Multiple Critical Vulnerabilities in Blackboard due to persistent Cross Site Scripting and Authorization bugs

Tung Tran – tunghack@gmail.com
Alireza Saberi - saberi.alireza@gmail.com

The current version of Blackboard (8.0.494.0, also known as 8.0 service pack 7) allows scripts (written in Javascript) to be placed into many user-provided fields and forms. We have been able to show that this ability, coupled with weak authorization errors, can be exploited to mount dangerous attacks, such as a student being able to change their grades as stored on Blackboard. More generally, these attacks can be used by a student to perform actions that would otherwise require the credentials of a TA or an instructor. Note that one of these attacks does not require any social engineering, and there isn't anything an instructor can do to protect themselves from the attack.

1. Vulnerabilities

1.1. XSS vulnerabilities

In Blackboard, a student can insert scripts into these user-provided fields and forms:

a. Personal website: in each course, a student has a personal website where he is allowed to use scripts.
b. Course list modification: a course name and URL fields can contain scripts.
c. My calendar: a calendar event name and content can contain scripts.
d. Textpad: a feature supported by Blackboard which can be added to a student's main page. It is possible to put scripts in the content of Textpad.
e. My tasks: another feature supported by Blackboard where a task name and content can contain scripts.

From this list, only a student's personal website is accessible by other users. It seems that only this feature can be exploited to carry out an XSS attack. Unfortunately, Blackboard is also susceptible to multiple authorization vulnerabilities, which make other features from the list useful in constructing effective attacks.

1.2. Authorization vulnerabilities

Without proper authorization, a student can be able to do the followings which he should not be allowed to:

1.2.1. Change, add and view calendar events of all people in course

In this attack a simple user (student) can add/change calender events of all the people in a course and inject some scripts to victim's event.

A student can view an event in his calendar by using this HTTP GET request:

GET bin/common/calendar.pl?subroutine=event_view&go_to=my_inst&location=mybb &_event_id=_46152_1

In this GET request, *_event_id* is a unique event id assigned to each event. By changing this field, a student can view events of others. The event id assigned to each new event increases sequentially, so it is easy to guess the next valid event.

Parameter Name	**Value**
course_id	**_296800_1**
location	**st**
filter	showAll
view	month
datetime	2010-10-15+17%3A55%3A00
_event_id	_pk1_pk2
subroutine	confirmation
update	add
subject	Event1<script>alert ("You can inject scripts here")</script>
message_f	
message_w	
messagetype	H
text_style	html
text_format_type	H
message	EvenBody<script>alert ("You can also inject scripts here!")</script>
calendar_mm	10
calendar_dd	15
calendar_yyyy	2010
start_date	2010-10-15+17%3A10%3A00
pickdate	
pickname	
end_date	2010-10-15+17%3A55%3A00
calendar_hh	05
calendar_mi	10
calendar_am	1
calendar_e_hh	05
calendar_e_mi	10
calendar_e_am	1
x	42
y	14

Table 1. Parameters of adding calendar event post request

There are different kinds of events in a student's calendar. They are personal, course and institution events. All of these events appear in the student's calendar.

It is possible that a student can change any of these events (modify, delete). For personal (belong to another) and institution events, he can delete and modify them, however, these events no longer appear in the victim's calendar. On the other hand, he can modify/add/delete a course event and this event will appear on calendars of all students who are taking the course. To add an event to a given course, he submits the following HTTP POST request:

POST /bin/common/calendar.pl?course_id= HTTP/1.1

There are several parameters used in this POST request as shown in Table 1. *course_id* is a valid course id (can be easily found from the course list category). Parameter *location* plays an important role in the authorization process.

1.2.2. Adding a task to Task list of all people in a course

Parameter Name	**Value**
action	u
display	priority
filter	--%21%21all
task_id	
isCourse	0
render_type	**EDITAB\\LE**
course_id	_296800_1
subject	Task1<script>alert ("Pwned")</script>
description_f	
description_w	
descriptiontype	H
text_style	html
text_format_type	H
description	TaskBODY<script>alert ("Pwned")</script>
duedate_mm	10
duedate_dd	1
duedate_yyyy	2010
pickdate	
pickname	
priority	N
status	N
submit.x	45
submit.y	15

Table 2. Parameters of adding task post request

In order add a task to a given course, a student submits the following HTTP POST request:

POST /bin/common/tasks.pl HTTP/1.1

There are several parameters used in this POST request as shown in Table 2. Parameter *render_type* plays an important role in the authorization process. By adding characters like "\\", we can bypass the security check.

1.2.3. Change/modify personal course homepages of other students

In order to do this, a student starts modifying his own course webpage; however, before he submits the changes, he modifies some fields from the POST request to make the changes apply to another student's webpage. For example, in order to modify his security course CSE509 webpage, he submits parameters to the following HTTP POST request:

POST /bin/common/homepage.pl?course_id=_296800_1&action=EDIT HTTP/1.1

Specifically, he needs to change the following fields so that his modifications can actually affect another student's website:

crsusrs_pk and a list of *props__course_users___603762_1__XXX* fields where 603762 is a student (victim) id.

In a given course, each student has a unique *crsusrs_pk* for his website. Moreover, Blackboard also assigns each student a unique id. If we change these values to those that belong to another student in the HTTP POST request, that student's website content will be changed. These values for each student can be easily found from the course related contents provided by Blackboard.

1.2.4. Change "My tasks" of other students

As for calendar event, in order to change someone else's tasks, a student starts modifying one of his tasks and submits his changes using this HTTP POST request:

POST /bin/common/tasks.pl HTTP/1.1

By providing a correct *task_id* parameter to this POST request, the task having *task_id* will be changed. Again, *task_id* is assigned sequentially to each new task.

2. Attacks

2.1. XSS and social engineering

A student can put scripts in his course homepage and ask people to visit it to steal their cookies. In addition, to make it more persuasive, a student first modifies a victim's course homepage and then tells the victim to check out the homepage.

2.2. Install back-door in a victim's Blackboard account

Now, we assume that a student (attacker) already got another student's cookies and had access to

the victim's session. Then the attacker can modify/add scripts to one of the items listed above (those that allow scripts). This allows the attacker to regain access whenever the victim logs into his account.

2.3. Inject scripts directly to a victim's Blackboard main page

As mentioned above, a student can modify/add an event to a course calendar (or task) that he wants. All users taking the course, TAs and instructors will have the event in their calendars. Moreover, the calendar and task feature are enabled by default. It means that whenever a victim logs in, the calendar and task will appear on his main page and the injected scripts will be executed. A student may try to hijack the instructor session or put ajax requests inside injected script to change grades as soon as the instructor logs in. This attack requires no social engineering.

3. Workarounds

We need to make sure that scripts are only allowed in user-provided forms and fields which require Javascript to operate properly. On the other hand, it is safer to disallow Javascript in all other supported features. Note that Blackboard needs Javascript enabled by the browser in order to work, so we cannot rely on temporarily disabling Javascript in the browser to avoid the attacks.

A temporary solution is to disable the personal homepage, calendar and task services in Blackboard.