OWASP PHP Security Training System

Timo Pagel
University of Applied Sciences Kiel
timo.pagel@owasp.org

Abstract

Web applications are omnipresent in business nowadays. The security of these applications is of particular importance, because security breaches could influence the good reputation of a company. One cause for flaws in a web applications is not sufficiently trained developers. The OWASP PHP Security Training System (PSeTS) *teaches developers* to attack and more importantly defend web applications. When working through the attack part, developers have to strike against a vulnerable application. Through this, they learn to *think like a hacker*. Weaknesses to detect and exploit are for example XSS or SQL Injections which are listed in the OWASP Top Ten 2013.

While operating in the defense part, the developer is introduced to secure the vulnerable application by safeguarding the code. As the system is not used for competitions, it allows real *code execution in the defense part*.

Keywords OWASP PSeTS; php security; web application security; e-learning; interactive training system

1. Introduction

PHP has 82% share in the web programming languages market as by December 2014 [1]. This emphasises the need for web application security education in PHP. The Open Web Application Security Project (OWASP) maintains several security related projects.

Common developer security training systems focus on the attack of vulnerable web applications. PSeTS provides a didactically oriented system consisting of several units. Every unit is divided in an attack and a *defense part*. Knowledge transfer of the system has to be evaluated. Additionally, the usability and the motivation enhancement has to be tested.

PSeTS is published as an OWASP project with a presentation including screenshots on the OWASP website[1].

This paper is organized as follows: Section 2 describes related interactive training systems. Section 3 provides background information for interactive training systems. Section 4 to 5 describe PSeTS and the technical realization. Section 6 gives an overview of the evaluation. Finally, in Section 7 the summarized results are being presented, and information about future work is provided.

2. Related Work

Security training applications are OWASP Web Goat, OWASP Hackademic or OWASP Mutillidae II. The developer takes or completes the following steps in a unit:

1. Read initial instructions
2. Attack a vulnerable application in the same window
3. Optional: Search for further information in the references

The references often include defense techniques, but there is no defense part inside a unit.

For a developer it is hard to use tools like Firebug to inspect the vulnerable source code of the application. The vulnerable application is on the same page as the instructions. This might confuse the developer. The advantage of the systems is that developer's source code is not excuted on the server. This provides the ability to run the applications on a public server or conduct a competion.

3. Background

Education in the class room is created and delivered by one or two trainers. In computer based training the trainer creates learning assets, which deliver knowledge from the computer based training system. The developer learns independently. The system is responsible for learning dialogues, knowledge checks and the flow control of the learning process.

As described in Section 2, education in cyber security is mostly attack-based. The developer doesn't learn to create secure software in existing training systems. Therefore, a system with a combination of an attack and defense part was needed. Additionally, in training systems like OWASP Mutillidae II the developer can't be sure that he launched the "correct" attack in relation to the instructions. The developer needs a means to check, if tasks have been solved correctly.

4. Solution

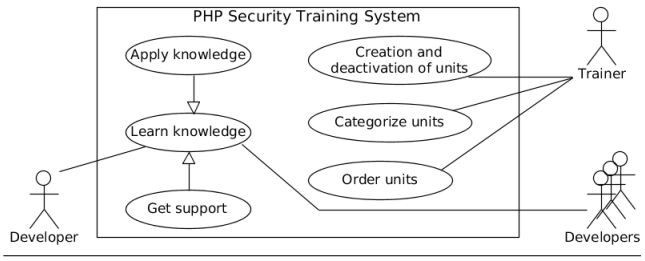

Figure 1. Use cases

The goal of PSeTS is to support developers to gain knowledge. Developers have to adopt knowledge, e.g. in multiple choice tests, and apply this knowledge in a practial task afterwards. The mainly used cases are shown in figure 1. The main usage is to benefit from PSeTS independently at home. But it can also be used by a teacher in a class room. Therefore, the teacher is able to modify units.

Different factors are important when acquiring knowledge through an computer based system as Mandl et al. describes [4]:

- Previous or expert knowledge means for developers:
  - Every day usage of web browsers and it's development tools.

[1] https://www.owasp.org/index.php/OWASP_PHP_Security_Training_Project

- Experience in HTML, JavaScript and PHP.
- Mastery of security fundamentals.
- Basic knowledge about HTTP.

- Objective: Willingness and ability to learn new things.
- Interest: Prioritization of learning and the willingness to spend time.

In order gain knowledge, different social-learning theories have to be applied:

- Bundle connected learning content [3, p. 159]
- Learning content is structured bottom up [2, p. 59]
- Superfluous learning content is removed [3, p. 159]

To bundle content, PSeTS uses categories. E.g. *Authentication and Session Management*, *Cross Site Scripting*, *Cryptographic Storage*, *Direct References* and *Injections*.

A unit is part of a category and consists of an attack and a defense part. After completely solving a unit the user is redirected to the next unit by PSeTS. The flow helps the developer to gain different perspectives, which supports the learning process [5, p. 151]. Knowledge gain is also supported by different problem statements [5, p. 151]. Therefore PSeTS is divided into different task types as shown in figure 2.

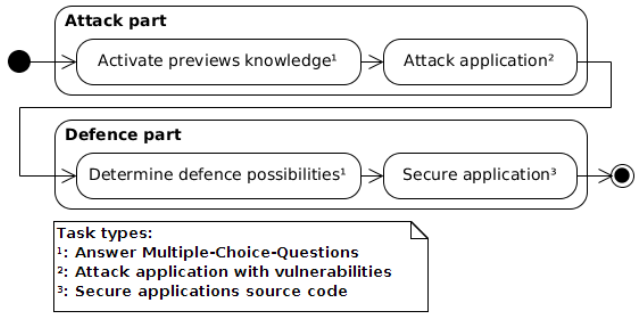

Figure 2. Task sequence

The task flow is shown in figure 3. A wrong answer of a developer has to be interpreted correctly. PSeTS has to ask the same question again, if a wrong answer is given [2, p. 56].

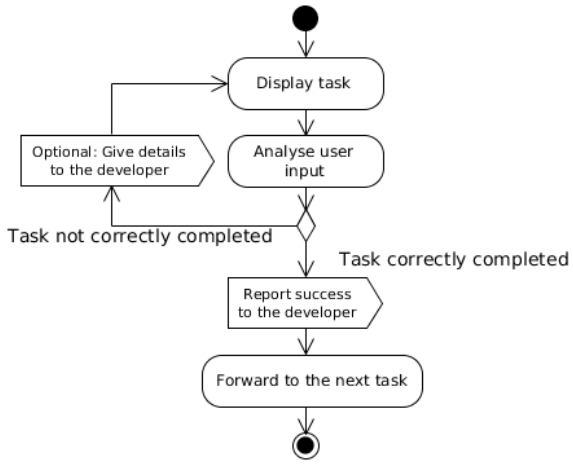

Figure 3. Task flow

5. Technical Implementation

PSeTS can be installed with vagrant[2] to create a virtualization image for VirtualBox. PSeTS is divided into a guidance part (accessible on http://guidesystem.local) and a unit part (accessible on http://unit
system.local). The guide system leads the developer through the system. The unit system provides a vulnerable application and validates answers.

5.1 Guide System

The guide system consists of the category menu, a unit menu, task information, help and the task itself. The category menu and the unit menu mark active and complete units and categories.

5.2 Unit System

An example for an attack is *directory traversal* in which the developer has to include a file from the filesystem which will be executed as part of the request. The vulnerable source code is `require_once ($_REQUEST['language']);`. In the defense part, the developer has to secure the source code. The unit system implements static and dynamic checks of user's source code. In the static check, the developer's source code is first cleaned by removing all whitespaces. This normalized source code is beeing compared with a prepared solution. If it matches, the task is marked as complete and the user gets notified.

The safeguarding part of the unit *directory traversal* is using the dynamic check method and validates the user's code by executing it with different parameters as shown in figure 4 and 5.

As demonstrated by the flow diagram in 4, the unit additionally validates that the business logic is working. PSeTS tries to include a hidden file, which is unknown by the developer. Individual feedback for unsafe solutions is provided. The specific messages along with an example input source code are listed in table 1 on the facing page.

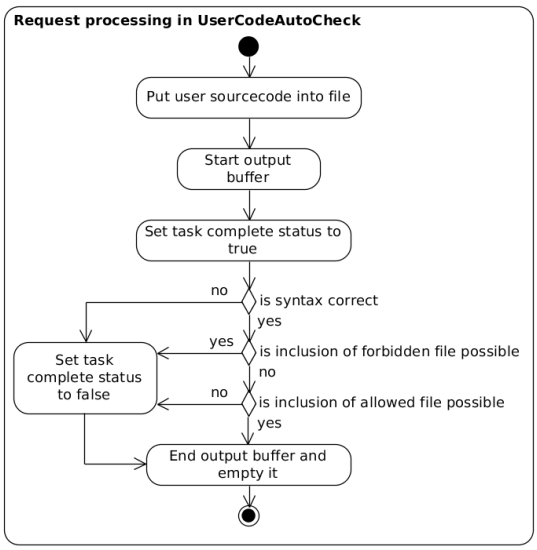

Figure 4. Request processing

6. Evaluation

The evaluation took place during the interdisciplinary week at the University of Applied Sciences Kiel. A heterogeneous group of 12 developers with different skills in PHP and web security.

[2] Mostly used for creating and configuring virtual environments.

Input Source Code	PSeTS Feedback
`// No safeguard in the source code`	File inclusion is still possible.

`if($_GET['language'] == "stock"){`
`  throw new UnsafeFileInclusionException("stock.php is forbidden");`
`}`
	File inclusion is still possible.

`if($_GET['language'] != "en"){`
`  throw new UnsafeFileInclusionException("Language is not en");`
`}`
	Inclusion of file *de.php* is not possible but should be possible.

`$allowedIncludeFiles = array("de","en");`
`if(!in_array($_GET['language'], $allowedIncludeFiles)){`
`  throw new UnsafeFileInclusionException("Unexpected language");`
`}`
	Question successfully answered, you get forwarded to the next task.

Table 1. Feedback from the system

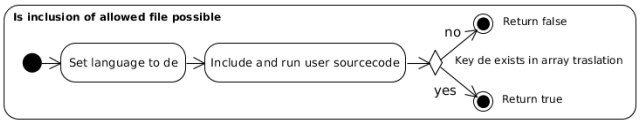

Figure 5. Inclusion check

6.1 Results

75% of the volunteers worked more than 10 hours per week as a software developer. To test the skill improvment by PSeTS, volunteers has to proof there knowledge before and after they use PSeTS. By the end of the evaluation only 9 developers were left. The results are shown in table 2.

Category	Subcategory	Correct Answer	
		Pre	Post
Input-Handling	Function `addslashes`	6 (50%)	7 (78%)
	Function `htmlspecialchars`	12 (100%)	9 (100%)
	Function `mysql_real_escape_string`	2 (17%)	7 (78%)
Output-Handling	Function `addslashes`	12 (100%)	9 (100%)
	Function `htmlspecialchars`	3 (25%)	7 (78%)
	Function `mysql_real_escape_string`	12 (100%)	9 (100%)
Hashfunction		1 (8%)	8 (89%)

Table 2. Summary of the knowledge survey

In the post survey, questions for the categories *general usage*, *user interface* and *navigation* amongst others were ask. The general usage is rated with an arithmetic mean of 2.0, the user interface with 1.74 and the navigation with 2.28.

6.2 Discussion

Only 8% in the pre survey were able to name a secure hash function. 89% name a secure hash function after using PSeTS.

The general usage of PSeTS is rated with 2.00 in the arithmetic mean. This shows that it is fun to solve the tasks and that PSeTS supports the gain of knowledge in the area of software security. The user interface has a rate of 1.7, which shows that PSeTS is easy to use. The navigation is constructed unstructured by viewing at the result of 2.56 in the arithmetic mean. It has been enhanced afterwards by flattening it from four to three hierarchical levels.

7. Summary and Future Work

Because a developer has the willingness to learn new things, he will not trick PSeTS to finish a task. This could be done in the defense part in which unlimited code "injection" and execution is possible. Therefore PSeTS is a good addition for a security training but not usable in a competition.

To establish a community, the following tasks has to be done: Implementation of internationalization to enhance the coverage of PSeTS. Existing defense tasks like safeguarding of SQL Injections should be used with dynamic checks. Implementation of unit tests to simplify the introducing phase of PSeTS developers. Other programming languages like Java should also be covered by PSeTS.

On the other hand, PSeTS could be integrated in a project like OWASP WebGoat which has a community already.

References

[1] Usage of server-side programming languages for websites. W3Techs, Retrieved December 2014. URL `http://w3techs.com/technologies/overview/programming_language/all`.

[2] Michael Kerres. *Multimediale und telemediale Lernumgebungen: Konzeption und Entwicklung*. Oldenbourg Verlag, 2001.

[3] Michael Kerres. *Mediendidaktik: Konzeption und Entwicklung mediengestützter Lernangebote*. Oldenbourg Verlag, 2012.

[4] Heinz Mandl, Wolfgang Schnotz, Sigmar-Olaf Tergan, and Steffen-Peter Ballstaedt. *Texte verstehen, Texte gestalten*. Urban und Schwarzenberg, 1981.

[5] Gerald A Straka and Gerd Macke. *Lern-lehr-theoretische Didaktik*, volume 3. Waxmann Verlag, 2002.