Determination of optimal security settings for LMS Moodle

Zlatko Stapić, Tihomir Orehovački, Mario Đanić
Faculty of Organization and Informatics
University of Zagreb
Address: Pavlinska 2, 42 000 Varaždin, Croatia
Phone: +385 42 390 800 Fax: +385 42 213 413
E-mail: {zlatko.stapic | tihomir.orehovacki | mario.djanic}@foi.hr

Abstract - E-learning usage at high-education institutions represents prerequisite of new, modern and quality education. Implementation of suitable learning management system (LMS) is only a first step in realizing this need. Through the last few years LMS Moodle imposed itself as the best solution, and is becoming one of the most common used systems in Croatia. Although it is developed by the „open source" model that allows quicker and more effective reaction to security bugs inside the LMS itself, vulnerability of the mentioned system greatly depends on the security measures configured on the server. In this paper, we will present a summary of most common security flaws and suggest optimal settings of Moodle LMS and the server itself. Our claims will be supplemented by the results of stress tests and security analysis in order to determine the optimal settings.

I. INTRODUCTION

Trying to explain new and popular way of learning which is computer-enhanced we often use term *e-learning*. One of many definitions used to describe e-learning associates process of education which includes information and communication technologies (ICT) along with process of continuous quality improvement in education and in its results as well. Depending on the ICT usage intensity, we can define few different e-learning forms. Basically, ICT can be used along with traditional face-to-face education, but its use can also create hybrid (mixed) mode or even full online mode [11].

In this paper last two e-learning modes will be taken into consideration because their implementation includes creation of *Virtual Learning Environments (VLE)* in which all aspects of a course can be handled through a consistent user interface. Such e-learning systems which provide VLE are sometimes called *Learning Management System (LMS)*, Course Management System (CMS), Learning Content Management System (LCMS), Managed Learning Environment (MLE), Learning Support System (LSS) or Learning Platform (LP) [17].

The first learning management system or better computer assisted instruction system was introduced in 1960 by University of Illinois and was called Plato (later described as Programmed Logic for Automated Teaching Operations). This system pioneered LMS key concepts such as online forums and message boards, online testing, email, chat rooms, picture languages, instant messaging, remote screen sharing, and multiplayer online games. After this pioneer, which was turned off in 2006 [23], hundreds of similar systems were introduced. Major milestone happened in 1997 when WebCT 1.0 was released and Blackboard was founded because these two LMSs attracted millions of users [3][5][17]. Nowadays, there are more than 150 different systems providing e-learning services, but after WebCT and Blackboard second milestone was *LMS Moodle*, which was introduced in 1998 and finally released in 2001 [3][6]. Moodle, as *Modular Object-Oriented Dynamic Learning Environment*, soon imposed itself as best solution and is becoming one of the most common used learning management systems [6].

Data obtained from official Moodle statistics sites confirms the mentioned fact. In February 2008, there was more than 38 000 registered sites with more than 16.61 millions of users [7]. After taking into consideration that not all Moodle sites or users are registered, these numbers could be several times bigger.

As every LMS, Moodle has an ability of tracking the learners' progress, which can be monitored by both teachers and learners. This fact implicitly includes both security and privacy threats and makes Moodle vulnerable system. Having all mentioned Moodle sites online, it becomes crucial to recommend necessary security and privacy protection mechanisms which should be implemented in order to minimize security and privacy vulnerabilities.

Subsequently, this paper will be divided into three main parts (not including introduction and conclusion). After introducing the problem, we will focus on security and privacy vulnerabilities. These vulnerabilities and threats will be discussed from the LMS point of view, and additionally these threats will be grouped into four main groups according to their LMS related type. After having all threats and vulnerabilities introduced, focus will be transmitted on Moodle modules architecture in order to emphasise its possible weak and for security and privacy interesting points. Finally, in the last chapter, several possible and different security settings will be presented, along with test results on each. Aim of these analyses is discussion on final recommendations of optimal security settings which should be implemented on server and client side in order to maintain maximal security and privacy. These final recommendations will not take into consideration security and privacy about mentioned Moodle modules and their development, but will be related only on server and client desirable settings.

II. LMS SECURITY VULNERABILITIES

Learning management systems are client/server web applications that, among rest, manage user requests coming from clients such as web browsers [24]. To handle the user requests, they often require accessing security-critical resources (e.g. databases and files) at the server end. In this

section we present description of the most critical security flaws [8][9] that are classified into four categories: authentication, availability, confidentiality and integrity attacks. Table 1 displays a summary of classified attack methods and vulnerabilities independent of the specific LMS implementation. Model used to group attack methods and security vulnerabilities is widely accepted AICA (Availability, Integrity, Confidentiality and Authentication) threat modeling approach.

TABLE I
ATTACK METHODS AND SECURITY VULNERABILITIES

Authentication attacks	
1.	Broken authentication and session management
2.	Insecure communication
Availability attacks	
1.	Denial of service
Confidentiality attacks	
1.	Insecure cryptographic storage
2.	Insecure direct object reference
3.	Information leakage and improper error handling
Integrity attacks	
1.	Buffer overflow
2.	Cross Site Request Forgery
3.	Cross Site Scripting
4.	Injection flaws
5.	Failure to restrict URL access
6.	Malicious file execution

A. Authentication attacks

Authentication attack occurs when an attacker steals password and thus identity of legitimate end-user with an aim of free access to paid e-learning services. When a LMS authentication has been broken, an attacker has an opportunity to perform availability, confidentiality or integrity type of attack. Today's most critical authentication vulnerabilities are:

1) Broken authentication and session management: vulnerability which occurs because account credential management functions (e.g. remember my password, forgot my password, change my password, etc.) and session tokens are not often properly protected. An attacker can compromise passwords or authentication token to assume other user identity. Furthermore, attacker can intercept and steal authenticated session of a legitimate user.

2) Insecure communications: vulnerability which appears during transmits of sensitive information (e.g. session tokens) without proper encryption. Attacker can misuse this flaw to impersonate user and access unprotected conversations.

B. Availability attacks

The main goal of availability attacks is to make e-learning services and data unavailable to authorized end-users. Most popular form of availability attack is denial of service (DoS) attack which aims to misuse finite bandwidth and connectivity resources of LMS system. DoS attacks are usually malicious but they can also be result of users' incautious behaviour. There are two general types of DoS attack: logic and flooding attacks. Logic attacks (e.g. ping) exploit existing LMS flaws to crash remote server or significantly decrease its performance [19]. Flooding attacks overloads LMS with a high number of requests to disable legitimate users from accessing e-learning resources. DoS attacks present threat to LMS systems because one request can be replicated to many participants.

C. Confidentiality attacks

Confidentiality attacks are passive kind of attacks which allows unauthorized access to confidential resources and data. The main intention of attacker is not data modification but data access and dissemination. The most frequently confidentiality flaws are:

1) Insecure cryptographic storage: flaw which is based on a fact that sensitive information does not have appropriate encryption. LMS systems rarely use cryptographic functions properly to protect data and credentials or use weak encryption algorithms. In both situations, valuable data is relatively easy to access by attacker who can conduct identity theft and similar crimes.

2) Insecure direct object reference: this vulnerability usually occurs when LMS uses object references directly in web interfaces without authorization checks being implemented. Mentioned object references can be files, database records and primary keys and are contained either by URL or form parameters. An attacker can misuse direct object references in order to access other objects without authorization.

3) Information leakage and improper error handling: refers to unintentional disclosure of sensitive data and unneeded information through error messages. LMS can leak sensitive information about its logic, configuration and other internal details (e.g. SQL syntax, source code, etc.). On the other hand, error messages that LMS generate may display too much information which can be useful to attackers in privacy violation or conducting even more serious attacks.

D. Integrity attacks

This group includes attacks which attempt to create new data or modify and even delete existing e-learning data. Most popular of them are:

1) Buffer overflow attack: occurs when a LMS component (e.g. libraries, drivers, server components) tries to store data into an available buffer without validating its size. By inserting larger values than expected (e.g. 800 characters in a limited length field), attackers can cause their malicious code to be executed. There are two ways how attacker can take control over application [15]: by injecting attack code or by using code which is already in LMS address space.

2) Cross Site Request Forgery (XSRF/CSRF): client side attack which exploits trust that a LMS has for the user [18]. When a user is logged into LMS, attacker can trick his browser into making a request to one of LMS task URLs which will cause a change on the server. While request comes with the user's cookies, server will perform it as it is original. Attacker could use this vulnerability to do anything what authenticated user can do.

3) Cross Site Scripting (XSS): refers to hacking technique which allows an attacker to supply vulnerable dynamic web page with malicious script and execute script in victim's browser in order to gather data from a user. There are three general types of XSS: persistent, non-persistent and DOM-based. In our case, the most important meaning have persistent (stored) attacks [22], in which malicious data are persistently stored on the target back end system (e.g. in database) and displayed to the user in a unfiltered form. This is extremely dangerous in LMS because users could see inputs of all other participants.

4) Injection flaws: may occur when data provided by user (e.g. in form fields) is sent to content checking routines as part of a command or query [20]. In such attacks, interpreter fail to detect or respond to character sequences that may be interpreted incorrectly, which then results in execution of malicious code by LMS. Finally, attacker could be able to create, update, read or delete all data available to LMS.

5) Malicious file execution: attack which is based on a fact that LMS fails to control or prohibit execution of uploaded files. Malicious code is usually uploaded via upload feature (e.g. homework or image). This kind of vulnerability can be found in many web applications, especially in those which are PHP based.

6) Failure to restrict URL access: some LMS resources are restricted to a small subset of privileged users (e.g. administrators). This weakness allows an attacker to retrieve URLs by guessing the address and perform unauthorized operations on unprotected LMS data.

III. MOODLE ARCHITECTURE

In previous chapter all LMS-relevant security threats and vulnerabilities were enumerated and grouped according to their type. As stated before, great majority of these vulnerabilities depends on system architecture as much as on system implementation and server settings. This chapter will bring into focus Moodle architecture in order to indicate weak and discussion worth points which could be possible threats and vulnerabilities.

Covering many collaborative and learning fields, Moodle is composed from independent modules; plug-ins. In order to ensure better understanding of a whole Moodle architecture, these modules will be presented in groups according to their purpose or use. From this perspective, there are six groups of modules as follows [2]:

1) *Communication modules*
2) *Productivity modules*
3) *Student involvement modules*
4) *Administration modules*
5) *Course delivery modules*
6) *Curriculum design modules*

1) Communication modules and tools: are backbones of all intra and extra communication features. These modules include *discussion forums, file exchange, internal and external email* and *real time chat.* Among other possibilities, while using discussion forums, users can include in their post different attachments, images and direct URLs. This feature, as well as file exchange feature which allows assignment submission, should be taken into consideration and observed as possible week point for a few threats. Due to possible *insecure communication* intruder could come into possession of any data that is sent in any private communication channel. Furthermore, *insecure direct object reference* could allow intruder to come into possession of any document he is not authorized for. Finally, almost all previously stated *integrity attacks* should also be taken into consideration.

2) Productivity modules: include *help module, search module, calendar module, progress* and *review modules.* Although these modules seem not to be threats, one issue must to be annotated. *Information leakage* must be strictly prohibited, because otherwise anybody could see important data, or search results he is not authorized for. For example student could see (accidentally or with purpose) grades of his colleagues. As well as information leakage, *insecure direct object reference* could also cause problems.

3) Student involvement modules: include *groupwork module* and *workshop module,* along with *self-assessment* and *student portfolio module*. After performing any previously mentioned illegal action intruder could either come into possession of others' data or change student or group-relevant data on server. Additionally, any system-side (also previously mentioned) threat should also be carefully taken care of.

4) Administration modules: should probably be most carefully considered and paid attention to, because gaining access into these modules results in having access in all other modules. The well known *authentication, course and user authorization, registration integration* and any other *hosted services module* goes into this group. The authentication modules allow Moodle to use LDAP, IMAP, POP3, NNTP and other databases as sources for user information. Discovering and fixing all security-related bugs in these modules becomes crucial in any LMS development. Intruders mostly attack modules in this group, often using any known method and vulnerability. All encountered threats should be taken into consideration in implementation of authentication and other related modules.

5) Course delivery modules: are probably second most vulnerable group of modules and are usually only

authorized by administrators and teacher for use. Representative modules in this group are *course management module, helpdesk module, online grading tools, students tracking module* and finally *automated and testing modules*. Beside omni-present *authentication attack* threats, discussing course delivery modules, we will focus on *integrity attacks* while these have the purpose of unauthorized data change. Course management module and online grading module should be considered to be security safe on possible integrity attacks in particular.

6) Curriculum design modules: finally form last group of modules, used in curriculum creation. *Course templates* and *customization modules* are main representatives. As last group of modules presented, they also have least negative impact as result of possible attacks. Data changes reflect on curriculum design are easily recognized and attackers usually do not have any particular interest in compromising these modules, while they work is usually accepted as more or less harmful or malign joke.

All mentioned modules form second layer in multi-tier Moodle architecture. Security and privacy threats typical at data-base layer or client-side layer will not be presented in this chapter. These security and privacy threats are well known and should be considered in any LMS implementation and development. Although Moodle has XMLDB as its database abstraction layer, which lets Moodle to interact with and access the database [6], usual and previously stated precautions actions.

Finally, encountered groups of modules, as can be seen in above security discussion, do not have same level of importance from the security and privacy point of view. Also, previously mentioned groups of attacks do not have same level of possible destruction if associated attack happens. Subsequently, the worst case could be authentication attack performed on any administrative module. From the privacy point of view, authentication attacks persist, but worst case scenario includes also confidentiality attacks performed on student personal data and private achievements.

IV. RECOMENDED SETTINGS

Following focus on Moodle's security and performance, a set of concise advices and recommendations will be presented in this part of paper to assist in building a stable environment for everyone to use. The software platform is based on Ubuntu Linux 8.04 and LAMP (Linux, Apache, MySQL and PHP), supporting Moodle installation.

Steps needed to achieve optimal configuration settings include detailed analysis of needs, selection of suitable hardware platform, performance tests execution, and following inspection of its results, implementation of necessary optimizations.

A. Hardware upgrades

Thinking raw strength, upgrading hardware is probably the easiest way to improve Moodle's installation performance. By using more RAM swap usage will be brought to minimum, bringing better performance and reduction of disk-activity as final result.

If system starts swapping, it is a sign that it needs more RAM. Suggestion is to first think of RAM upgrades, since it will almost certainly be the biggest bottleneck in the overall environment.

After that performance will be upgraded using better and faster hard disks, hopefully 4 SCSI disks forming RAID10 array, for data redundancy and performance. Although a processor shouldn't be such a problem, in cases with lots of users and SSL, it is advisable to upgrade it as well. Finally, utilization of multiple web servers with load balancing techniques will also improve performance [16].

B. Performance optimizations

Results of various optimizations and hardware upgrades can be shown by doing various stress tests, and this article will present results collected by running *Apache ab*, tool built for benchmarking Apache Hypertext Transfer Protocol (HTTP) server [1]. Some tests that could also be done manually, but are outside of this article's scope, include *FireBug* and *Yslow* Firefox extensions, which in conjunction will create a performance report, based on the rules for high performance web sites [4][12][13]. Inspecting the report will give us a better insight into what's happening, and how we can improve our implementation. General set of advices for performance improvements involve, but are not limited to, tuning PHP settings by turning off features that are not used, taking benefit from one of various PHP Accelerators, and optimizing Apache settings for specific environment Moodle is setup at.

Still on the topic of performance measuring and optimizations, we are going to look at database. As mentioned earlier, database being used is MySQL. Moodle contains a script which will display some key database performance statistics from the ADOdb performance monitor [10]. It can usually be reached on by pointing your browser to *dbperformance.php*, with path relative to Moodle installation.

Data collected that way can be used as a guide for tuning and improving performance. Although not mentioned here, one of the other ways to improve would be a switch to better-performance database. Possible optimization settings in Moodle for improving performance include caching as much as possible, reducing logs life-span, and other optimizations techniques available in Moodle's admin interface.

TABLE 2
RESULTS OF PERFORMANCE TESTS

Performance tests	First configuration	Second configuration
RAM	256	768
Server Software	Apache/2.2.4	Apache/2.2.4
Concurrency Level	10	100
Time taken for tests	30.602456 s	30.223306 s
Complete requests	46	51
Requests per second	1.50 [#/s]	1.69 [#/s]
Transfer rate	13.17 [Kb/s]	14.53 [Kb/s]
Optimizations	No	Yes

The results given in Table 2 show a clear benefit of optimizations and hardware upgrades. For consistency, both configurations were installed as Virtual Machines, with equal software platforms leaving no room for speculations about the reliability of results. The optimized and hardware upgraded configuration shows high concurrency, and can serve more requests then the initial configuration, which serves less requests even with concurrency of ten times lower then the second one.

C. Security

1) Preventing DoS attacks: availability attacks may occur at multiple points, ranging from server, router or entire network, focus will be on web server, and it's configuration to prevent possible availability attacks. Two attacks of such kind are known, simple DoS and distributed DoS. If you are facing the second, there is little to do, however with proper preparations its effect can be minimized. On the other hand, DoS can most of the time be completely eliminated. One of the steps that have to be done in order to stop it is surely setting *MaxClients* directive to desired maximum, causing a host-to-host attack to abort long before memory is exhausted. Generally, it is also recommend to install and setup *mod_evasive apache2 module*, an evasive maneuvers module for Apache to provide evasive action in the event of an HTTP DoS, DDoS attack or brute force attack. It is also designed to be a detection and network management tool, and can be easily configured to talk to various services, reporting abuses via email and syslog facilities.

2) Dealing with insecure cryptographic storage: most of the web application, including Moodle uses hashing algorithms to prevent others from discovering users' passwords, even if they get a hold of the database. However, this approach, while in theory and following mathematics is a really good one, is crackable by using a method called *Rainbow Tables*, a set of hash-plain pairs, which can be searched with great efficiency, and password can be broken [14]. Suggested way to fix the above mentioned problem would be to use the *bcrypt library* [21], utilizing optimized *Blowfish encryption*, which uses the idea of adaptive hashing. The advantage over other algorithms and libraries used for cryptographic storage is that you are able to configure its setup time, and this is where adaptive hashing shows its advantages. As computers get faster, the same block of code continues to produce passwords that are hard to crack.

3) Preventing buffer overflows: in most cases, buffer overflow problems can be avoided by either careful programming in languages which do not provide in-built buffer overflow protection (like C or C++), or by using more modern languages and their variations, like the C-language variants, Cyclone which uses method of attaching size information to arrays.

4) Information leakage and improper error handling: problem often found with web applications, especially those early in development whose publishing before they are really ready for public has become a big trend in today's Web 2.0 era, often come out without proper testing, and without disabling usual development parameters, one of the important one in this case being the so called *development mode* or *debug mode* which shows all errors and parameters passed to the applications, including, but not limited database connection options, its name, username and password. That allows random users of site to gain access to the database itself, and do malicious actions over it. Also debug messages could be analyzed to exploit potential security problems in web application. Suggested way to prevent this is an *automated production deployment strategy*, which would warn of any existing development settings, and advise you to switch to production mode. Also, it is very important to perform *code audit* before release in production, to avoid problems later.

5) Dealing with insecure communications: generally, this problem can be avoided by protecting parts of LMS site, especially those that are information-critical by SSL certificate. It can be either self-generated, or bought from one of the SSL vendors. It is also important to note that SSL causes increased load on the server itself, however it is crucial to protect user's privacy with this type of encryption.

6) Malicious file execution prevention: files are usually uploaded to */tmp* directory, which is writable by anyone, so it could be a potential exploit. To prevent such problems, it is advisable to modify filesystem, putting */tmp* on separate partition, and mounting in with *noexec* and *nosuid* properties, which would prevent problems caused by malicious file execution.

7) Miscellaneous advices: entire Moodle environment should be backed up regularly, and the easiest way to do it, is via *scheduled cron db dump* and *scp'ing it* to a secure location. When reported, security bugs get highest priority, so make to subscribe to relevant software and security mailing lists gaining a head-start in preventing security problems before users become aware of the potential exploits. Ubuntu distribution uses *apt* as it's frontend to *dpkg*, and it can be elevated to do upgrade simulation, sending us mail with upgrades that are available to our system. Although *rootkits presence* often shows the need of system reinstallation because it has been compromised, it is good to know when and if that happens by doing regular scans with *rkhunter* and *chrootkit*, all of which can be scheduled via *cron*. General advice is to keep your settings as paranoid as possible, while not causing troubles for the users, and includes strict *iptables* rules by only opening required ports, and those should be set to irregular value, especially the *ssh* port which should utilize *ssh-keys based authentication*, instead of password-based one. Passwords users generally choose are weak, and can be easily broken by social engineering or some other methods, and therefore it is advisable to enforce proper *passwords policy*, which can be configured in Moodle's admin interface.

V. CONCLUSION

Security settings depend on various software and hardware configuration factors. All these factors should be

taken into consideration in order to determine optimal security and privacy settings. In the domain of learning management systems, and with case study of Moodle, we identified four major groups of attacks. Most critical security flaws are classified in group of *authentication*, *availability*, *confidentiality* or *integrity attacks*. Short description on each of twelve security flaws is given from the LMS perspective, while recognition of different and possible vulnerabilities is first step in dealing with them.

Moodle architecture is divided into six different groups of modules. *Communication*, *productivity*, *student involvement*, *administration*, *course delivery* and *curriculum design* modules are recognized. Previously grouped security flaws are also discussed but in scope of particular module and with the purpose of critical spots determination. Finally, we tried to present set of concise advices regarding system *hardware upgrades*, *performance optimizations* or previously mentioned *security*. Given advices on *DoS attacks prevention*, *securing cryptographic storage*, *buffers overflow prevention*, *information leakage*, *improper error handling*, *securing communications and malicious file execution prevention* should be entry points into security and privacy safe learning management system implementation.

REFERENCES

[1] *** *Apache* [Online], "ab – Apache HTTP server benchmarking tool", Apache Software Foundation, 2008, Retrieved February 12 2008, URL: http://httpd.apache.org /docs/2.0/programs/ab.html

[2] *** *EduTools* [Online], "ArchiveCMS: Product Comparison System", 2008, Retrieved January 10 2008, URL: http://www.edutools.info/compare.jsp?pj=8&i=358

[3] *** *Elearning India* [Online], "Learning Management Systems", 2008, Retrieved January 24 2008, URL: http://elearning-india.com/content/blogcategory/19/38/

[4] *** *Firebug* [Online], "Firebug – Web development evolved", Parakey Inc., 2008, Retrieved February 10 2008, URL: http://www.getfirebug.com/

[5] *** *Learning Circuits* [Online], "Field Guide to Learning Management Systems", 2005, Retrieved December 30 2007, URL: , http://www.learningcircuits.org/

[6] *** *Moodle* [Online], "Moodle Official Site", Moodle.org, Retrieved December 20 2007, URL: http://moodle.org/

[7] *** *Moodle* [Online], "Moodle Official Statistics", Moodle.org, Retrieved February 11 2008, URL: http://moodle.org/stats/

[8] *** *OWASP Top Ten Project* [Online Report], Open Web Application Security Project - the open application security community, 2004, Retrieved January 25 2008, URL: http://www.owasp.org/index.php/Top_10_2004

[9] *** *OWASP Top Ten Project* [Online Report], Open Web Application Security Project - the open application security community, 2007, Retrieved January 25 2008, URL: http://www.owasp.org/index.php/Top_10_2007

[10] *** *PHPLens* [Online], "Adodb perfromance monitoring library", Retrieved February 15 2008, URL: http://phplens.com/lens/adodb/docs-perf.htm

[11] *** *University of Zagreb* [Online], "E-Learning Strategy 2007–2010", Strategy Development Comity, Zagreb, 2007, Retrieved February 14 2008, URL: http://rektorat.unizg.hr /fileadmin/rektorat/dokumenti/eucenje_strategija/University _of_Zagreb-E-learning_strategy.pdf

[12] *** *Yahoo!* [Online], "Exceptional performance", Yahoo.com, 2008, Retrieved February 1 2008, URL: http://developer.yahoo.com/performance/

[13] *** *Yahoo!* [Online], "Yslow for Firebug", Yahoo.com, 2008, Retrieved February 1 2008, URL: http://developer.yahoo.com/yslow/

[14] A. Biryukov, Some Thoughts on Time-Memory-Data Tradeoffs, *Cryptology ePrint Archive*, Report 2005/207, 2005, URL: http://eprint.iacr.org/2005/207

[15] C. Cowan, P. Wagle, C. Pu, S. Beattie, J. Walpole, "Buffer overflows: attacks and defenses for the vulnerability of the decade", *Foundations of Intrusion Tolerant Systems*, p. 227 - 237, 2003.

[16] A. Iyengar, E. Nahum, A. Sheikh, R. Tewari, "Enhancing Web Performance", *Proceedings of the IFIP 17th World Computer Congress - TC6 Stream on Communication Systems, Vol. 220*, p. 95–126., 2002.

[17] M. Jenkins, T. Browne and R. Walker, "VLE Surveys: A longitudinal perspective between March 2001, March 2003 and March 2005 for higher education in the United Kingdom", UCISA, 2005, URL: http://www.ucisa.ac.uk /groups/tlig/vle/vle_survey_2005.pdf

[18] N. Jovanovic, E. Kirda, C. Kruegel, "Preventing Cross Site Request Forgery Attacks", *IEEE Securecomm and Workshops*, p. 1 – 10, 2006.

[19] D. Moore, C. Shannon, D. Brown, G. M. Voelker, S. Savage, "Inferring Internet Denial-of-Service Activity", *ACM Transactions on Computer Systems (TOCS)*, Vol. 24, No. 2, p. 115 – 139, 2006.

[20] G. Ollmann, "Writing secure code", *Network Security*, Vol. 2007, No. 5, p. 16 – 20, 2007.

[21] N. Provos, D. Mazieres, "A Future-Adaptable Password Scheme", *Proceedings of the 1999 USENIX Annual Technical Conference*, June 6-11, 1999, Monterey, California, USA, 1999.

[22] P. Vogt, F. Nentwich, N. Jovanovic, E. Kirda, C. Kruegel, G. Vigna, "Cross-Site Scripting Prevention with Dynamic Data Tainting and Static Analysis", *Proceedings of the Network and Distributed System Security Symposium*, 2007.

[23] D. R. Woolley, "Plato: The Emergence of Online Community", *Thinkofit*, 1994, URL: http://thinkofit.com /plato/dwplato.htm

[24] W. Xu, S. Bhatkar, R. Sekar, "Practical dynamic taint analysis for countering input validation attacks on web applications", *Technical Report SECLAB-05-04*, Department of Computer Science, Stony Brook University, 2005.