Schoolmate Vulnerabilities

Report ID	File	Variables/Input	Classification	Explanation	Proof-of-concept Attack	Fix
2, 3, 4, 6, 10	maketop.php	$schoolname	False Positive	Data is pulled without validation from DB. It is sanitized, however, during input to DB.		No action required.
53	header.php	$schoolname	False Positive	The schoolname value is assumed to be pre-existant (non-tainted) in the DB, then is only updated via a query that sanitizes the value. This is a false positive.		
321	ReportCards.php	$data	False Positive	This vulnerability is not actually exploitable, since the potentially tainted data is written to a PDF.		
16	AddAnnouncements.php	$page, $page2	True / Reflected	Vulnerable variables are hidden in the form itself, but can be intercepted, tampered with, and are later used without sanitization.	- Login as schoolmate/test - Go to Announcements, press Add - Intercept request with WebScarab - Use payloads for 11.1 or 13.2 on the respective vars	Sanitize page and page2 variables through intval. This can be done since the software assumes that all page and page2 variables assume integer values (page IDs).
16.1		$_POST["page"]	True / Reflected		//take existing page value e.g. 2, then add '><a href="http://unitn.it">XSS on page</a><br'	AddAnnouncements.php line 3: - <input type='hidden' name='page' value='$page'> + <input type='hidden' name='page' value='".intval($page)."'>
16.2		$_POST["page2"]	True / Reflected		//take existing page2 value e.g. 2, then add '><a href="http://unitn.it">XSS on page2</a><br'	AddAnnouncements.php line 3: - <input type='hidden' name='page2' value='$page2'> + <input type='hidden' name='page2' value='".intval($page2)."'>
18	AddUser.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
19	AddTerm.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
63	AddTeacher.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
70	AddStudent.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
71	AddSemester.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
90	ParentViewStudents.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
93	AddParent.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
126	ViewCourses.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
138	StudentViewCourses.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
141	AddClass.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
186	AdminMain.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
191	DeficiencyReport.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
212	PointsReport.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
230	VisualizeClasses.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
238	VisualizeRegistration.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
241	GradeReport.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
272	ManageAttendance.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
299	Registration.php	$page, $page2	True / Reflected	as 16	as 16, but on different page and/or different user	as 16
11	AddAssignment.php	$page,$page2, $selectclass	True / Reflected	as 16, also applies to the selectclass variable. Since selectclass is used in an SQL query, the query must first be ended with --	as 16, but on different page and/or different user and with the addition of the selectclass variable	as 16. selectclass can also be sanitized with intval since a class ID is assumed to be an int by the software
11.1		$_POST["page"]	True / Reflected	as 16.1	as 16.1	as 16.1
11.2		$_POST["page2"]	True / Reflected	as 16.2	as 16.2	as 16.2
11.3		$_POST["selectclass"]	True / Reflected		//take existing selectclass value e.g. 1, then add -- '><a href="http://unitn.it">XSS on selectclass</a><br'	Enclose all selectclass occurrences in intval, both in forms and SQL queries AddAssignment.php line 3: - <input type='hidden' name='selectclass' value='$_POST[selectclass]' /> + <input type='hidden' name='selectclass' value='.intval($_POST[selectclass]
87, 88	ViewClassSettings.php	$page, $page2, $selectclass	True / Reflected	as 11		as 11
89	ClassSettings.php	$page, $page2, $selectclass	True / Reflected	as 11		as 11
165	StudentMain.php	$page, $page2, $selectclass	True / Reflected	as 11		as 11
180	TeacherMain.php	$page, $page2, $selectclass	True / Reflected	as 11		as 11
181	ViewStudents.php	$page, $page2, $selectclass	True / Reflected	as 11		as 11
200, 201	ViewGrades.php	$page, $page2, $selectclass	True / Reflected	as 11		as 11
316	ManageGrades.php	$page, $page2, $selectclass	True / Reflected	as 11		as 11
41	EditAnnouncements.php	$page, $page2, $delete	True / Reflected	as 16, also applies to the delete variable. Since delete is used in an SQL query, the query must first be terminated with --)	- Login as schoolmate/test - Go to Announcement - Select an announcement, press Edit - Intercept request with WebScarab - use payloads on any of the tainted vars	as 16, and sanitize the remaining variable $id[0] using intval. replacing all its occurences both in the form and in the SQL queries
41.1		$_POST["page"]	True / Reflected	as 16.1	as 16.1	as 16.1
41.2		$_POST["page2"]	True / Reflected	as 16.2	as 16.2	as 16.2

Schoolmate Vulnerabilities

Report ID	File	Variables/Input	Classification	Explanation	Proof-of-concept Attack	Fix
41.3		$_POST["delete"]			`//take existing delete value e.g. 1, then add` ` --) '><a href="http://unitn.it">XSS on selectclass</a> '`	`EditAnnouncements line 23:` `-  <input type='hidden' name='announcementid' value='$id[0]'>` `+  <input type='hidden' name='announcementid' value='".intval($id[0])."'>` `EditAnnouncements line 2:` `- $id = $_POST["delete"];` `+ $id = intval($_POST["delete"]);`
44	EditTerm.php	$page, $page2, $delete	True / Reflected	as 41	as 41 but on different page and/or different user	as 41
85	EditSemester.php	$page, $page2, $delete	True / Reflected	as 41	as 41 but on different page and/or different user	as 41
111	EditTeacher.php	$page, $page2, $delete	True / Reflected	as 41	as 41 but on different page and/or different user	as 41
115	EditStudent.php	$page, $page2, $delete	True / Reflected	as 41	as 41 but on different page and/or different user	as 41
149	EditUser.php	$page, $page2, $delete	True / Reflected	as 41	as 41 but on different page and/or different user	as 41
161	EditParent.php	$page, $page2, $delete	True / Reflected	as 41	as 41 but on different page and/or different user	as 41
239	EditClass.php	$page, $page2, $delete	True / Reflected	as 41	as 41 but on different page and/or different user	as 41
37	EditAssignments.php	$page, $page2, $delete, $selectclass	True / Reflected	as 11	- Login as teacher - Go to Assignments, press Add - Select an assignment, press Edit - Intercept request with WebScarab - use payloads on any of the tainted vars	same procedure as 11, except when noted
37.1		$_POST["page"]	True / Reflected	as 16.1		
37.2		$_POST["page2"]	True / Reflected	as 16.2		
37.3		$_POST["delete"]	True / Reflected	as 41.3		as 41.3
37.4		$_POST["selectclass"]	True / Reflected	as 11.3	`//take existing selectclass value e.g. 1, then add` `'><a href="http://unitn.it">XSS on selectclass</a> '`	as 11.3
146, 147, 148	ViewAnnouncements.php	$page, $page2, $onpage	True / Reflected	as 16, but also involves the hidden variable onpage. The "View Announcements" page actually has a form with no submit button. A ficticious button can be added with a malicious value on the onpage variable to prove injection is possible	- Log in as student/parent/teacher - Click on "Announcements" - Intercept request with WebScarab - Add a new field called onpage, set it to 1 plus the injection below - Alternatively, inject payloads to page/page2 - Link is displayed	
146, 147, 148.1		$_POST["page"]	True / Reflected	as 16.1		as 16.1
146, 147, 148.2		$_POST["page2"]	True / Reflected	as 16.2		as 16.2
146, 147, 148.3		$_POST[onpage]	True / Reflected		`//take existing onpage value, then add` `'><a href="http://unitn.it">XSS on onpage</a> '`	Fixed by enclosing all occurrences of onpage in intval, both for forms and for SQL `ViewAnnouncements.php line 67:` `-  <input type='hidden' name='onpage' value='$_POST[onpage]'>` `+  <input type='hidden' name='onpage' value='".intval($_POST['onpage'])."'>`
257	ManageAnnouncements.p	$page, $page2, $onpage	True / Reflected	as 146		as 146
260	ManageTerms.php	$page, $page2, $onpage	True / Reflected	as 146		as 146
268	ManageSemesters.php	$page, $page2, $onpage	True / Reflected	as 146		as 146
273	ManageTeachers.php	$page, $page2, $onpage	True / Reflected	as 146		as 146
283	ManageUsers.php	$page, $page2, $onpage	True / Reflected	as 146		as 146
288	ManageParents.php	$page, $page2, $onpage	True / Reflected	as 146		as 146
293	ManageStudents.php	$page, $page2, $onpage	True / Reflected	as 146		as 146
320	ManageClasses.php	$page, $page2, $onpage	True / Reflected	as 146		as 146
183, 184	ViewAssignments.php	$page, $page2, $selectclass, $onpage	True / Reflected	as 11 and 146 combined. Affects Parent and Student users. Since the page is shared, the fix applies to both		
183, 184.1		$page	True / Reflected	as 16.1	as 16.1	as 16.1
183, 184.2		$page2	True / Reflected	as 16.2	as 16.2	as 16.2
183, 184.3		$_POST[selectclass]	True / Reflected	as 11.3	as 11.3	as 11.3
183, 184.4		$_POST[onpage]	True / Reflected	as 146.3	as 146.3	as 146.3
13	AddAttendance.php	$page, $page2, $student, $semester	True / Reflected	as 11	- Login as schoolmate/test - Go to Attendance, press Add - Intercept request with WebScarab - Use payloads on any of the tainted vars	same procedure as 11
13.1		$page		as 16.1	as 16.1	as 16.1
13.2		$page2		as 16.2	as 16.2	as 16.2
13.3		$_POST[student]				
13.4		$_POST[semester]				

Schoolmate Vulnerabilities

Report ID	File	Variables/Input	Classification	Explanation	Proof-of-concept Attack	Fix
76	EditGrade.php	$page, $page2, $delete, $selectclass, $assignment	True / Reflected	as 37	- Log in as teacher - Go to Grades - Select one, click Edit - Intercept request with WebScarab - Use payloads for any of the tainted vars	
76.1		$page	True / Reflected	as 16.1	as 16.1	as 16.1
76.2		$page2	True / Reflected	as 16.2	as 16.2	as 16.2
76.3		$_POST['delete']	True / Reflected	as 41.3	as 41.3	as 41.3
76.4		$_POST['selectclass']	True / Reflected	as 11.3	as 11.3	as 11.3
76.5		$_POST['assignment']	True / Reflected		// take existing assignment value, then add '><a href=\"http://unitn.it\">XSS on assignment</a><br'	Fixed by enclosing all occurrences of onpage in intval, both for forms and for SQL EditGrade.php line 52: - <input type='hidden' name='assignment' value='$_POST[assignment]' /> + <input type='hidden' name='assignment' value='".intval($_POST ['assignment'])."' />
92	ManageSchoolInfo.php	$page, $page2, $numperiods, $numsemesters, $phone, $address, $schoolname	True / Hybrid	Values in the Manage School interface as administrator are not sanitized.		
92.1		$page	Reflected	as 16.1	as 16.1	as 16.1
92.2		$page2	Reflected	as 16.2	as 16.2	as 16.2
92.3		$numperiods	False Positive	This value is stored without sanitization, but since it is stored as an int it cannot be exploited.		
92.4		$numsemesters	False Positive	as 92.3		
92.5		$phone	Stored	The field is stored as a varchar and output without sanitization, but the length of 15 characters is too short to inject a code snippet, the best that can be done is to insert a small HTML element by balancing quotes and closing the tag before It still is a vulnerability that needs to be fixed.	- Log in as test/schoolmate - Go to School page - In the "phone" field, insert "><i>x</i> - Press update This breaks the page, but the italic x can be seen	sanitize output by applying htmlspecialchars ManageSchoolInfo.php line 12: - $phone = mysql_result($query,0); + $phone = htmlspecialchars(mysql_result($query,0)); sanitize DB input by enclosing the phone number in the update query in header.php with htmlspecialchars
92.6		$address	Stored	This field is in the same situation as $phone, but has 51 characters available, meaning that an exploit is possible.	- Log in as test/schoolmate - Go to School page - Press the Update button - Intercept request with WebScarab - Inject payload on address variable - Resulting page has a link "> <a href="http://unitn.it">XSS</a> <input	sanitize output by applying htmlspecialchars ManageSchoolInfo.php line 7: - $address = mysql_result($query,0); + $address = htmlspecialchars(mysql_result($query,0)); as above, fix by enclosing the address in the update query in header.php with htmlspecialchars
92.7		$schoolname	False Positive	as 53		
194	ParentMain.php	$page, $page2, $selectclass, $student	True / Reflected	as 11, with the addition of the student variable	as 11, but on different page and/or different user and with the addition of the student variable	
194.1		$page		as 16.1	as 16.1	as 16.1
194.2		$page2		as 16.2	as 16.2	as 16.2
194.3		$_POST[selectclass]		as 11.3	as 11.3	as 11.3
194.4		$_POST[student]			//take existing student value e.g. 1, then add -- '><a href="http://unitn.it">XSS on student</a><br'	ParentMain.php line 99: - <input type='hidden' name='student' value='$_POST[student]' /> + <input type='hidden' name='student' value='".intval($_POST ["student"])."' />
142	ParentViewCourses.php	$page, $page2, $student		as 16 and 194	as 16 and 194, but on different page and/or user	
142.1		$page		as 16.1	as 16.1	as 16.1
142.2		$page2		as 16.2	as 16.2	as 16.2
142.3		$_POST[student]		as 194.4		as 194.4
269	AddClass.php	$page, $page2, $fullyear	True	as 16, with the addition of fullyear	- Log in as test/schoolmate - Go to Classes -> Add - Click on "Full Year" - Intercept request with WebScarab - Use payloads on any of the tainted variables	as 16. fullyear can also be sanitized with intval since the year's ID is assumed to be an int by the software
269.1		$page		as 16.1	as 16.1	as 16.1
269.2		$page2		as 16.2	as 16.2	as 16.2
269.3		$_POST[fullyear]	True / Reflected		//take existing fullyear value e.g. 1, then add '><a href="http://unitn.it">XSS on fullyear</a><br'	AddClass.php line 202: - <input type='hidden' name='fullyear' value='$_POST[fullyear]' /> + <input type='hidden' name='fullyear' value='".intval($_POST ["fullyear"])."' />

Schoolmate Vulnerabilities

Report ID	File	Variables/Input	Classification	Explanation	Proof-of-concept Attack	Fix
309	ManageAssignments.php	$page, $page2, $selectclass, $onpage	True / Reflected	as 16 and 146	as 16 and 146, but on different page and/or different user	
309.1		$page		as 16.1	as 16.1	`as 16.1`
309.2		$page2		as 16.2	as 16.2	`as 16.2`
309.3		$_POST[selectclass]		as 11.3	as 11.3	`as 11.3`
309.4		$_POST[onpage]		as 146.3	as 146.3	`as 146.3`
105	Login.php	$page, $message	True / Hybrid	This vulnerability is actually double the vulnerability on $page is reflected, the one on $message is stored. Both these values are not sanitized from user input		
105.1		$page	Reflected		as 16.1	`as 16.1`
105.2		$message	Stored	Data is not sanitized in the school loging MOTD text field and stored in the DB. This vulnerability is particularly dangerous, as it affects even non-logged in users.	- Log in as test/schoolmate - Go to School - Edit the "Message of the Day" with <script>alert("XSS on motd!"); </script> - Alert is displayed at each visit of the login page	`Login.php line 10:` `- $message = mysql_result($query,0);` `+ $message = htmlspecialchars(mysql_result($query,0));` `header.php line 11:` `-  $query [...] sitemessage = '$_POST[sitemessage]', [...]` `+  $query [...] sitemessage = '".htmlspecialcharacters($_POST ["sitemessage"]."',[...]`
54	Login.php	$text	True / Stored	Data is not sanitized in the school login page text field and stored in the DB, then displayed in the login page. This vulnerability is particularly dangerous as it affects even non-logged users just by visiting the login page.	- Log in as test/schoolmate - Go to School - Edit the "text for login page" with <script>alert("XSS on text!"); </script> - Alert is displayed at each visit of the login page	`The fix sanitizes the input in the DB in header.php and the output in Login.php` `The login page text is assumed, in this case, to be simply a free non-HTML text.` `Login.php line 13:` `- $text  = mysql_result($query,0);` `+ $text  = htmlspecialchars(mysql_result($query,0));` `header.php line 11:` `-  $query [...] sitetext = '$_POST[sitetext]', [...]` `+  $query [...] sitetext = '".htmlspecialchars($_POST["sitetext"])."', [...]`
30, 31	ViewAssignments.php	$coursename	True / Stored	Data for the course name is not sanitized in the DB, and is printed as-stored. This vulnerability affects all pages that	- Log in as test/admin - Go to Courses - Add a class, named <a href="/">XSS</a> - Fill the remaining of the form - A link can be seen already in the Manage Classes page, and in all pages mentioning the course's name	`The fix sanitizes the output in the ViewAssignments.php file by applying htmlspecialcharacters.` `Additionally, the input is sanitized by adding htmlspecialchars to the insert and update queries in ManageClasses.php` `ManageClasses.php line 23:` `-  $query = mysql_query("INSERT [...] '$_POST[title]', [...]` `+  $query = mysql_query("INSERT [...] '".htmlspecialchars($_POST ["title"])."', [...]` `[...]` `ManageClasses.php line 73:` `- $query = mysql_query("UPDATE [...] '$_POST[title]', [...]` `+ $query = mysql_query("UPDATE [...] '".htmlspecialchars($_POST ["title"])."', [...]`
207	ManageAssignments.php	$coursename	True / Stored	Data is not sanitized in the course name field. This data is then printed, among others, in the Assignment Management interface. The actual vulnerability is due to the insertion of a malicious course name (see #30/31)	- Log in as test/admin - Go to Courses - Add a class, named <a href="/">XSS</a> - Fill the remaining of the form - A link can be seen already in the Manage Classes page, and in all pages mentioning the course's name - Log in as teacher - Click on the link to the new "xss" course (notice the page is also partly broken) - Click on Assignments - Malicious link is present as title of the course	`The fix sanitizes the output in the Assignment Management interface. Since this is supposed to be free non-HTML text, htmlspecialchars is applied.` `ManageAssignments.php line 7:` `- $coursename = mysql_result($query,0);` `+ $coursename = htmlspecialchars(mysql_result($query,0));` `Fix for the input is provided as part of #30/31`
234	ManageSemesters.php	$term	True / Stored	Data is not sanitized in the term name field. This data is then printed in the Semester Management interface.	The vulnerability is difficult to exploit due to the fact that the database stores the term name as a varchar[15], which very short. As a proof of concept, an italic element is inserted. - Log in as test/schoolmate - Go to Terms -> Add - In the name field, use a brief HTML such as <i>HTML</i> - Fill in with the remaining data - Go to Semesters -> Add - Fill the form, use the malicious term as term name - Submit - Semester with malicious tag is displayed	`The fix sanitizes the output in the Semester Management interface. Since this is supposed to be free non-HTML text, htmlspecialchars is applied.` `ManageSemesters.php line 133:` `- $term = mysql_result($query2,0);` `+ $term = htmlspecialchars(mysql_result($query2,0));` `As an extra step, sanitization is also applied to the insert and update queries in ManageTerms.php as in #30/31`