Security testing report

Gianluca Bortoli
DISI - University of Trento
Student id: 179816
gianluca.bortoli@studenti.unitn.it

1. INTRODUCTION

This work aims at performing a security analysis study on *Schoolmate*. This web service is PHP/MySQL solution for elementary, middle and high schools where different type of users can manage all the needed information to fulfill their job.

This report is structured as follows. Section 2 describes the naming convention used for the test cases and how they are structured into packages. Section 3 describes the vulnerabilities that have been found on the application, their root causes and how to fix them. Section 4 reports the results of the proof-of-concept (PoC) attacks described in Section 2. Finally, Section 5 depicts the steps that have to be taken in order to be able to run all the tests from scratch, while Section 6 gives a general evaluation of the whole work.

2. SECURITY TEST CASES

Pixy is a scanner static code analysis tools that scans PHP applications for security vulnerabilities. This software is used in order to spot the possible vulnerabilities in *Schoolmate*, the subject web application.

The workflow followed developing this work is the following:

1. run Pixy on the application code to produce vulnerability reports regarding Cross Site Scripting attacks (XSS).

2. manually analyze all the reports to classify the vulnerabilities. Thus, each of them has to be categorized as **false positive**, **reflected XSS** or **stored XSS** (see *xss_classification.pdf attachment*).

3. write all the test cases for all the true positives (TP) found (ie. the vulnerabilities classified either as reflected or stored XSS) in the form of a PoC attack against the application under test.

4. check that every test case fails on the bugged application to demonstrate *Schoolmate* has such security vulnerabilities.

5. modify the *Schoolmate*'s PHP code to fix the vulnerabilities found in step 2.

6. run again the test suite and check all the tests pass, meaning that the PoC attack implemented in each test case is no longer possible.

7. run again Pixy on the fixed application source code to check the vulnerabilities are really fixed and making sure step 5 did not introduce any security breach.

The test suite developed to demonstrate that the application under revision has security concerns makes use of JWebUnit, a Java-based testing framework for web applications. Every Pixy report is identified by a number and may contain multiple vulnerabilities. The test suite's source code is structured as follows:

- the **src/main/java** folder contains a Java file for each Pixy report which has at least one TP. The test-case Java file naming follows the convention *Test<PixyReportNumber>.java* so that it is possible to easily match the test case with its Pixy report counterpart.

- the **src/main/java/common** folder contains three packages that implement utility functions that are available to all the test cases. These functions allows not to soil the actual test case implementation, separating all the sequences of operations that has to be done in many tests in order to navigate the web page depending on the situation.

A noteworthy aspect is how tests' assertions are written. When dealing with security PoC test cases, the security expert writes a test that is thought to fail if the vulnerability can be exploited and pass otherwise. Hence, in this scenario a failing test (and its assertion) means that the vulnerability is still present, while a passing one means that the vulnerability is not present any more.

3. SOURCE CODE FIXES

Listing 1: Fixing all the vulnerabilities in index.php

```php
<?php
foreach ($_POST as $k => $val) {
 $_POST[$k] =
    htmlentities($val, ENT_QUOTES, "UTF-8");
}
...
?>
```

4. TESTING OUTCOMES

5. PRELIMINARY STEPS

6. CONCLUSIONS