# E-Learning Software Security

## Tested for Security Vulnerabilities & Issues

George E. Violettas, Tryfon L. Theodorou
ICT Department
Technical Trainers College (TTC)
Riyadh, Kingdom of Saudi Arabia

George C. Stephanides
Department of Applied Informatics
University of Macedonia (UoM)
Thessaloniki, Greece

***Abstract*— This paper intends to shed light on the security problems faced by e-learning software; which, until now has not been systematically tested for vulnerabilities and weaknesses. These result to web programs exposed to attacks that are fairly easy to implement. Additionally, e-learning software deals with intelligent, studious individuals; The intended audience for the product has the knowledge, or is potentially being taught the knowledge, to compromise the software. The problem is that in a situation like this, the compromised software remains unrevealed, probably for ever, because the beneficiaries have absolutely no need to reveal their findings, but rather exploit the very best of it.**

***Keywords—electronic learning; information security; web design;***

## I. INTRODUCTION

The term e-learning is used to describe learning using electronic means: the acquisition of knowledge and skill using electronic technologies such as computer and Internet-based courseware and local or wide area networks.

Most major universities of the world offer online courses and study material under the umbrella of a software distributing suite. The software has some common characteristics whether it is open source or commercial. The various software suites serve the same purpose, addressing the issue in the same way more or less; material is divided into modules and lessons with different levels of user's rights depending on the user's role, i.e. student, teacher, and administrator. These users are further classified based on their individual role, year of study, enrolled lessons, and assigned exercises. Many university software suites offer free access to a portion of the suite to users not affiliated with the university.

All the above described complicated hierarchy, along with the complexity and differentiation of the teaching material and purpose (video, audio, pdf, or even executable programs) must be organized, secured and well served in an environment of academic freedom where research, curiosity and even pranks are allowed and encouraged!

In this paper there is a primary detailed security evaluation for some of the more popular open source software offerings. There is also a detailed description of the most common vulnerabilities found in today's web applications and some initial suggestions on mitigating these vulnerabilities in the programming stage.

There are also some suggestions for future work about addressing the security of web applications in general and how this issue should be raised more concerning all involved parties such as programmers, designers, participants.

This work was financially and morally supported by Technical Trainers College (TTC), an accredited Technical University in Riyadh, Kingdom of Saudi Arabia. (TTC is a project run by the Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH).

## II. LITERATURE REVIEW

### A. e-Learning & Web Applications

In [1] the authors examined the privacy principles that have to be applied during the implementation of e-learning software, along with the identification of components that require privacy and security safeguards. All this is described in a pure theoretical level of prevention.

In [2] the authors admit that in an environment such as an e-learning suite, the exchange of personal and collaborative data is intense, so a strong protection of participant's privacy should be an absolute must. In the same paper we find that users (students) are highly concern about their privacy, the use (and misuse) of their personal data and are highly concern about the possible hijacking of it. Students are also concerned about the usage of data that seems not to be very "personal", such as like browsing history, material downloaded etc. This seems to be the concern of the student that the teacher will discover the student's weaknesses or habits and manipulate future exams or tests.

Concerning the web applications security, one of the best surveys available is the Web Application Security Consortium (WASC), Web Application Security Statistics Project [3]. This project was a collaborative industry wide effort to pool together sanitized website vulnerability data and to gain a better understanding about the web application vulnerability landscape. The project had some extraordinary findings, such as the following: (1) more than 13% of the sites tested can be compromised automatically with attacks such as BFA (Brute Force Attack), BO (Buffer Overflow), OSC (OS Commanding), Path Traversal, Remote File Inclusion, SSI Injection, Session Fixation, SQLi (SQL Injection), Insufficient Authentication, or Insufficient Authorization (2) about 49% of web applications contain vulnerabilities of high risk level detected during automatic scanning; a manual or white box automated assessment detected a probability of 80-96% of those vulnerabilities.

Among the conclusions of this project were the following:

- The most widespread vulnerabilities are XSS (Cross-site Scripting), different types of Information Leakage, SQLi, HTTP Response Splitting;

- The probability to detect an urgent or critical error in dynamic web application is about 49% by automatic scanning and 96% by comprehensive expert analysis
- Administration issues are 20% more frequently the cause of a vulnerability than system development errors.

Since e-learning software is web software with its own diversities, a targeted essay like this can highlight the vulnerabilities.

In [4] there is a very interesting finding: after an actual research the authors conclude that any kind of participation (active or passive) is positively related to e-learning performance. So, the more some students participate to the e-learning environment provided, the better they become, or the greeted knowledge they acquire. But then those students are more exposed to the security risks and lacks of such a system, or they are potential attackers of the system due to the increased knowledge they have about it.

### B. *Vulnerability Scanners*

In [5] the authors did a very comprehensive detailed analysis of black-box web application vulnerability scanners. Since this was not a commercial product comparison but a potential future research there is no final score about the applications tested, but there are some very important findings:

*1)* All the applications focused on the vulnerabilities considered to be the most serious and widespread in the previous chapter, i.e. Information Disclosure of some kind, XSS, SQLi, and other forms of XCS (Cross Channel Scripting).

All the applications were mandated to search for all categories of the OWASP Top-10 [6]. Although the applications searched, named and grouped various vulnerabilities in a very different way, the results showed that scanners devote most testing to information leakage vulnerabilities, followed by XSS and SQLi vulnerabilities, which is absolutely consistent with all the above findings about the importance and potential danger of those attacks.

The authors claim that the scanners in total did a generally good job of detecting previously known vulnerabilities. They did particularly well in the Information Disclosure and Session Management classifications, leading to the hypothesis that effective test vectors are easier to add for these categories than others. The scanners also did a reasonable job of detecting XSS and SQLi vulnerabilities, with about 50% detection rate for both. But if we analyze these results in another way, we see that, in the best case, the scanners find 6 out of 10 XXS type 1 vulnerabilities (those considered to be the easiest ones), while they find only 15% of XSS Type 2, 21% of SQLi, 30% of information leakage and 20% of XCS.

CWE International (Common Weakness Enumeration) is a nonprofit organization trying to provide a unified, measureable set of software weaknesses depending on the specific application and target industry. So there is a scoring system [7], with which each user can assign specific weights to every weakness depending on the nature of the targeted business and end up with a measurable score of how "safe" the application in question is.

As a result, those scanners and their results should be just a guide or a measure and not at all a final reassuring report that the application in question is safe. If and when all the security holes and vulnerabilities indicated by at least two of those scanners are all fixed, it is really up to the programmer of the application to test it again since new attacks and methods are circulated every day in the wild.

## III. E-LEARNING DEFINITION

We can still find an open debate about what exactly e-learning is and what its accurate definition is. For example in [8],[9] the authors do not reach a solid conclusion about the definition of e-learning and the difference (or not) between e-learning, online learning and blended learning.

For the needs of the current paper, we define as e-learning software, the web based suite of programs, under any kind of license (open source, free or commercial) that is used to provide access to individuals in studying material and also provides the means to both students and teachers to interact and communicate asynchronically. This communication may include but not be limited to exchange of e-mails, messages, participating to bulletin boards, chat rooms, uploading and/or downloading any kind of content that is available and suitable for the needs of the lesson or seminar in question.

One of the most comprehensive definitions of e-Learning could be the following "…*We will call e-Learning all forms of electronic supported learning and teaching, which are procedural in character and aim to effect the construction of knowledge with reference to individual experience, practice and knowledge of the learner. Information and communication systems, whether networked or not, serve as specific media (specific in the sense elaborated previously) to implement the learning process*…" [10]. This definition is considered to be accurate and wide enough to accommodate all the e-learning usages today, and is perfectly describing all the software being evaluated in this paper.

## IV. NECESSITY OF THIS RESEARCH

As an answer to the obvious question why the current research is important, it is very well mentioned in [11] *"…Technology is often marketed as capable of doing 'just what you want it to do'. However, there is lack of a matching educational technology to be used by the youth, or the speed at which they are being developed is not at par with what the youth expect…"*

In [12] the author is evaluating e-learning software packages. For this purpose he is examining the multiple criteria evaluation of Learning Object Repositories (LORs) and their technological quality based on score ranking results. There is a comprehensive set of criteria on a very detailed schema where security is "buried" in *Architecture* between *Performance and Scalability* and *Interoperability*! E.g. *Graphical User Interface (GUI)* is under *Quality in use* criteria, and has nothing to do with *security*. This is the main problem still today about security: Researchers don't even think that the primary parameter about the *GUI* or e.g. *Full Text Search* (another

criterion) is not whether it exists or not, or if it works well (which is obviously important) but how safe it is and if it leaks important information or not!

Security should be addressed as an overall criterion of the total software and not as another minor aspect we have to take care of, sometime in the future.

## V. WEB SOFTWARE VULNERABILITIES

### A. SANS Institute

The SANS (SysAdmin, Audit, Network, Security) Institute mentions the top 25 software errors [13]. For various reasons, not all of them apply to web based e-learning software[1]. The most serious of those mentioned that can be applied here, are: (1) SQL Injection (2) Cross Site Scripting (3) Buffer Overflow or similar weaknesses (4) Sensitive Information transmitted over http (Not over https). (Fig.1). The latter has a very broad field of exploration, including credential(s) interception, magic URL & hidden form fields, eavesdropping, spoofing, replay, tampering, hijacking.

### B. OWASP Organization

In OWASP Top 10-2013 report [6], the non-profit organization is addressing the vulnerabilities to be the most important issues to address in a software security testing: The OWASP Top-10 is based on risk data from 8 firms that specialize in application security, including 4 consulting companies and 4 tool vendors (2 static and 2 dynamic). This data spans over 500,000 vulnerabilities across hundreds of organizations and thousands of applications. The Top-10 items are selected and prioritized according to this prevalence data, in combination with consensus estimates of exploitability, detectability, and impact estimates.

The OWASP Top-10, 2013 is:

A1 – Injection, A2 – Broken Authentication and Session Management, A3 – Cross-Site Scripting (XSS), A4 – Insecure Direct Object References, A5 – Security Misconfiguration, A6 – Sensitive Data Exposure, A7 – Missing Function Level Access Control, A8 – Cross-Site Request Forgery (CSRF), A9 – Using Known Vulnerable Components, A10 – Unvalidated Redirects and Forward. (Αυτά μπορούμε να τα βάλουμε το ένα κάτω από το άλλο για να μεγαλώσουμε το κείμενο)

### C. Problem Classification

While the above two lists (SANS vs OWASP) are not identical; they address the problems differently, with different names and slightly different descriptions. There is no vulnerability or problem in the one that is not mentioned into the other.

In this paper, we will not deal at all with the security problems arising from social engineering techniques or bad usage of security measures such as weak passwords, or any kind of revealing security related information, e.g. version of the http server (apache) of the installation. Those are all beyond the scope of this work. Such matters are addressed within the organization security measures such as security policy, risk management, risk analysis, software updates, etc.

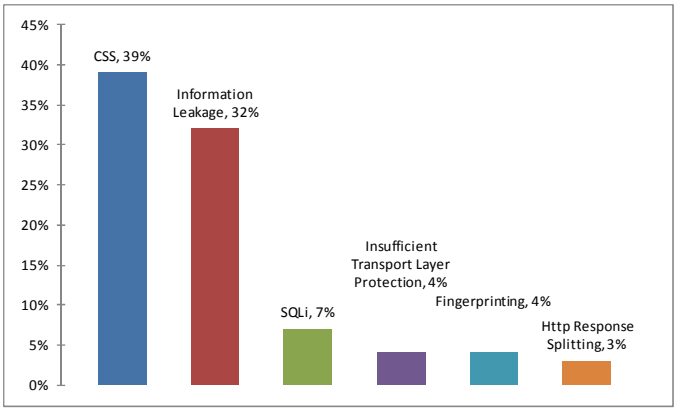


Fig. 1. The most widespread vulnerabilities in web applications [3].The 3 first categories are by far very important, because of the wild spread and also of the devasting consequences the can have over a system. They all lead to the full "hijacking" of a system or equivelantly serious situations.

## VI. VULNERABILITIES CHECKING SOFTWARE

Security Vulnerability Testing Software is a type of software that intends to find vulnerabilities of a web application and determine that its data are protected from possible intruders. There are various Security Vulnerability Testing software applications in the market providing either free or paid software editions. For the needs of the current study two different software packages, Netsparker and N-Stalker were used.

### A. Netsparker

Netsparker Community Edition is developed by the "mavitunasecurity" team as a web vulnerability scanner [14]. The Community Edition is a free edition that shares many features with professional edition. It can detect XSS issues and various forms of SQLi vulnerability, which are the main purpose of this study. It also includes Error Based, Boolean, Blind and Time Based SQLi. It detects Permanent-Stored and Reflective XSS in two ways either via remote file injection or inside the URL Address. Moreover it searches for Local File Inclusion and Arbitrary File Reading issues like if an attacker can access files and source code from the server on both Windows and Linux systems. Another important issue is that it can identify if the website sends passwords over HTTP. Netsparker provides a detailed report for the findings to ensure that the issue can be correctly addressed by developers. It displays solutions besides the issues and enables you to see the browser view and HTTP request/response. Netsparker was very helpful not only on identifying the errors listed below, but it also gave detailed description and zone of impact of each error. Some of the descriptions of the secondary errors in chapter VII are based on those descriptions. Netsparker also proceeds to suggestions and solutions for each specific problem that should be followed.

[1] E.g. Number two weakness CWE-78 is about Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection'). Such a thread although it exists also in web based e-learning software, is beyond the scope of this paper, because it needs some way of access to the machine where the software lies physically. This scenario is extremely unlike in most cases, so it is excluded from this research. Even if this scenario happens, it falls to the category of system or network security of the organization in question and not for the specific software in use.

### B. N-Stalker

The second software that was used for this study is the free edition of N-Stalker [15]. N-Stalker Web Application Security Scanner 2012 Free Edition provides a restricted set of free Web Security Assessment checks to enhance the overall security of a web server infrastructure, using a web attack signature database and giving the user the options to conduct security scanning based on custom or OWASP testing guides. While the free edition of N-Stalker is restricted compared to the full edition, it has all the needed features for the purpose of this paper. N-Stalker is founded upon the U.S. Patent Registered Technology of Component-oriented Web Application Security Scanning. N-Stalker allows for a quick assessment of Web Applications under the secure development life cycle (SDLC) perspective. N-Stalker Free Edition provides the assessment of 3rd-party packages as well as web server vulnerabilities. A restricted set of attack patterns such as CSS is also included.

The application testing one of the applications chosen in the next chapter for testing is depicted in Fig.2.

## VII. E-Learning Software Tested

Today dozens of e-Learning software platforms exist and are under heavy usage from education institutes and students all over the world. In this study we selected four of them which are:

1) *ATutor*
2) *CommSy*
3) *eFront*
4) *Moodle*

The reasons for selecting these four (4) e-Learning platforms are the following:

1) *They are all open source applications.* There is such a big difference in the price of non-free e-Learning applications, that a robust budget is needed to buy all of them (or long term negotiations with the selling companies) and, if someone is purchasing a software package the cost of the application may weigh more in the decision than the security offered

2) *They are widely accepted in the education community:* All the above software is widely accepted today by the academic community [16-19] with dozens of installations in universities and other institutes. Of course there is other software we know of, some of them also popular today. In a future additional study, less popular application software will be reviewed.

3) *They are easy to use:* Although this is a very vague term and it should definitely be measured after a detailed questionnaire and examined in a future survey, the general consensus the authors received from questioning a wide variety of members of the academic community.

4) *A lot of installations exist in the wild today:* The authors are aware of many institutions using the above software packages. Specific reasons for current usage is, of course, another topic for a future survey

5) *Most important, there are no comparisons of popularity installation sites or any other criteria:* What is the main reason or justification of the criteria mentioned above is the fact that the authors were not able to find any kind of survey or other work which will deal with number of installations, popularity of this software, user number, satisfaction etc

### A. ATutor

ATutor [16] is an Open Source Web-based Learning

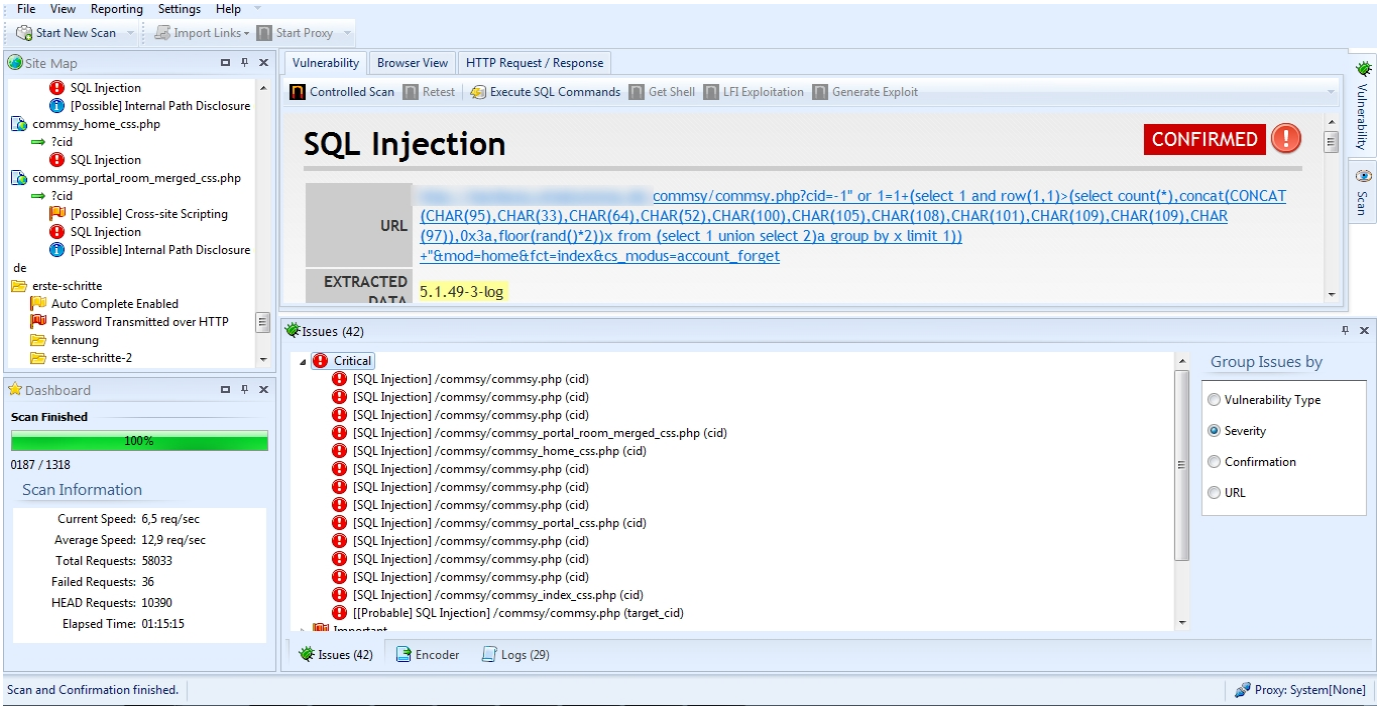


Fig.2 A screenshot of N-Stalker in action. An SQLi is confirmed in one of the four E-Learning Software investigated. The beginning of the URL was blured for obvious reasons. Keep in mind that there is still a big distance between a simple SQLi and an actual possibility of damaging the system or extractiing information, but years of experience say that if it is possible, it is doable!

Content Management System. It is very easy to install or update and extend its functionality. ATutor has also the IMS/SCORM (Sharable Content Object Reference Model) content packaging specifications, allowing content developers to create reusable content that can be swapped between different e-learning systems.

### B. eFront

eFront [17] is an easy to use, visually attractive, SCORM compatible, eLearning system. Most of the interface options are self-explanatory. eFront is an Ajax enabled, Unicode, LDAP and SCORM supporting, multilingual eLearning platform. Also in eFront are integrated sound pedagogical concepts that guide users and keep them motivated. Although it is distributed as free software, eFront is being supported by a professional team of highly skilled developers.

### C. CommSy

CommSy [18] is an open source web-based community system software, originally developed at the University of Hamburg, Germany, to support learning communities, networks and project work. CommSy is designed in order to be easy to use, easy to learn and easy to modify. It is set by open standards and therefore easy to add to other media. It can also easily be integrated into existing infrastructures. CommSy is used nationwide in diverse fields of schools, universities, teacher training, freelancer networks, extra curriculum learning and by study groups. An actual installation of the software is depicted in Fig. 3 under a primitive attack where a script has successfully passed through one of the application's security holes.

### D. Moodle

Moodle [19] is a Learning Management System (LMS). It is an open source web application used as a tool for creating online dynamic web sites for teaching. Moodle is very popular among educators because it provides educators many tools to manage and promote learning. It needs to be installed on a web server either on one local computer or one at a web hosting company. Moodle has features that allow it to scale from small to very large deployments, from few to hundreds of thousands of students. It can be used form a primary school or an education hobbyist to institutions using it as the main platform to conduct online courses.

## VIII. The Study

This study was based on the counting of vulnerabilites of software today as they are mentioned in [9], and are depicted in Fig.6.

### A. Computer Lab Setup for Conducting the Study

In order to conduct the study a lab was setup with the following setting:

- One server running Linux operating system with Apache web server software and MySQL database system. On that local web server four websites were deployed with the four under investigation e-learning web applications. All of them were downloaded from their official web sites selecting the most recent stable versions.
- Two clients running web browsers under Microsoft Windows 7 Operating System. On client No 1 the Netsparker 2.5.2.0 Community Edition was installed and on client No 2 N-Stalker Web Application Security Scanner Free edition.

### B. Study Results - Most Important Vulnerabilities

In the current study because of a lack of space and based on importance we mostly discussed the most common (near the 50%) of the security vulnerabilities in web today and most dangerous, which are XSS, Information Leakage, SQLi, and Insufficient Transport Layer Protection (Fig. 1). As a rule, XSS, SQLi and HTTP Response Splitting vulnerabilities are caused by design errors, while Information Leakage, Insufficient Transport Layer Protection and Fingerprinting are often caused by insufficient administration (e.g., access control) [3].

*1) XSS Vulnerability:* Cross Site Scripting (XSS) is one

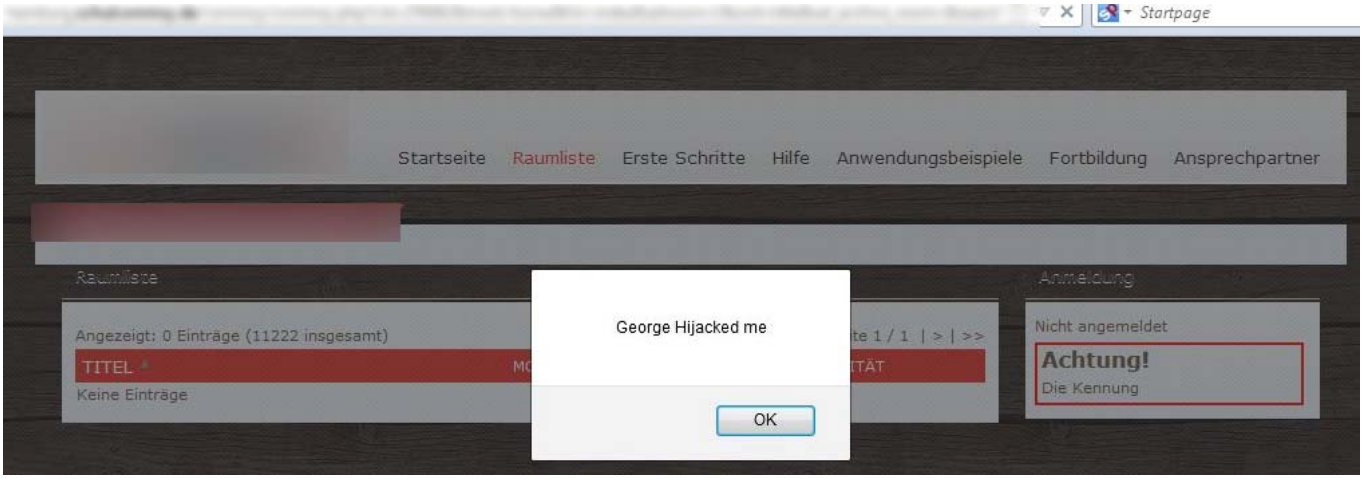


Fig.3 An actual "Commsy" Installation after a successful XSS attack. Some parts (URL, Logo) are blured for obvious reasons. Notice that this is a very primitive initial testing of the applications vulnerabilities. In one of the application's input fields (e.g. text input field where the user will input her e-mail) a malicious code such as (<SCRIPT>alert("George Hijacked me")</SCRIPT> ) is inserted. Because this input is not checked for vadility (sanitized) the ambitious hacker can launch a CSS attack through this field

of the most common application layers attacking techniques. In addition, other attack methods, such as, Content Spoofing and Information Disclosure could be side-effects of an XSS attack. In general, XSS refers to that hacking technique that leverages vulnerabilities in the code of a web application toallow an attacker to send malicious content from an end-user and collect some type of data from the victim. XSS allows an attacker to embed malicious JavaScript, VBScript, ActiveX, HTML, or Flash into a dynamic web page to fool the user, executing the script on his machine in order to gather data. The data is usually formatted as a hyperlink containing the malicious content and which is distributed over any possible means on the internet.

Any e-learning application can be vulnerable to this type of attack because the majority of e-learning platforms pass parameters to a database. Usually these are present in Login forms, Search fields, Forgot Password forms, etc… The use of XSS might compromise private information, manipulate or steal cookies, create requests that can be mistaken for those of a valid user, or execute malicious code on the end-user systems.

In Fig.4 we can see –among the rest- the results of the XSS investigation that was conducted on the above four e-learning platforms using the two vulnerability search software applications. In order to investigate the accuracy of the results and double confirm the above vulnerability we conducted manual XSS attack on the e-learning web sites that were found vulnerable from the security testing software(s).

E.g. in the "Comssy" e-learning web application in a particular field a script was inserted that was executed from the application and returned the result that is depicted in Fig.3 proving the XSS vulnerability. Since the reason of this study is the confirmation of the existence of vulnerability, no further action was taken i.e. gathering sensitive data from the web site, because this would be obviously as a hacking attack, which beyond the scope of this paper.

*2) SQL Injection Vulnerability:* SQLi is a type of security exploit that exploits the security vulnerability in application in which the attacker adds Structured Query Language (SQL) code to a Web form input box or URL address to gain access to resources or make changes to data at the database layer of an application. It is one of the most common vulnerabilities found in websites that can be easily exploited due to poor coding. Even though it is very common, very little attention is given to prevention of these types of attack.

SQLi is an ideal technique that can be used in order to attack data driven applications like e-learning web applications. The impact of such attack can be leakage of sensitive information, modification of sensitive information, loss of control of the database server, data loss and denial of service. Preventing SQL injection is a matter of careful database communication coding.

In Fig.4 we can observe the results of the SQLi investigation that was conducted on the four e-learning platforms using the two vulnerability search software.

*3) Password transmission over HTTP Vulnerability:* Using unsecure channel for transmitting credentials is the most basic reason for information leakage (responsible for the 32% of the identified vulnerabilities in Fig.1. it is not mandatory; as such, all of the installations we reviewed in the wild were installed with plain http resulting in the presence of the most common vulnerability

| **E-Learning Platform** | **Main Vulnerabilities Scan Confirmation** | | |
|---|---|---|---|
| | ***XSS*** | ***SQLi*** | ***Info over http*** |
| Atutor | Yes | No | Yes |
| Commsy | Yes | Yes | Yes |
| eFront | No | No | Yes |
| Moodle | No | No | Yes |

Fig.4 Main results of the vulnerability testing for all four e-Learning platforms examined. Notice here that if the vulnerability was discovered from only one of the two vulnerability checking software, this is simly noted as existing, because this is clearly the inability of the other sotware to discover it. So far a false positive was not found, whilst a false negative is obviously just an inability.

## C. Study Results - Other Minor Vulnerabilities

After the tests for the critical vulnerabilities, both testing software applications were run to test the other vulnerabilities they are capable of searching for as depicted in Fig. 5 below:

| **Secondary Vulnerabilities** | **Minor Vulnerabilities Scan Results** | | | |
|---|---|---|---|---|
| | ***Atutor*** | ***CommSy*** | ***eFront*** | ***Moodle*** |
| Internal Server Error | No | Yes | Yes | Yes |
| Cookie not marked as HTTPOnly | Yes | Yes | Yes | Yes |
| Auto Complete Enabled | Yes | Yes | Yes | Yes |
| Version Disclosure (PHP) | Yes | Yes | Yes | Yes |
| Cookie Not Marked as Secure | Yes | No | No | No |
| Database Error Message Disclosure | No | Yes | No | No |
| Programming Error Message | Yes | No | No | No |

Fig. 5 Results of the secondary vulnerability testing for all four e-Learning platforms examined. Although they are called minor, some of those vulnerabilites should be considered very seriously because they might look innocuous or minor at the first glance, but they can easily lead to a major security breech.

Those vulnerabilities are:

*1) Cookies Vulnerabilities*

- *Cookie not marked as HTTPOnly:* HTTPOnly cookies cannot be read by client-side scripts. As a result if cookies are marked as HTTPOnly, they can provide an additional layer of protection against XSS attacks. The impact of that vulnerability is not direct; it comes as a side effect combined with a XSS attack. In that case the attacker can easily access cookies and hijack the victim's session, in particular to steal

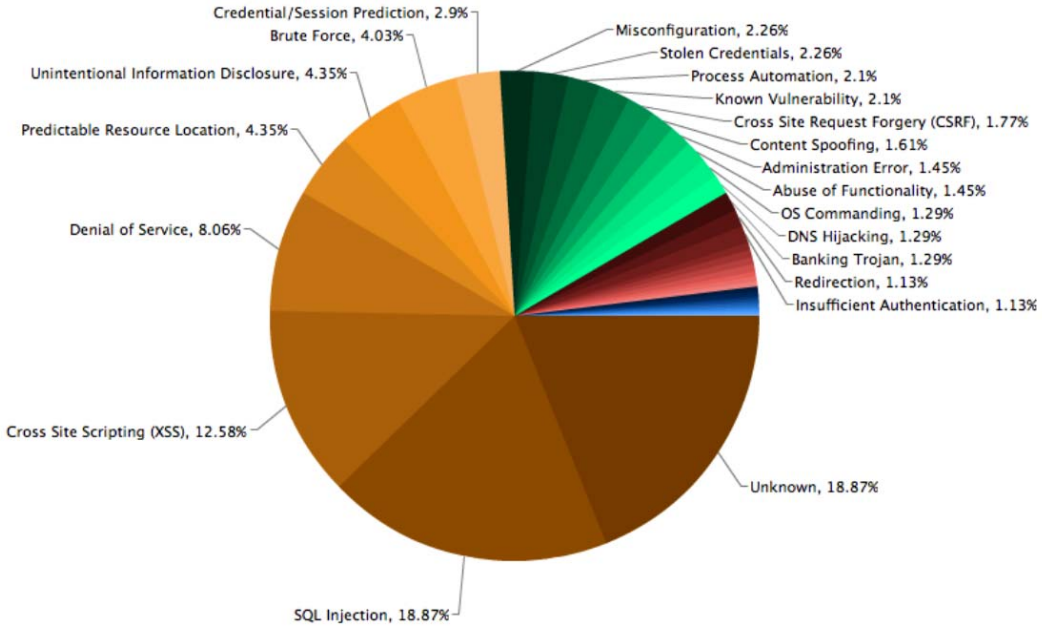


Fig. 6 A full list of the vulnerabilities that can be found on a software today. Some of them might look harmless, but when combined together they can lead to fully comrpomise the system in question [20]

cookies via JavaScript. Unfortunately, all four of the e-Learning applications under investigation, were found vulnerable to this since they don't mark cookies as HTTPOnly. In order to avoid this type of problem, developers have to mark the cookie as HTTPOnly and that will be an extra layer of defense against XSS. However that alone does not mean the application is secure because there are other ways to use XSS attacks.

- *Cookie Not Marked as Secure:* A Cookie that in not marked as secure and not transmitted over HTTPS could potentially be stolen by an attacker. Therefore if this cookie is a session cookie, an attacker might intercept it and hijack a victim's session and decrypt the traffic. If the attacker manages to carry out a "Man in the middle" attack, she can force the victim to make an HTTP request to steal the cookie. But in order to exploit this vulnerability the attacker requires local access to the web server or victim's network. The conducted study found that "ATutor" is vulnerable because it has cookies which are not marked as secure. Marking all cookies as secure can protect the application.

*2) Auto Complete Enabled*

If the "Auto Complete" property is set to ON in the form tag or to individual "input" fields of an application, data entered in these fields will be cached by the browser. An attacker with access to the victim's browser could steal this information. This can be more serious if these fields are either "password" fields or important fields in general. All four of the e-Learning applications were found vulnerable to this because the security scanners showed they have important input fields with the "Auto Complete" property set to ON.

*3) Version Disclosure (PHP)*

The disclosure of the PHP version in its HTTP response from the application web server might help an attacker to gain a greater understanding of the system. The disclosed information can potentially develop further attacks targeted at the specific version of PHP. All four of the e-Learning applications disclose such information, letting an attacker to harvest specific security vulnerabilities for the identified version.

*4) Internal Server Error*

Security scanners report as vulnerability the case that a web application Server responds with an "HTTP status 500" error. This server-side error should be analyzed carefully because it can indicate further vulnerabilities that may vary depending on the conditions. In most cases this indicates poor coding practices, not enough error checking and sanitization.

A good practice in order to avoid this type of vulnerability is to review the application code in order to handle unexpected errors, generally not to disclose further information upon an error and handle all errors at server side only. In the current study, most of the e-Learning applications were found to be vulnerable to Internal Server Errors.

*5) Database Error Message Disclosure*

An error message directly from the database server of a web application is vulnerability because it is one of the most critical indications that a web site can be vulnerable to a SQLi. In the current study this was confirmed because CommSy is the only one (of the four) e-Learning software found to disclose Database Error Messages and as mentioned earlier, it

is vulnerable to SQLi as well. Handling database server error messages without exposing them to the user is an absolute must nowadays and should be given the proper attention.

*6) Programming Error Message*

Disclosing run time programming errors to the user without prior handling from the application indicates poor programming technique, but it can be also a vulnerability issue. Disclosing uncontrollable error messages can disclose sensitive information which may be used by an attacker to mount new attacks. Atutor was the only e-Learning application in this study which appears to be vulnerable because of programming error messages. Saving error messages to the backend storage such as a log file using some encoded text as well is a good error handling technique that can avoid further security pit falls.

## IX. Further Work & Discussion

There are two basic procedures on testing a software package: (1) white box, (2) black box. They both try to find security flaws in the software, but using completely different approaches depending whether there is access to the source code or not [20].

In the case of white-box testing there is access to the source code, so there is an extensive search through the code and the overall design of the software. In such a case there is a great possibility of finding a majority of (up to 90% sometimes) possible errors and flows in the software and even find flaws during the risk analysis. The only disadvantage of white-box testing is that it might return some false negatives.

Black-box is testing an actual running program. It can be done against the real installation of the program (in the case of this paper, web software) or on an installation especially for this purpose, but without the knowledge or access to source code or binary code. The importance about black box is that if a vulnerability or flow is found, this is absolutely real since it was found on the actual version of the software and not on the ideal world of the lab and the source code. An example of black-box testing is depicted in Fig. 3 where an actual XSS attack took place against an actual installation on the wild.

It is obvious that a complete test of software should include both tests, including the search for all vulnerabilities depicted in Fig. 6. Both tests can be done today by automated tools (like the two software packages used in chapter VI).

A future work extending the current paper would explore the following paths:

1) *Categorize and Analyze basic problems in software design*. Especially in php on which a high percentage of web installations exist, it is possible to properly distinguish and suggest correction methods basically for the three (3) major flaws depicted in Fig. 1,4.

2) *Construct a parser which will check php code for the above errors systematically*. The parser will correctly distinguish the flaws and probably suggest the correction automatically.

Both the above could lead to the construction of an open source tool for finding and correcting security errors in software.


## Acknowledgment

The authors wish to thank Michelle Gately for proofing the document.


## References


[1] K. El-Khatib, L. Korba, Y. Xu, and G. Yee, "Privacy and security in e-learning," *International Journal of Distance Education Technologies (IJDET)*, vol. 1, no. 4, pp. 1–19, 2003.

[2] M. May and S. George, "Using students' tracking data in E-learning: Are we always aware of security and privacy concerns?," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, 2011, pp. 10–14.

[3] WASC, "The Web Application Security Consortium," *Web Application Security Statistics*, 2008. [Online]. Available: http://projects.webappsec.org/w/page/13246989/Web%20Application%20Security%20Statistics. [Accessed: 01-Apr-2013].

[4] E. Y. Huang, S. W. Lin, and T. K. Huang, "What type of learning style leads to online participation in the mixed-mode e-learning environment? A study of software usage instruction," *Computers & Education*, vol. 58, no. 1, pp. 338–349, 2012.

[5] J. Bau, E. Bursztein, D. Gupta, and J. Mitchell, "State of the art: Automated black-box web application vulnerability testing," in *Security and Privacy (SP), 2010 IEEE Symposium on*, 2010, pp. 332–345.

[6] J. Williams, "OWASP Top 10 - 2013 - RC1.pdf," OWASP, rc1, Mar. 2013.

[7] CWE, "CWE - Common Weakness Scoring System (CWSS)," 2011. [Online]. Available: http://cwe.mitre.org/cwss/. [Accessed: 02-Apr-2013].

[8] J. L. Moore, C. Dickson-Deane, and K. Galyen, "e-Learning, online learning, and distance learning environments: Are they the same?," *The Internet and Higher Education*, vol. 14, no. 2, pp. 129–135, 2011.

[9] A. Sangrà, D. Vlachopoulos, and N. Cabrera, "Building an inclusive definition of e-learning: An approach to the conceptual framework," *The International Review of Research in Open and Distance Learning*, vol. 13, no. 2, pp. 145–159, Feb. 2012.

[10] D. Tavangarian, M. E. Leypold, K. Nölting, M. Röser, and D. Voigt, "Is e-learning the Solution for Individual Learning," *Electronic Journal of E-learning*, vol. 2, no. 2, pp. 273–280, 2004.

[11] J. K. Njenga and L. C. H. Fourie, "The myths about e-learning in higher education," *British Journal of Educational Technology*, vol. 41, no. 2, pp. 199–212, 2010.

[12] E. Kurilovas, "Evaluation and Optimisation of e-Learning Software Packages: Learning Object Repositories," in *Software Engineering Advances, 2009. ICSEA'09. Fourth International Conference on*, 2009, pp. 477–483.

[13] SANS/CWE, "TOP 25 Most Dangerous Software Errors," 27-Jun-2011. [Online]. Available: http://www.sans.org/top25-software-errors/#cat1. [Accessed: 01-Apr-2013].

[14] Netsparker, "Netsparker Web Application Security Scanner." [Online]. Available: http://www.mavitunasecurity.com/about/. [Accessed: 05-Apr-2013].

[15] N-Stalker, "N-Stalker The Web Security Specialists." [Online]. Available: http://www.nstalker.com/. [Accessed: 05-Apr-2013].

[16] Atutor, "Atutor Learning Managment Tools." [Online]. Available: http://atutor.ca/atutor/download.php. [Accessed: 07-Apr-2013].

[17] eFront LMS, "Enterprise Learning Management System Software." [Online]. Available: http://www.efrontlearning.net/. [Accessed: 07-Apr-2013].

[18] Commsy, "Home Page." [Online]. Available: http://www.commsy.net/. [Accessed: 07-Apr-2013].

[19] Moodle.org, "open-source community-based tools for learning." [Online]. Available: https://moodle.org/. [Accessed: 07-Apr-2013].

[20] G. McGraw and B. Potter, "Software security testing," *IEEE Security and Privacy*, vol. 2, no. 5, pp. 81–85, 2004.