SECURITY VULNERABILITIES IN MODERN LMS

Alexei Scerbakov, Frank Kappe and Nikolai Scerbakov
Graz University of Technology, Graz (Austria)

ABSTRACT

An effective learning management system (LMS) is a vital component of an E-Learning infrastructure in universities nowadays. Learning Management System can be seen as a structured repository of courseware materials provided with additional communication facilities such as discussion forum, annotations, chats, etc. LMS can be also seen as a communication area where students upload their assignments and get feedback from the teachers. From the technical perspective, LMS is an Internet-Based Information system serving a big number of remote users. In this paper, we present a list of security issues that we experienced during the years of practical LMS usage on the university level. We especially investigate security issues that are specific for the modern LMS that are constructed in accordance with the so-called AJAX architecture. The practical value of this paper is defined by possible reproduction of the security preconditions that are also described in the paper.

KEYWORDS

E-Learning, Learning Management System, Software Architecture, Cloud Service, Security

1. INTRODUCTION

Learning Management System is a vital component of the educational infrastructure in schools and universities. From the technical perspective, LMS is an Internet-Based Information system serving a big number of remote users (Cochran, 2015; Ramani, 2017; Uzunboylu, Bicen & Cavus, 2011). Common LMS functionality includes libraries of educational materials, course announcements, file repositories, curriculum descriptions, file exchange, discussion forums, chats, email exchange, online quizzes, opinion polls, document annotation, evaluation facilities (Ramani, 2017; Uzunboylu et al., 2011; Dobre, 2015), etc. Modern LMS are often built using comparatively new architectural solutions known as Asynchronous Java and XML (AJAX) (Khanna & Mistry, 2012). Two aspects make LMS a frequent target for security attacks. LMS is a huge repository of sensitive data and the system functionality is highly important for the university as such. Hence, the LMS must be provided with a relevant level of security. Graz University of Technology has been using LMS for 20 years. During such a long period of actual application of the system for serving dozens of thousand students simultaneously, a number of basic security threats were identified and the attacks were successfully prevented.

In this paper, we mainly concentrate on security vulnerabilities that are related to the specific functionality of LMS and to special properties of AJAX architecture. It is demonstrated that LMS vendors need to take these kinds of threats seriously.

2. COMMON LMS SECURITY THREATS

LMS must support a certain functionality by definition (A. Scerbakov, Ebner & N. Scerbakov, 2015; N. Scerbakov, 2018; Kappe & N. Scerbakov, 2017). Thus, there must be facilities for

- uploading students' training assignments and other materials to the server;
- commenting on the materials on the server;
- participation in the topic or course-oriented discussion forums;
- checking the students' knowledge online, etc.

All such LMS specific actions create reasonable security threats that are discussed below.

2.1 Sharing User Accounts

Assessing learner progress is an important phase of e-learning programs (N. Scerbakov, Kappe & Pak, 2018). To grade the students' progress with the educational materials, teachers need to carry out certain evaluations and assessments. Potentially electronic exams is a highly effective tool for testing learner knowledge.

An important component of the electronic examination is correct identifying the student that performs the e-learning tasks. Simply stated, the credentials issued for the particular student must be used

- by this particular student;
- exclusively by this student;
- by the student without assistance from other students.

We address a possible violation of the rules above as "Sharing User Accounts". There are potentially three methods for assuring fair play during the online examinations:

The first method is rather obvious, the online examination is carried out from an especially allocated room equipped with a sufficient number of computers. In this case, the examination is monitored by the teacher or an assistant who is responsible for checking the user IDs, and their behavior during the examination.

Another method is based on the usage of video conferencing software. In this case, all the students must login into the video session first. The teacher may check the students' IDs and behavior during the examination by means of the video conferencing tool.

Finally, the LMS may be equipped with the special software component that can:

- Randomly make snap shots of the user in front of computers;
- Recognize the faces to prove that the students with the particular credentials, are indeed seating in front of computers.
- Recognize the scene in front of the computer to prove that the student works alone and is not assisted by someone else seating along.

2.2 Deny of Services via Uploading

Any LMS provides facilities for gathering (uploading) students' assignments, commenting and evaluating the assignments by teachers (N. Scerbakov & Kappe, 2018). Normally, this functionality is provided by the combination of two components: HTML form on the client that allows selecting the local file for uploading and a server-side component that get the HTTP request from the form and physically creates the new file on the server.

There are two aspects that make this situation potentially dangerous:

- HTML 4.0 does not allow to check size and extension of the file selected by the user before physical uploading this file on the server.
- The HTML form contains all the information on the server application (URL + parameters) as plain text.

As a result, users may potentially select files of a rather big size and send them for uploading. For example, we monitored attempts to upload whole distributions of operating systems, huge movie files, etc.

More dangerous would be a usage of Internet bots that send requests to the uploading URL automatically with the correct set of parameters. Such bots can easily initiate uploading of hundreds of files in a second. This may cause "deny of service" for other users.

LMS can be easily protected from this kind of malicious activity. Recent extension (HTML 5.0) provide a special file object that allows reading size and extension of local files before uploading. Thus, the LMS may be easily prevented from uploading oversized files of the wrong type. Uploading bots can be prevented from the functionality by using special tokens giving permission for uploading as parameters of the HTML forms (see below).

2.3 Injection of Malicious Code

The malicious code can be injected for two reasons (A. Scerbakov, Kappe & N. Scerbakov, 2018). The first and most common situation is an attempt to post the executable code or fragment of such code to the server where it can be executed to retrieve some sensitive data or disrupt the server functionality.

Obviously, the files containing the executable code can be uploaded as student assignments (see above). Such HTML 5.0 extension as the new "file" objects allows verification of the files on the client site before uploading. Unfortunately, such verification of files on the client is not fully sufficient. As it was mentioned above, the file uploading transactions to the server can be easily mocked up. Hence, special preconditions like temporary tokens are needed. Moreover, all the files uploaded by the students must be verified on the server immediately after uploading.

Another type of thread is based on code snippets embedded into textual messages to be interpreted on the user clients. The simplest way for this kind of attacks is embedding references to external JavaScript files or the JavaScript snippets directly into texts of messages or comments. The messages and comments are visualized on the client as components of the dynamically generated HTML files, and, thus, will be automatically interpreted on the clients.

Many people think that injection JavaScript fragments is not dangerous and can be seen as a kind of kidding. In reality, such fragments can be used to carry out serious attacks on the clients and the server.

Thus, for example, JavaScript code may access the cookies on the client computer, the cookies normally contain the session ID. The "xmlhttp" JavaScript object may potentially send the session ID to another server. If the session ID is used as the special token for the verification of the HTTP requests, the server security can be essentially compromised. Client security can be also very seriously compromised. For example, if the generated HTML document contains references to other files as for example for downloading the files from the trusted server, the references can be easily scanned and replaced with references to other malicious executable files. The files then will be downloaded by users as training materials and potentially executed locally. Hence, the LMS must automatically scan all the incoming messages and remove any executable fragments and/or JavaScript references to other servers.

3. AJAX SECURITY THREATS

Many LMS are implemented nowadays as so-called AJAX applications. AJAX is a technique for accessing Web services from the client by means of special XMLHTTP JavaScript objects. AJAX-based LMS can be seen as a combination of two main components: server-side WEB services (Back-End Application) and an especially developed client (Front-End application). Front-End is developed using concepts of HTML objects and JavaScript programming. This combination of HTML and JavaScript is called Dynamic HTML (DHTML).

DHTML documents can be downloaded from the server and visualized on the browser screen. The documents are dynamic in the sense that the documents change their appearance directly on the client without re-loading the documents from the server. Such DHTML documents can access the WEB services and request data from the server. The client sends the HTTP request to the server by means of the special JavaScript object. The WEB service processes the request on the server site and sends the response back to the client. The client gets the response and processes data on the client side. Normally, Front-End and Back-End components use Extensible Markup Language (XML) and/or JavaScript Object Notation (JSON) as a communication protocol.

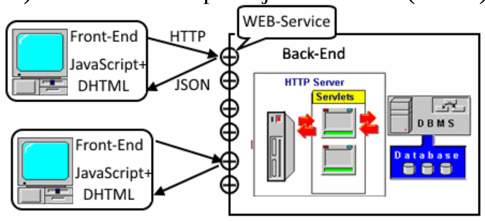

Figure 1. AJAX Architecture

AJAX architecture has a number of well-known advantages:

- Such systems demonstrate very good performance and comfortable response time because the user actions do not require re-loading HTML pages from the server;
- The workload on the server site is greatly decreased since the server just implements a number of WEB services for accessing/modifying database, and all the data processing is carried out on the client side.

Thus, such applications facilitate user satisfaction with the system functionality and performance especially in the situations where hundreds or even thousands of students work simultaneously in the condition of time restrictions, for example, online examination, approaching deadlines, etc. Moreover, the actual configuration of the production server for the LMS back-end does not require powerful hardware and/or special software solutions like server clusters, load balancers, reverse proxy, etc.

Another important aspect of this architectural solution is a possibility to use multiple front-ends for communicating with one and the same Back-End (Cloud). In other words, there may be different user-clients that can be used in different circumstances (desktop client, tablet, smartphone, etc.) working with the same WEB Services. Even mobile applications that are distributed via vendor-oriented software repositories (application shops) can be developed as such clients for the AJAX applications.

Moreover, the architecture provides new possibilities for implementing User Interface in LMS. Traditionally, User Interface is implemented as a combination of „Jump" actions when users click on active HTML fragments - so-called "Anchors", to download and visualize another document. AJAX architecture allows using such modern User Interface solutions as - document event model, dynamic prompts, floating windows and virtual pop-ups. The AJAX architecture allows embedding so-called local sensors into the communication between users and the system as well. Thus, video camera, information on the current location, sound, etc. can be used as the interface elements.

AJAX architecture itself creates a certain security threat. Recollect that we see the AJAX based system as two components – Front-End and Back-End that communicate by means of the HTTP protocol. The Front-End generates HTTP requests by means of JavaScript objects. The Back-End process the requests, potentially retrieve and/or modify data and generate the response to the server. The security of such AJAX LMS can be compromised by means of two types of attacks discussed below.

3.1 Mocking the Client-Server Communication

The functionality of the AJAX LMS is distributed over two different components – Front-End and Back-End. Back-End is normally responsible for retrieving the data from the database and modifying the data in the database. Front-End implements GUI and certain data processing logic. Thus, the Back-End component expects HTTP requests that come from the relevant Front-End. The requests invoke WEB Services that retrieve and/or modify the data in accordance with the logic of the data processing implemented by the client. The WEB services potentially can be invoked by any Internet client that can mock the request format (see Figure 2).

In this case, the WEB services are invoked without the proper Front-End context and potentially can illegally retrieve the data, or even do the illegal modifications of the database. This problem can be efficiently solved by dynamically generated tokens that identify authorized clients. The tokens can be applied using the following schema: as a particular client is authorized, the system generates a token that is kept on the server as a session variable and on a client as a JavaScript variable. Any further requests are additionally authorized with this token that is checked on the server before fulfilling the request.

This method is not secure enough since the token value is kept on the client as a JavaScript variable, and the token value is valid for a number of requests. Thus, after performing the first request, the value may be read and used to generate a further set of requests.

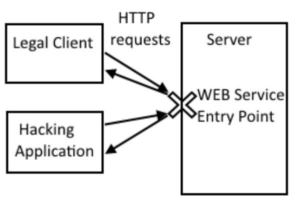

Figure 2. Mocking the client-server communication

In cases that require a higher level of security, tokens are generated for each request.

3.2 Changing JavaScript on the Client Site

The second type of security leak is explained by the main AJAX principle where software (JavaScript functions) are fetched from the server and evaluated on the client side. Using a simple tool like FireBug, anyone can change JavaScript code and parameters on the client side. If anyone takes time and studies LMS for a while, they can learn how to change JavaScript code resulting in hacking LMS.

Actually, the problem of manipulation with client-side scripts does not have an ultimate solution in AJAX. The only possible remedy is double checking the most sensitive transactions on the server site. For example, if a WEB service that returns info on a particular student by ID is invoked, the WEB service may check whether the client has really got rights to do so by checking user-name and privileges on the server session. Generally speaking, when any JavaScript function calls the WEB service, the server side needs to check if this client is authorized to do this action. Building additional server-side permissions mechanism to prevent unwanted actions takes a lot of time and almost double the LMS development expenses. To implement such a mechanism, for every action the LMS must validate it on the server, and can only do this by fetching the needed data from DB.

Of course, some security preconditions can be done on the client site as well. Obvious things could be decreasing readability of the JavaScript. JavaScript files can be compressed and all the function/variable names replaced with plain numeric notation. The JavaScript fragments can look, for example, as:

```
var v000234=false;var v000237='';var v000238='';var v000244='/wbtmaster/room_icn/';function
v000315(v000656){var v000234=v000656;var v000237=v000656.indexOf(';'+v000713+':');
if(v000612==-1) …
```

The other method is based on checking the flow of control between the JavaScript functions. The function may identify a so-called “caller”, i.e. another function that calls this one. The list of all legal callers can be predefined, and if the function is called from a not-legal caller, the function may send an alert to the server.

REFERENCES

1. Cochran, K. (2015). Report - LMS Trends 2015: Is it Time for Something Different? Retrieved from https://www.docebo.com/blog/report-lms-trends-2015-brandon-hall/
2. Dobre, I. (2015). Learning Management Systems for Higher Education - An Overview of Available Options for Higher Education Organizations. Procedia - Social and Behavioral Sciences, 180, 313-320. doi:10.1016/j.sbspro.2015.02.122
3. Kappe, F., & Scerbakov, N. (2017). File Uploading Scenarios in a Modern Learning Management System. In L.G. Chova, A.L. Martínez & I.C. Torres (Eds.), Proceedings of EDULEARN17, 9th International Conference on Education and New Learning Technologies (pp. 4904-4909), Barcelona, Spain: IATED Academy. doi:10.21125/edulearn.2017.2100
4. Khanna, S.V.O., & Mistry M. (2012). Impact of Ajax in Web Applications. International Journal of Advanced Engineering Technology, 3(1), 144-145. e-ISSN: 0976-3945
5. Ramani, R. (2017). 2017 Preview: 11 Learning Management System Technology Trends by the Industry's Top Analysts. Retrieved from https://elearningindustry.com/11-learning-management-system-technology-trends-top-analysts-2017-preview
6. Scerbakov, A., Ebner, M., Scerbakov, N. (2015). Using Cloud Services in a Modern Learning Management System. Journal of Computing and Information Technology, 23(1), 75-86. doi:10.2498/cit.1002517
7. Scerbakov, A, Kappe, F. & Scerbakov N. (2018). Block-Chain Based Grading Students Assignments. In L.G. Chova, A.L. Martínez & I.C. Torres (Eds.), Proceedings of ICERI2018, 11th International Conference on Educational Data Mining (pp. 8773-8778), Seville, Spain: IATED Academy. doi:10.21125/iceri.2018.0615
8. Scerbakov, N. (2018). TU Graz WBT-Master. Retrieved from http://coronet.iicm.tugraz.at/wbtmaster/welcome.html
9. Scerbakov, N. & Kappe, F. (2018). WBT-Master - Modern Learning Management System. In R. Damaševičius, G. Vasiljevienė (Eds.), Proceedings of the 24th International Conference on Information and Software Technologies, ICIST 2018, Vilnius Lithuania, Communications in Computer and Information Science, 920, 461-475. doi:0.1007/978-3-319-99972-2_38
10. Scerbakov, N., Kappe, F. & Pak, V. (2018). Collaborative Document Authoring as an E-Learning Component. In T. Bastiaens, J. Van Braak, M. Brown, L. Cantoni, M. Castro, R. Christensen, G. Davidson-Shivers, K. DePryck, M. Ebner, M. Fominykh, C. Fulford, S. Hatzipanagos, G. Knezek, K. Kreijns, G. Marks, E. Sointu, E. Korsgaard Sorensen, J. Viteli, J. Voogt, P. Weber, E. Weippl & O. Zawacki-Richter (Eds.), Proceedings of EdMedia, World Conference on Educational Media and Technology (pp. 101-107), Amsterdam, Netherlands: Association for the Advancement of Computing in Education (AACE).
11. Uzunboylu, H., Bicen, H., & Cavus, N. (2011). The Efficient Virtual Learning Environment: A Case Study of Web 2.0 Tools and Windows Live Spaces. Computers & Education, 56(3), 720-726. doi:10.1016/j.compedu.2010.10.014