Password
Goal	Login even though password was forgotten
Actors	Registered Customer / Registered Employee
Pre-Conditions	Email-Id is registered.
Main Course of Execution	• Open bank site • Click on “Forgot Password” • Fill in Email Id [Email Id must be valid] • Click “Submit” button
Alternate Courses	-
Exceptions	Email Id is not registered at bank
Post-conditions	• User receives an email with a link • Click on the link. User will be taken to change password page
Data formats used	-
Additional Info	We found a severe issue in the standard implementation of this functionality. Via simple altering of a get parameter anybody could request a temporary password token for any account since the framework doesn’t check if the account belongs to the email adress.

Name	Encrypted Tan Delivery By Email
Goal	Receive Tans per Email in secured PDF
Actors	Newly Approved Customer
Pre-Conditions	• User is logged in • User chose Tan Preferences Tans
Main Course of Execution	• Login with a new user for first time • User will see UI for setting TAN Preference • Select Option “By mail” • Submit the form
Alternate Courses	-
Exceptions	-
Post-conditions	• A mail will be received with pdf as attachment • Second mail will be received with a Password for PDF
Data formats used	-

Name	SCS Download
Goal	Download SCS
Actors	Customer
Pre-Conditions	
Main Course of Execution	• Login a new client and set tan preference as “By SCS” • An option is visible on home page to download SCS and get SCS Pin • Click on “Download SCS” Link.
Alternate Courses	-
Exceptions	-
Post-conditions	• SCS jar (Simulator.jar) will be downloaded • Run java -jar Simulator.jar
Data formats used	-

Name	Request New SCS PIN
Goal	Get a new PIN if the old one was lost
Actors	Customer
Pre-Conditions	• Account is logged in • Tan preference for this user is SCS
Main Course of Execution	• Login with a client with tan preference as “By SCS” • An option is visible on home page to get new SCS Password. • Click on “Request SCS Pin” Link.
Alternate Courses	-
Exceptions	-
Post-conditions	• Receive email with new SCS Pin
Data formats used	-

Name	Transfer using SCS (Single Transaction)
Goal	Transfer Money in a single transaction with an SCS generated code
Actors	Approved Customer (With TAN Preference as SCS)
Pre-Conditions	Account is logged in
Main Course of Execution	● Click on “Create New Transaction” ● A transaction Code will be visible in Single Transaction Section ● Fill all the details for single transaction. ● Open SCS application (java -jar Simulator.jar) ○ Select Single Transaction ○ Enter SCS Pin Number (6 digits long) ○ Enter To Account ○ Enter Amount (xxx.xx format) ○ Enter transaction Code (20 characters long) ○ A Tan Number (64 digits) will be displayed ● Enter the generated Tan Number in field “Tan from SCS” ● Submit
Alternate Courses	-
Exceptions	● Wrong values are entered in Simulator ● Wrong values are entered at site form.
Post-conditions	Transaction will be successful

Name	Transfer Using SCS (Batch Transaction)
Goal	Transfer Money to several accounts using SCS generated code
Actors	Approved Customer (With TAN Preference as SCS)
Pre-Conditions	Account is logged in

Main Course of Execution	• Click on “Create New Transaction”
	• A transaction Code will be visible in batch transaction section.
	• Select the file.
	• Open SCS application (java -jar Simulator.jar)
	  ○ Select Batch Transaction
	  ○ Enter SCS Pin Number (6 digits long)
	  ○ Enter transaction Code (20characters long)
	  ○ Upload Batch File
	  ○ A Tan Number (64 digits) will be displayed
	• Enter the generated Tan Number in field “Tan from SCS”
	• Submit

Alternate Courses	
Exceptions	• Wrong values are entered in Simulator • Wrong values are entered at site form.
Post-conditions	• Transaction will be successful

Name	Session Timeout
Goal	Protect users from forgotten logout on public computers
Actors	Customer/Employee
Pre-Conditions	
Main Course of Execution	• Login a user • Wait for 10 minutes without any action/click • Click on any link say “Change Password”
Alternate Courses	
Exceptions	• User has not checked “Remember me” option.
Post-conditions	• User will be logged out

Name	Remember Me Option
Goal	Don’t get logged out automatically after timeout
Actors	Customer/Employee
Pre-Conditions	Account is logged in
Main Course of Execution	• Go to Bank Site • Enter login details and select “Remember me • Click on Submit • Wait for 10 minutes • Click on any link say “Change Password”
Alternate Courses	
Exceptions	
Post-conditions	• User will not be logged out

Name	Employee able to initialize balance
Goal	Set initial balance of customer
Actors	Employee
Pre-Conditions	A newly registered client say client1
Main Course of Execution	• Login as employee • Go to Approve User page • client1 should be visible . • A option to set initial balance should be visible • Fille the amount • Select the 2nd radio box • Submit
Alternate Courses	
Exceptions	
Post-conditions	• Login with client1 • client1 can see the initial amount

Name	Set TAN Preference
Goal	Customer able to tell his preference on first login
Actors	Newly approved Customer
Pre-Conditions	
Main Course of Execution	• Login with a new client • Client will see an option to select tan preference • Select By mail or By SCS • Submit
Alternate Courses	
Exceptions	

Post-conditions	• If “By mail” is selected
	  ○ A mail will be received containing pdf with TANs
	  ○ Another mail with password
	• If “By SCS” is selected
	  ○ A mail will be received with SCS Pin

Name	Lock account for Spurious attacks
Goal	Account get locked if multiple unsuccessful login attempts made
Actors	Customer/Employee
Pre-Conditions	
Main Course of Execution	• Go to Bank Site • Enter valid username • Enter wrong password • Press Submit • Repeat this for approx 6-7 times
Alternate Courses	
Exceptions	
Post-conditions	• User will get message “your account is locked” • User is no more able to login with correct password

Fixes (Reported Bugs)

Name	Fixed?
Clickjacking	YES
Session Management	YES
Reflected XSS	YES
Check on Number of Login Attempts	YES
Database structure visible on GUI	YES
CSRF protection	YES

Fixes (Reported Bugs) cont.

Name	Fixed?
HTTPS	YES
Authorization Bypass	YES
Strong Validations on Input Fields	YES

Clickjacking

This vulnerability was fixed by disallowing our site to be opened in any IFrame. This is done by inserting a java script in the header section of our site. The script checks if the page is opened in top window element or not. If not, it set the site as the top window element.

Script: We added the following script in header.php, CustomerHeader.php and EmployeeHeader.php

```
<style id="antiClickjack">body{display:none !important;}</style>
<script type="text/javascript">
        if (self === top) {
                var antiClickjack = document.getElementById
("antiClickjack");
                antiClickjack.parentNode.removeChild
(antiClickjack);
        } else {
                top.location = self.location;
        }
</script>
```

PhpSec

We switched our application to use the OWASP phpsec framework that helped us in solving the following security issues

- Strong password policy
    i. The password is considered as strong NOT on basis of a pattern rather the framework use a “heuristics and algorithmic” mechanism that ensure that users don’t choose weak, easily guessable passwords
- Session Management
    i. Duration of sessions are tracked. Session fixation is prevented.
- Multiple Login Attempts
    i. A user is blocked if multiple unsuccessful attempts are made for login
- Password retrieval

XSS

In order to make our site secure from XSS and CSRF attacks we escaped all input parameters using `htmlspecialchars` () function of php.

Changes are made in `DefaultController`.php to add a new function `SanitizeAllInputData (Line: 58)`

This function is then called by `BaseCustomerController.php (Line: 28)`, `BaseUserController.php (Line: 11 )` and `BaseEmployeesController.php (Line: 23 )` where they sanitize all POST and GET parameters of a request.

Database Structure visible on UI

Earlier the command that was used to run the batch program and related DB errors was shown on GUI and was disclosing a part of DB Schema to the user. Changes have been made to make the GUI more user friendly and hiding back end details.

- Change are made in CreateTransactionController.php to not display how the C++ program is called.
- C++ program was changed to catch exceptions
- Database stored procedure emits exceptions with defined text in cases of errors which can be directly displayed.

Unencrypted communication

In phase 1 all communication was made via the unencrypted HTTP protocol. Since everybody in public networks, especially wireless networks can eavesdrop this communication we reconfigured apache to use https only for our bank application.

Authorization Bypassing

In phase 1 there was a vulnerability where one employee could access the details of other employees by simply changing the URL. To prevent this we:

- avoided get parameters
- don’t expose internal details like userid to the website

CSRF Protection

The OWASP phpsec framework claims to provide CSRF protection classes for inserting tokens in forms. However this feature is one of the most recent commits in the phpsec git repository and not really usable. Therefore we were forced to write our own CSRF protection class in phpsec/libs/form/csrf/csrf.php.

We have a check in every controller that is processing input from forms for validation of this token. Additionally every form has a hidden field with this token.

Only forms that are accessible for logged in customers/employees are protected that way.

Thank You