Security Testing

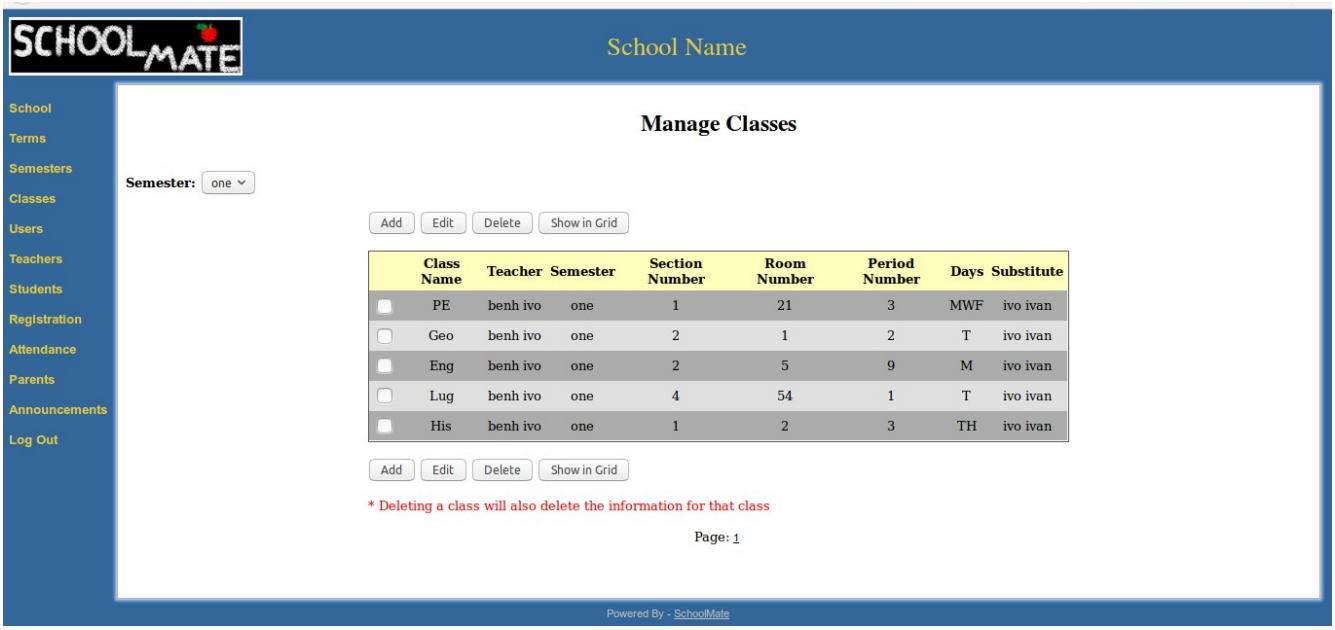
Kayongo Ivan
DISI - University of Trento
38050 Povo - Trento, Italy
email: benhurivo@gmail.com
ivan.kayongo@studenti.unitn.it

Introduction

The purpose of the project was to analyse 'schoolmate' a PHP/MySQL web application for elementary, middle and high schools; to identify the **Cross Site Scripting (XSS)** vulnerabilities it contains and fix them. Then, tests were carried out to verify if the vulnerabilities were fixed right. This was done using a tool called **JwebUnit**, a Java-based testing framework for web applications.

Procedure

The first step was to download, install and setup schoolmate. Below is an image of schoolmate project running

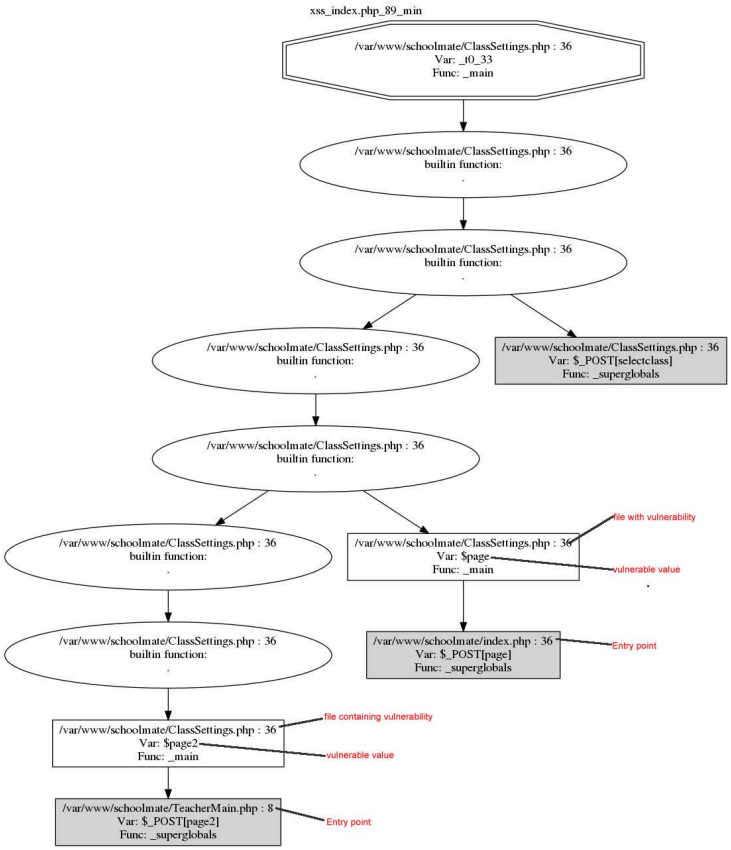


Schoolmate project setup and running

Schoolmate has four main users with different roles and permissions ie;

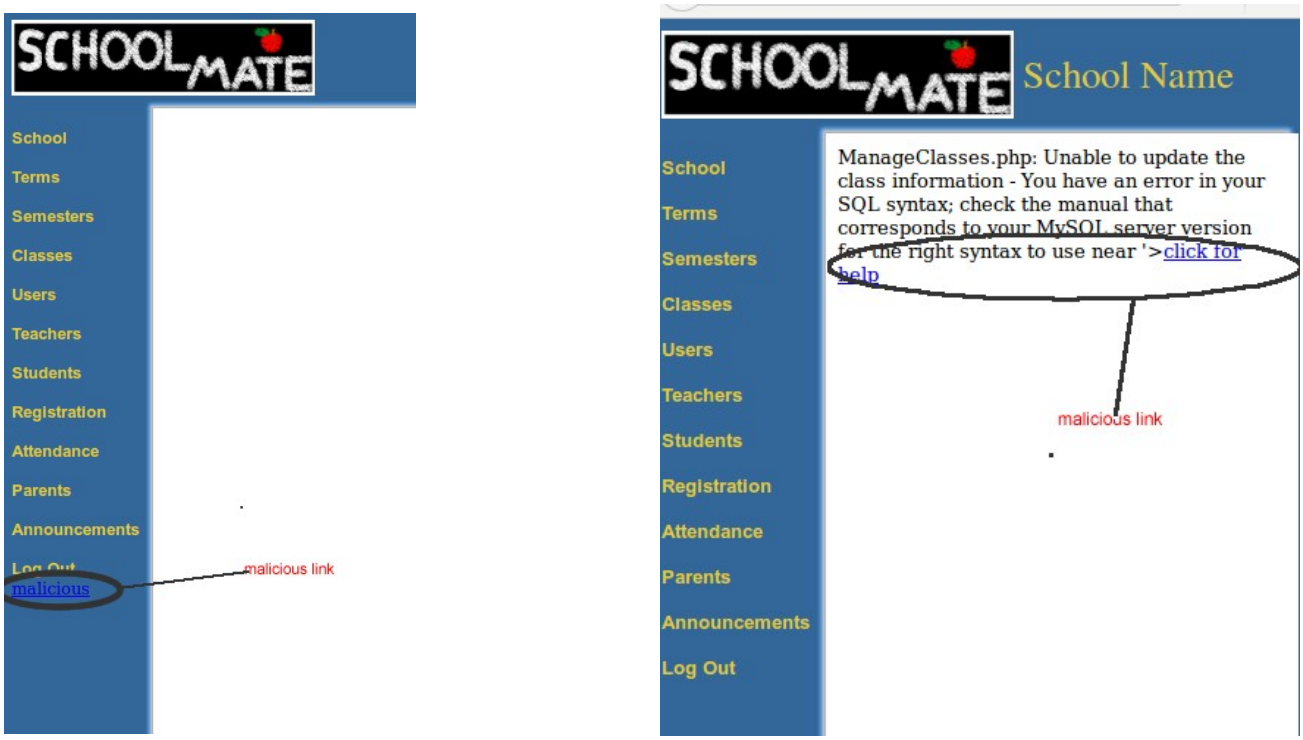
- Administrator; he has the most rights in the system. He can add,delete and modify entities in the system such as classes, terms, semesters,users and can also publish announcements and edit the school information.
- Teacher; he has access to his own classes and can edit them. He can also add assignments to the class' students and grade them. He can also view information about the school
- Parents can access information about their children in the school ie student's career and general information about the school
- Student; he has access to information about his career, classes, assignments and grades.

After downloading schoolmate and installing it, with everything set and running smoothly, I had to perform a PHP taint analysis on it. The taint analysis involves checking which variables can be modified by an external user / attacker to cause an attack or unwanted outcome.
For the taint analysis, another tool called **pixy** was used. It was run on the schoolmate project and it generated the tainted analysis reports out of which graphs of the tainted data flow where generated using **Graphviz**. These facilitated easy analysis of the report. Pixy generated 71 files out of which about 185 variables were identified and needed to be checked.
Below is a sample of one of the files generated; the first is the report from pixy and the second is the graph of the tainted data flow from the pixy report.



tainted analysis report from pixy ***graph of the tainted data flow***

Verifying the report from pixy
Having generated the graphs, the vulnerabilities reported had to be verified to find the true and false ones. To achieve this, I had to follow the tainted data flow graphs, analyzing the variables, checking the entry point and also framing real attacks using the generated graphs as a guide. These had the entry point at which the attack could be done and the affected value. I used another tool called '**webscarab**' which helped me intercept communication between the schoolmate user interface and the server side. Its through this that I was able to make cross site scripting (xss) injections to frame attacks to the system. These attacks could be in different forms. Below are images showing some of the attacks.



some of the possible xss injections to schoolmate

During the injection process, I discovered three kinds of xss attacks ie

- those whose values are intercepted as they are being posted from one page to another
- those intercepted and saved in the database, every time a user accesses this value from the database, the attack comes along. These can affect numerous users.
- those where attacker aims at breaking the sql syntax so that the xss injection could be thrown back with the error notification (second diagram).

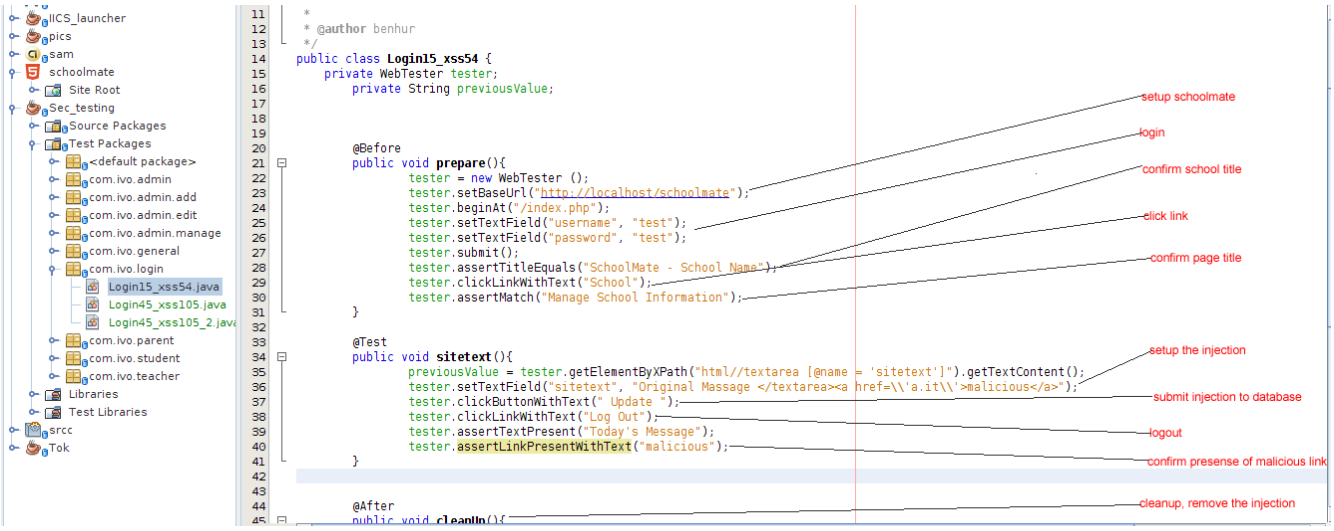
I then confirmed the **true** and **false positives** according to the flow of events, how variables are stored or handled ie if sanitized or not. I also framed xss injection attacks for the true positive variables. A table showing the full analysis is included in the appendix.

Automation

The next stage was to automate the attacks. I used **Netbeans IDE** and **JwebUni**t to achive this. I wrote ***security test cases*** for all the **true positive variables** I had identified. This involved writing a **java** class for each test which would follow the steps a normal user would go through while using the system and at some point create an xss injection at the vulnerable point identified by pixy. Then I also checked if the injection was successful or not ie if the malicious link had been successfully injected. A test case would run successfully if the requirements were met. An example of the test classes with brief explanation is shown below.

In the java test classes, there are annotations ie **@Before**, **@Test**,**@After** and these have different uses. Under the @Before annotation, I put a method in which I do prerequisites before the actual tests. The real tests happen in the methods under the @Test annotation. Here I performed the xss injection and confirmed its success or failure. In all my test classes, the method name corresponds to the name of the vulnerable variable I was testing.

Finally under the @After annotation are methods in which the system is restored to its original state ie before the injection happened.



One of the java test case classes showing automation of the attack

With the injection test cases for every true positive variable done, the next step was to fix the vulnerabilities.

Fixing vulnerabilities and testing

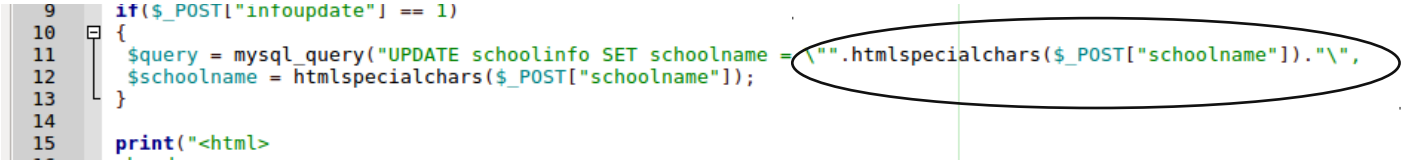
The process of fixing vulnerabilities is termed to as **sanitization**. This involves identifying the most appropriate position of the vulnerability and adding a statement or 'check' to ensure only wanted values are accepted to pass; this involved adding checks that only integers or words that don't contain HTML syntax characters are accepted.

The main vulnerabilities identified where that most of the values were posted and stored without being sanitized so a hacker could modify them using xss injections. I used two fixes for the vulnerabilities in the PHP code ie;

- **htmlspecialchars()** which converts some predefined characters to HTML entities thus they can not have any effect on the page eg < = ***<*** **,** >= ***>***
- **intval()** used to get the integer value of a variable

Having used these methods for the strings and integers variables respectively, I was able to sanitize the variables thus fixing the vulnerabilities.

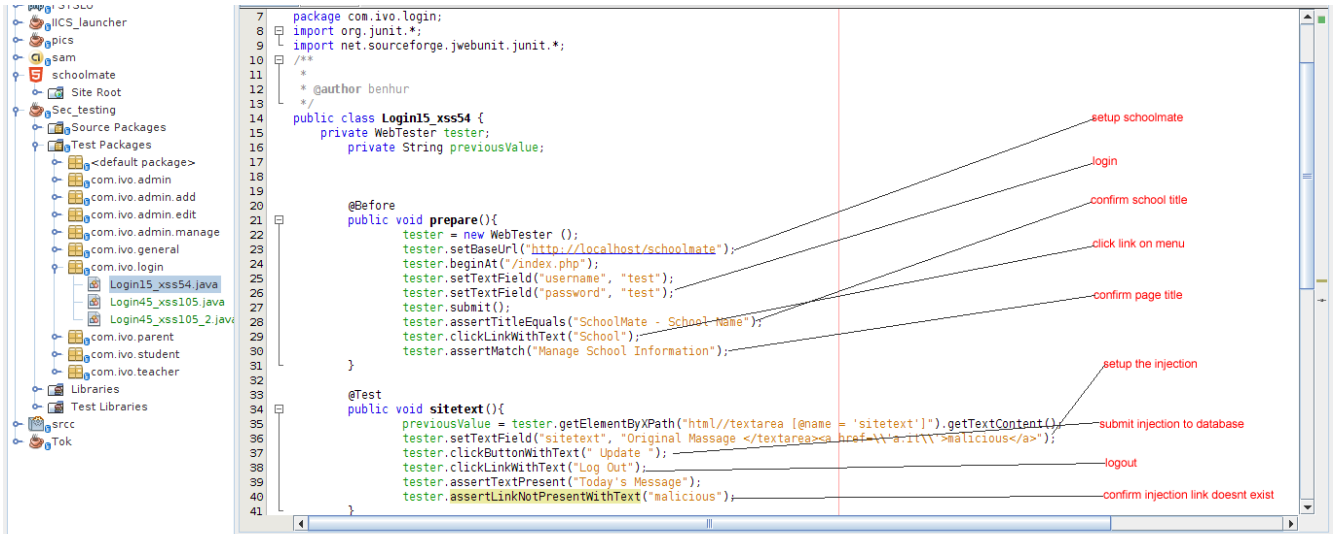
Eg for this sql update query, I used the htmlspecialchars() method for sanitizing



an example of the sanitization code

For the automated test cases, where I was expecting a malicious link, I had to change so that we expect no such link. In case of an anomaly, the test case would fail ie if we found a link.

All the test cases passed after fixing all the vulnerabilities. A java class showing the modification follows below



A java test case class validating a fixed vulnerability

Verifying the sanitization

After fixing the vulnerabilities, I had to test that they had been successfully removed and that no new vulnerabilities had been created. Finally I run pixy again on the fixed code of schoolmate project. This time pixy generated 14 tainted analysis reports unlike the first 71. All the 14 had variables which were false positives since on cross checking I realized the reported variables were already sanitized, thus I had successfully fixed all the vulnerabilities.

Appendix:

A full analysis of the report generated by pixy

ID	Page contained	variable	Entry point	True +ve	Reason
xss_index.php_194_min.dot	ParentMain.php : 93	$page2	ParentMain.php : 8	Y	
xss_index.php_115_min.dot	EditStudent.php : 51	$page2	AdminMain.php : 7	Y	
xss_index.php_85_min.dot	EditSemester.php : 43	$page2	AdminMain.php : 7	Y	
xss_index.php_212_min.dot	PointsReport.php : 121	$page2	AdminMain.php : 7	Y	
xss_index.php_92_min.dot	ManageSchoolInfo.php : 124	$page2	AdminMain.php : 7	Y	
xss_index.php_19_min.dot	AddTerm.php : 3	$page2	AdminMain.php : 7	Y	
xss_index.php_16_min.dot	AddAnnouncements.php : 3	$page2	AdminMain.php : 7	Y	This variable is posted from one page to another and stored in hidden textfields without any sanitization ie **<input type='hidden' name='page2' value='$page2'>**. It is possible for a hacker to intercept and modify the value thus the modification could be dangerous. He can include html scripts which break the hidden file and insert a malicious link in the page at which the variable is stored. I was able to make an injection of this malicious link '>malicious<br'
xss_index.php_90_min.dot	ParentViewStudents.php : 37	$page2	ParentMain.php : 8	Y	
xss_index.php_268_min.dot	ManageSemesters.php : 175	$page2	AdminMain.php : 7	Y	
xss_index.php_180_min.dot	TeacherMain.php : 83	$page2	TeacherMain.php : 8	Y	
xss_index.php_230_min.dot	VisualizeClasses.php : 138	$page2	AdminMain.php : 7	Y	
xss_index.php_260_min.dot	ManageTerms.php : 164	$page2	AdminMain.php : 7	Y	
xss_index.php_37_min.dot	EditAssignment.php : 27	$page2	TeacherMain.php : 8	Y	
xss_index.php_309_min.dot	ManageAssignments.php : 257	$page2	TeacherMain.php : 8	Y	
xss_index.php_146_min.	ViewAnnouncements.php	$page2	ParentMain.php : 8	Y	

dot	: 67			
xss_index.php_71_min.d				
ot	AddSemester.php : 28	$page2	AdminMain.php : 7	Y
xss_index.php_111_min.d				
ot	EditTeacher.php : 49	$page2	AdminMain.php : 7	Y
xss_index.php_41_min.d	EditAnnouncements.php :			
ot	23	$page2	AdminMain.php : 7	Y
xss_index.php_149_min.				
dot	EditUser.php : 68	$page2	AdminMain.php : 7	Y
xss_index.php_184_min.	ViewAssignments.php :			
dot	85	$page2	StudentMain.php : 8	Y
xss_index.php_161_min.				
dot	EditParent.php : 73	$page2	AdminMain.php : 7	Y
xss_index.php_269_min.				
dot	AddClass.php : 177	$page2	AdminMain.php : 7	Y
xss_index.php_87_min.d	ViewClassSettings.php :			
ot	36	$page2	ParentMain.php : 8	Y
xss_index.php_191_min.	DeficiencyReport.php :			
dot	90	$page2	AdminMain.php : 7	Y
xss_index.php_44_min.d				
ot	EditTerm.php : 23	$page2	AdminMain.php : 7	Y
xss_index.php_148_min.	ViewAnnouncements.php			
dot	: 67	$page2	TeacherMain.php : 8	Y
xss_index.php_126_min.				
dot	ViewCourses.php : 58	$page2	TeacherMain.php : 8	Y
xss_index.php_241_min.				
dot	GradeReport.php : 144	$page2	AdminMain.php : 7	Y
xss_index.php_93_min.d				
ot	AddParent.php : 39	$page2	AdminMain.php : 7	Y
xss_index.php_273_min.	ManageTeachers.php :			
dot	181	$page2	AdminMain.php : 7	Y
xss_index.php_316_min.				
dot	ManageGrades.php : 270	$page2	TeacherMain.php : 8	Y
xss_index.php_299_min.				
dot	Registration.php : 240	$page2	AdminMain.php : 7	Y
xss_index.php_181_min.				
dot	ViewStudents.php : 83	$page2	TeacherMain.php : 8	Y
xss_index.php_138_min.	StudentViewCourses.php			
dot	: 63	$page2	StudentMain.php : 8	Y
xss_index.php_70_min.d				
ot	AddStudent.php : 28	$page2	AdminMain.php : 7	Y
xss_index.php_18_min.d				
ot	AddUser.php : 3	$page2	AdminMain.php : 7	Y
xss_index.php_13_min.d				
ot	AddAttendance.php : 3	$page2	AdminMain.php : 7	Y
xss_index.php_186_min.				
dot	AdminMain.php : 87	$page2	AdminMain.php : 7	Y
xss_index.php_320_min.				
dot	ManageClasses.php : 303	$page2	AdminMain.php : 7	Y
xss_index.php_165_min.				
dot	StudentMain.php : 75	$page2	StudentMain.php : 8	Y
xss_index.php_147_min.	ViewAnnouncements.php			
dot	: 67	$page2	StudentMain.php : 8	Y
xss_index.php_141_min.				
dot	AddClass.php : 63	$page2	AdminMain.php : 7	Y
xss_index.php_238_min.	VisualizeRegistration.php			
dot	: 142	$page2	AdminMain.php : 7	Y
xss_index.php_272_min.	ManageAttendance.php :			
dot	181	$page2	AdminMain.php : 7	Y
xss_index.php_239_min.				
dot	EditClass.php : 142	$page2	AdminMain.php : 7	Y
xss_index.php_11_min.d				
ot	AddAssignment.php : 3	$page2	TeacherMain.php : 8	Y
xss_index.php_288_min.				
dot	ManageParents.php : 203	$page2	AdminMain.php : 7	Y
xss_index.php_293_min.	ManageStudents.php :			
dot	211	$page2	AdminMain.php : 7	Y
xss_index.php_76_min.d				
ot	EditGrade.php : 50	$page2	TeacherMain.php : 8	Y
xss_index.php_142_min.	ParentViewCourses.php :			
dot	64	$page2	ParentMain.php : 8	Y
xss_index.php_88_min.d	ViewClassSettings.php :			
ot	36	$page2	StudentMain.php : 8	Y
xss_index.php_183_min.	ViewAssignments.php :			
dot	85	$page2	ParentMain.php : 8	Y
xss_index.php_89_min.d				
ot	ClassSettings.php : 36	$page2	TeacherMain.php : 8	Y
xss_index.php_257_min.	ManageAnnouncements.p			
dot	hp : 161	$page2	AdminMain.php : 7	Y
xss_index.php_283_min.				
dot	ManageUsers.php : 188	$page2	AdminMain.php : 7	Y
xss_index.php_63_min.d				
ot	AddTeacher.php : 26	$page2	AdminMain.php : 7	Y
xss_index.php_200_min.	ViewGrades.php : 102	$page2	ParentMain.php : 8	Y

xss_index.php_201_min.dot	ViewGrades.php : 102	$page2	StudentMain.php : 8	Y
xss_index.php_194_min.dot	ParentMain.php : 93	$page	index.php : 36	Y
xss_index.php_115_min.dot	EditStudent.php : 51	$page	index.php : 36	Y
xss_index.php_85_min.dot	EditSemester.php : 43	$page	index.php : 36	Y
xss_index.php_212_min.dot	PointsReport.php : 121	$page	index.php : 36	Y
xss_index.php_92_min.dot	ManageSchoolInfo.php : 124	$page	index.php : 36	Y
xss_index.php_19_min.dot	AddTerm.php : 3	$page	index.php : 36	Y
xss_index.php_16_min.dot	AddAnnouncements.php : 3	$page	index.php : 36	Y
xss_index.php_90_min.dot	ParentViewStudents.php : 37	$page	index.php : 36	Y
xss_index.php_268_min.dot	ManageSemesters.php : 175	$page	index.php : 36	Y
xss_index.php_180_min.dot	TeacherMain.php : 83	$page	index.php : 36	Y
xss_index.php_230_min.dot	VisualizeClasses.php : 138	$page	index.php : 36	Y
xss_index.php_260_min.dot	ManageTerms.php : 164	$page	index.php : 36	Y
xss_index.php_37_min.dot	EditAssignment.php : 27	$page	index.php : 36	Y
xss_index.php_309_min.dot	ManageAssignments.php : 257	$page	index.php : 36	Y
xss_index.php_146_min.dot	ViewAnnouncements.php : 67	$page	index.php : 36	Y
xss_index.php_71_min.dot	AddSemester.php : 28	$page	index.php : 36	Y
xss_index.php_111_min.dot	EditTeacher.php : 49	$page	index.php : 36	Y
xss_index.php_41_min.dot	EditAnnouncements.php : 23	$page	index.php : 36	Y
xss_index.php_149_min.dot	EditUser.php : 68	$page	index.php : 36	Y
xss_index.php_184_min.dot	ViewAssignments.php : 85	$page	index.php : 36	Y
xss_index.php_161_min.dot	EditParent.php : 73	$page	index.php : 36	Y
xss_index.php_269_min.dot	AddClass.php : 177	$page	index.php : 36	Y
xss_index.php_87_min.dot	ViewClassSettings.php : 36	$page	index.php : 36	Y
xss_index.php_191_min.dot	DeficiencyReport.php : 90	$page	index.php : 36	Y
xss_index.php_44_min.dot	EditTerm.php : 23	$page	index.php : 36	Y
xss_index.php_148_min.dot	ViewAnnouncements.php : 67	$page	index.php : 36	Y
xss_index.php_126_min.dot	ViewCourses.php : 58	$page	index.php : 36	Y
xss_index.php_241_min.dot	GradeReport.php : 144	$page	index.php : 36	Y
xss_index.php_93_min.dot	AddParent.php : 39	$page	index.php : 36	Y
xss_index.php_273_min.dot	ManageTeachers.php : 181	$page	index.php : 36	Y
xss_index.php_316_min.dot	ManageGrades.php : 270	$page	index.php : 36	Y
xss_index.php_299_min.dot	Registration.php : 240	$page	index.php : 36	Y
xss_index.php_181_min.dot	ViewStudents.php : 83	$page	index.php : 36	Y
xss_index.php_138_min.dot	StudentViewCourses.php : 63	$page	index.php : 36	Y
xss_index.php_70_min.dot	AddStudent.php : 28	$page	index.php : 36	Y
xss_index.php_18_min.dot	AddUser.php : 3	$page	index.php : 36	Y
xss_index.php_13_min.dot	AddAttendance.php : 3	$page	index.php : 36	Y
xss_index.php_186_min.dot	AdminMain.php : 87	$page	index.php : 36	Y
xss_index.php_320_min.dot	ManageClasses.php : 303	$page	index.php : 36	Y
xss_index.php_165_min.	StudentMain.php : 75	$page	index.php : 36	Y

In the schoolmate project, the $page variable is used to identify which page has been called by the user. This value is posted from one page to another and is also store in hidden textfields without any sanitization ie **<input type='hidden' name='page' value='$page' />.** A hacker could intercept and modify this variable thus injecting an attack which could break the hidden field and print a malicious link. I was able to inject a malicious link ie **' >malicious<br'** which was displayed on the next page the variable is stored

xss_index.php_147_min.dot	ViewAnnouncements.php : 67	$page	index.php : 36	Y
xss_index.php_141_min.dot	AddClass.php : 63	$page	index.php : 36	Y
xss_index.php_238_min.dot	VisualizeRegistration.php : 142	$page	index.php : 36	Y
xss_index.php_272_min.dot	ManageAttendance.php : 181	$page	index.php : 36	Y
xss_index.php_239_min.dot	EditClass.php : 142	$page	index.php : 36	Y
xss_index.php_105_min.dot	Login.php : 45	$page	index.php : 36	Y
xss_index.php_11_min.dot	AddAssignment.php : 3	$page	index.php : 36	Y
xss_index.php_288_min.dot	ManageParents.php : 203	$page	index.php : 36	Y
xss_index.php_293_min.dot	ManageStudents.php : 211	$page	index.php : 36	Y
xss_index.php_76_min.dot	EditGrade.php : 50	$page	index.php : 36	Y
xss_index.php_142_min.dot	ParentViewCourses.php : 64	$page	index.php : 36	Y
xss_index.php_88_min.dot	ViewClassSettings.php : 36	$page	index.php : 36	Y
xss_index.php_183_min.dot	ViewAssignments.php : 85	$page	index.php : 36	Y
xss_index.php_89_min.dot	ClassSettings.php : 36	$page	index.php : 36	Y
xss_index.php_257_min.dot	ManageAnnouncements.php : 161	$page	index.php : 36	Y
xss_index.php_283_min.dot	ManageUsers.php : 188	$page	index.php : 36	Y
xss_index.php_63_min.dot	AddTeacher.php : 26	$page	index.php : 36	Y
xss_index.php_200_min.dot	ViewGrades.php : 102	$page	index.php : 36	Y
xss_index.php_201_min.dot	ViewGrades.php : 102	$page	index.php : 36	Y

xss_index.php_194_min.dot	ParentMain.php : 93	$_POST[selectclass]	ParentMain.php : 93	Y
xss_index.php_180_min.dot	TeacherMain.php : 83	$_POST[selectclass]	TeacherMain.php : 83	Y
xss_index.php_37_min.dot	EditAssignment.php : 27	$_POST[selectclass]	EditAssignment.php : 27	Y
xss_index.php_309_min.dot	ManageAssignments.php : 257	$_POST[selectclass]	ManageAssignments.php : 257	Y
xss_index.php_184_min.dot	ViewAssignments.php : 85	$_POST[selectclass]	ViewAssignments.php : 85	Y
xss_index.php_87_min.dot	ViewClassSettings.php : 36	$_POST[selectclass]	ViewClassSettings.php : 36	Y
xss_index.php_316_min.dot	ManageGrades.php : 270	$_POST[selectclass]	ManageGrades.php : 270	Y
xss_index.php_181_min.dot	ViewStudents.php : 83	$_POST[selectclass]	ViewStudents.php : 83	Y
xss_index.php_165_min.dot	StudentMain.php : 75	$_POST[selectclass]	StudentMain.php : 75	Y
xss_index.php_11_min.dot	AddAssignment.php : 3	$_POST[selectclass]	AddAssignment.php : 3	Y
xss_index.php_76_min.dot	EditGrade.php : 50	$_POST[selectclass]	EditGrade.php : 50	Y
xss_index.php_88_min.dot	ViewClassSettings.php : 36	$_POST[selectclass]	ViewClassSettings.php : 36	Y
xss_index.php_183_min.dot	ViewAssignments.php : 85	$_POST[selectclass]	ViewAssignments.php : 85	Y
xss_index.php_89_min.dot	ClassSettings.php : 36	$_POST[selectclass]	ClassSettings.php : 36	Y
xss_index.php_200_min.dot	ViewGrades.php : 102	$_POST[selectclass]	ViewGrades.php : 102	Y
xss_index.php_201_min.dot	ViewGrades.php : 102	$_POST[selectclass]	ViewGrades.php : 102	Y

The variable **$_POST[selectclass]** is posted from different pages and stored in a hidden textfield ie **<input type='hidden' name='selectclass' value='$_POST[selectclass]' />** without any sanitisation. A hacker could intercept and modify this variable thus injecting an attack. I was able to inject a malicious link ie **' >malicious<br'** which was displayed on the next page the variable is stored

xss_index.php_194_min.dot	ParentMain.php : 93	$_POST[student]	ParentMain.php : 93	Y
xss_index.php_13_min.dot	AddAttendance.php : 3	$_POST[student]	AddAttendance.php : 3	Y
xss_index.php_142_min.dot	ParentViewCourses.php : 64	$_POST[student]	ParentViewCourses.php : 64	Y

The variable **$_POST[student]** is posted from different pages and stored in a hidden textfield ie **<input type='hidden' name='student' value='$_POST[student]' />** without any sanitisation. A hacker could intercept and modify this variable thus injecting an attack which could break the hidden field and print a malicious link. I was able to inject a malicious link ie **' >malicious<br'** which was displayed on the next page the variable is stored

xss_index.php_115_min.dot	EditStudent.php : 51	$id[0] /$_POST[delete]	EditStudent.php : 2	Y	
xss_index.php_85_min.dot	EditSemester.php : 43	$id[0]	EditSemester.php : 2	Y	
xss_index.php_37_min.dot	EditAssignment.php : 27	$id[0]	EditAssignment.php : 2	Y	The variable **$id[0]** comes from another variable **$_POST["delete"]** which is an array containing a number of variables. When the variables are being added to the array, they are not sanitised, and thus the first variable ie **$id[0]** is not sanitised too. It is then stored in hidden textfields eg **<input type='hidden' name='announcementid' value='$id[0]'>** from where it is used. It is possible for a hacker to intercept the values, modify them and launch an xss attack on the page. I was able to insert a malicious link to the page using **'>malicious<br'**
xss_index.php_111_min.dot	EditTeacher.php : 49	$id[0]	EditTeacher.php : 2	Y	
xss_index.php_41_min.dot	EditAnnouncements.php : 23	$id[0]	EditAnnouncements.php : 2	Y	
xss_index.php_149_min.dot	EditUser.php : 68	$id[0]	EditUser.php : 2	Y	
xss_index.php_161_min.dot	EditParent.php : 73	$id[0]	EditParent.php : 2	Y	
xss_index.php_44_min.dot	EditTerm.php : 23	$id[0]	EditTerm.php : 2	Y	
xss_index.php_239_min.dot	EditClass.php : 142	$id[0]	EditClass.php : 2	Y	
xss_index.php_76_min.dot	EditGrade.php : 50	$id[0]	EditGrade.php : 2	Y	
xss_index.php_92_min.dot	ManageSchoolInfo.php : 37	$schoolname		N	
xss_index.php_3_min.dot	maketop.php : 2	$schoolname		N	
xss_index.php_53_min.dot	header.php : 15	$schoolname		N	This is a false positive because the variable comes from the database and it is sanitised imediately before it is displayed on the page ie **$schoolname = htmlspecialchars($_POST["schoolname"])**
xss_index.php_10_min.dot	maketop.php : 2	$schoolname		N	
xss_index.php_6_min.dot	maketop.php : 2	$schoolname		N	
xss_index.php_2_min.dot	maketop.php : 2	$schoolname		N	
xss_index.php_4_min.dot	maketop.php : 2	$schoolname		N	
xss_index.php_92_min.dot	ManageSchoolInfo.php : 37	$address		Y	This variable comes from the database and is displayed on the page without any sanitization. It is also saved into the database without any sanitization, this means an injection can be made and it will be saved to the database and will come up everytime the user accesses this variable. I was able to inject an image ie **\'><img src=\'a.it\'**
xss_index.php_92_min.dot	ManageSchoolInfo.php : 37	$phone		Y	Just like the previous variable, the **$phone** variable comes from the database and is displayed on the page without any sanitization. It is also saved into the database without any sanitization, this means an injection can be made and it will be saved to the database and will come up everytime the user accesses this variable. I was able to inject an image ie **\'><img src=\'a.it\'**
xss_index.php_92_min.dot	ManageSchoolInfo.php : 37	$numsemesters		N	Much as the **$numsemesters** variable comes from the database without sanitization and can be saved without any sanitization, it is impossible to store an injection because the variable is stored as an **integer** in the database with only **length of 3 characters**
xss_index.php_92_min.dot	ManageSchoolInfo.php : 37	$numperiods		N	Just like the **$numsemesters** variable, **$numperiods** comes from the database without sanitization and can be saved without any sanitization, but it is impossible to store an injection because the variable is stored as an **integer** in the database with only **length of 3 characters**
xss_index.php_268_min.dot	ManageSemesters.php : 175	$_POST[onpage]	ManageSemesters.php : 175	Y	
xss_index.php_260_min.dot	ManageTerms.php : 164	$_POST[onpage]	ManageTerms.php : 164	Y	
xss_index.php_309_min.dot	ManageAssignments.php : 257	$_POST[onpage]	ManageAssignments.php : 257	Y	
xss_index.php_146_min.dot	ViewAnnouncements.php : 67	$_POST[onpage]	ViewAnnouncements.php : 67	Y	
xss_index.php_184_min.dot	ViewAssignments.php : 85	$_POST[onpage]	ViewAssignments.php : 85	Y	The variable **$_POST[onpage]** is posted from different pages and stored in a hidden textfield ie **<input type='hidden' name='onpage' value='$_POST[onpage]'>** without any sanitisation. A hacker could intercept and modify this variable thus injecting an attack which could break the hidden field and print a malicious link. I was able to inject a malicious link ie **' >malicious<br'** which was displayed on the next page the variable is stored
xss_index.php_148_min.dot	ViewAnnouncements.php : 67	$_POST[onpage]	ViewAnnouncements.php : 67	Y	
xss_index.php_273_min.dot	ManageTeachers.php : 181	$_POST[onpage]	ManageTeachers.php : 181	Y	
xss_index.php_320_min.dot	ManageClasses.php : 303	$_POST[onpage]	ManageClasses.php : 303	Y	
xss_index.php_147_min.dot	ViewAnnouncements.php : 67	$_POST[onpage]	ViewAnnouncements.php : 67	Y	
xss_index.php_288_min.dot	ManageParents.php : 203	$_POST[onpage]	ManageParents.php : 203	Y	
xss_index.php_293_min.	ManageStudents.php :	$_POST[onpa	ManageStudents.php :	Y	

dot	211	ge]	211		
xss_index.php_183_min.dot	ViewAssignments.php : 85	$_POST[onpage]	ViewAssignments.php : 85	Y	
xss_index.php_257_min.dot	ManageAnnouncements.php : 161	$_POST[onpage]	ManageAnnouncements.php : 161	Y	
xss_index.php_283_min.dot	ManageUsers.php : 188	$_POST[onpage]	ManageUsers.php : 188	Y	
xss_index.php_234_min.dot	ManageSemesters.php : 139	$term		Y	The variable **$term** is assigned from the database by selecting the **title** from the **terms table**. The variable **title** is stored in the database without any sanitization and is retrieved still without any, thus a malicious script could easily be stored and displayed on a user's page. I was able to create an injection using **mal**
xss_index.php_30_min.dot	ViewAssignments.php : 9	$coursename		Y	**The variable $coursename** is assigned from the database by selecting the **coursename** from the **courses table**. The variable is stored in the database without any sanitization and is retrieved still without any, thus a malicious script could easily be stored and displayed on a user's page. I was able to create an injection using **PE\'>malicious<br\'**
xss_index.php_31_min.dot	ViewAssignments.php : 9	$coursename		Y	
xss_index.php_207_min.dot	ManageAssignments.php : 115	$coursename		Y	
xss_index.php_269_min.dot	AddClass.php : 177	$_POST[fullyear]	AddClass.php : 177	Y	The variable **$_POST[fullyear]** is posted from a page and stored in a hidden textfield ie **<input type='hidden' name='fullyear' value='$_POST[fullyear]' />** without any sanitisation. A hacker could intercept and modify this variable thus injecting an attack which could break the hidden field and print a malicious link. I was able to inject a malicious link ie **' >malicious<br'** which was displayed on the next page the variable is stored
xss_index.php_13_min.dot	AddAttendance.php : 3	$_POST[semester]	AddAttendance.php : 3	Y	The variable **$_POST[semester]** is posted from a page and stored in a hidden textfield ie **<input type='hidden' name='semester' value='$_POST[semester]' />** without any sanitisation. A hacker could intercept and modify this variable thus injecting an attack which could break the hidden field and print a malicious link. I was able to inject a malicious link ie **' >malicious<br'** which was displayed on the next page the variable is stored
xss_index.php_105_min.dot	Login.php : 45	$message		Y	The **$address** variable comes from the database and is displayed on the page without any sanitization. It is also saved into the database without any sanitization, this means an injection can be made and it will be saved to the database and will come up everytime the user accesses this variable. I was able to create an injection using **</textarea>malicious**
xss_index.php_76_min.dot	EditGrade.php : 50	$_POST[assignment]	EditGrade.php : 50	Y	The variable **$_POST[assignment]** is posted from a page and stored in a hidden textfield ie **<input type='hidden' name='assignment' value='$_POST[assignment]' />** without any sanitisation. A hacker could intercept and modify this variable thus injecting an attack which could break the hidden field and print a malicious link. I was able to inject a malicious link ie **' >malicious<br'** which was displayed on the next page the variable is stored
xss_index.php_54_min.dot	Login.php : 15	$text		Y	**The** variable **$text** is assigned from the database by selecting the **sitetext** from the **schoolinfo table**. The variable **sitetext** is stored in the database without any sanitization and is retrieved still without any, thus a malicious script could easily be stored and displayed on a user's page. I was able to create an injection using **</textarea>malicious**