A HackIt Framework for Security Education in Computer Science
Raising Awareness in Secondary School and at University with a Challenge-based Learning Environment

Florian Kerber, Jan Holz, Hendrik Thüs and Ulrik Schroeder
Learning Technologies Research Group, RWTH Aachen University, Ahornstr. 55, 52074 Aachen, Germany
florian.kerber@rwth-aachen.de, {holz, thues, schroeder}@informatik.rwth-aachen.de

Keywords: Computer Science Education, Challenge, Game-based Learning, Framework, Security, Higher Education.

Abstract: HackIts are short computer security challenges which are often Web-based. Their purpose is to raise awareness for common security issues by showing different intrusion possibilities in today's computer security. We present a framework that allows security education in a safe, modular, and motivating way, with the possibility of flexible and low-cost integration into existing curricula. By solving challenges within this learning environment, the leaner gets confronted with IT security problems and learns how to prevent them.

1 INTRODUCTION

Today's society relies more and more on communication and storing data online, which results in security aspects becoming more and more important. To increase the focus on the growing importance of security aspects during Internet usage and software development, the HackIt!-Framework that will be introduced in this paper, puts the learners in the attacker's perspective to raise awareness for this topic.

In a game-based learning environment, the students learn how to identify and counter common security vulnerabilities. Each challenge (called a HackIt) is one level in the whole set of challenges of a course and addresses one or more security issues. They have to be discovered and exploited by the learners to solve the challenge and to advance to the next one.

The HackIt!-Framework is suitable for and aiming at secondary school students and university students in their early semesters. Both of them can use the framework during workshops, which are offered by our research group (Bergner et al., 2012) (Apel et al., 2012), or as exercises during or after lectures. After this, the learners should have a keen sense of security problems in today's Web software.

2 DESIGN DECISIONS

When making the first design decisions for the HackIt!-Framework, we had several aspects in mind that required consideration. Some of these are the flexible intergration into existing curricula, the importance of fun and motivation while learning and the practical relevance und application of security principles. In the following we will describe these considerations.

2.1 Flexible Curricular Integration

Safe software and cautious usage of current electronic media are vital for a risk-aware participation in modern life. This is especially important for computer science students, who will be future developers of those systems. So, integrating security aspects into computer science education is an important, but also complicated task. As Yang pointed out, most educators lack the knowledge needed for integrating security aspects into existing courses (Yang, 2001).

Nevertheless, we want to provide the possibility to integrate these important aspects flexibly into appropriate courses, which can stem from various different areas of computer science (Border and Holden, 2003). Due to limited time and resources in designing courses, we decided for a modular approach, which is not directly linked to a specific lecture or workshop (Edge and Stamey, 2010) as further described in section 2.3.

2.2 Aspects of Games in CS Education

Within the field of computer science education, the interactive learning style of game-based learning mostly invites to use this approach for teaching algorithms.

Shabanah and Chen list the following benefits of using serious computer games for algorithm learning: Computer games are popular, interactive and competitive and they utilize entertainment and simplify assessment (Shabanah and Chen, 2009). These characteristics are easily applicable to other areas of computer science education as well. Therefore, we will use a competitive game-based approach to motivate learners while playing the different challenges of the HackIt!-Framework.

2.3 Target Groups

The HackIt!-Framework provides a platform for a plethora of different security challenges, as described in section 3.1. Thus, it is applicable for different target groups. The basic content of the framework is intended for students that visit the InfoSphere - the students' lab for Computer Science at RWTH Aachen University (Bergner et al., 2012). These school students are mostly not yet familiar with the concepts of computer and information security. This framework helps the students to get informed about the responsibility to sanitize their source code. By confronting them with common mistakes and risks that arise thereof, they will be sensitized to avoid these mistakes in their future implementations.

Besides the usage for school students, this framework provides ideal conditions for higher education. Computer science students, e.g. participants of a Web-Technologies lecture, get the possibility to solve and maybe to develop challenges as part of their learning process. Especially the design and implementation of challenges support the learners in creating safer software (Pothamsetty, 2005). Depending on their prior knowledge, the framework allows to flexibly build individual task-sets of different difficulties.

3 FEATURES OF THE HackIt!-FRAMEWORK

The HackIt!-Framework will provide its users a full-fledged modular system for creating, modifying, and supervising HackIt!-based security challenges. It ships with an extensive administration panel, allowing the lecturer to modularly create a customized set of challenges via drag and drop (see Figures 1). Additionally, users will be able to request hints for the challenges, which, depending on the administrative settings, will cause scoring penalties or not. Optionally, the administrator can enable a high score for this challenge set, allowing users to view the progress

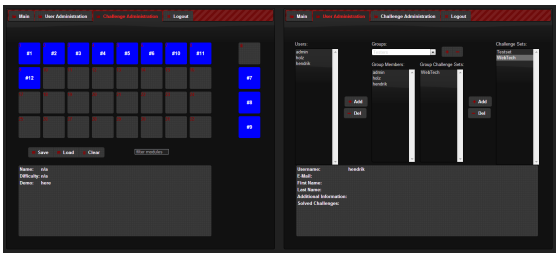

Figure 1: The tabs for challenge and user administration. Learners can be grouped together and challenges can be assigned to them.

of their classmates, which encourages competitive gameplay.

3.1 Challenge Types

All challenges are categorized by their type, providing a pleasant overview and the possibility to easily create mixed challenge sets.

- **Basic.** These challenges contain simple plain JavaScript challenges and are mostly used to warm up and give the user a basic understanding of JavaScript password checks.
- **Realistic.** Realistic challenges confront the user with real-world problems of today's software. They include but are not limited to SQL injection and cross-side scripting (XSS) due to missing input sanitization, as well as htaccess security and weak or commonly-used passwords. Some missions will also focus on the differentiation between legit and phishing mail, amongst others preventing the user to fall prey to fraud attacks.
- **Review.** During these challenges, the user gets confronted with the source code of a particular security feature and is responsible for finding the vulnerability or problem in it. These challenges might get extended to allow the learner to repair them.
- **Advanced.** These kinds of challenges require ssh key breaking, reverse engineering of self-modifying executables, custom hash breaking, and/or deep domain knowledge and should not be used for common school exercises, or at least be marked as optional. There are currently only a few challenges of this type with the purpose to challenge participants that already solved everything else.

These categories cover most of the top ten security issues, published by the OWASP Top Ten Project[1].

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

After the user sucessfully solved a challenge, which in most cases corresponds to obtaining the hidden password or gaining access to a protected area, the user is informed about this success and brought back to the challenge overview. In this overview, the just-solved challenge changes its color and thus eases the overview over the overall progress (see Figure 2).

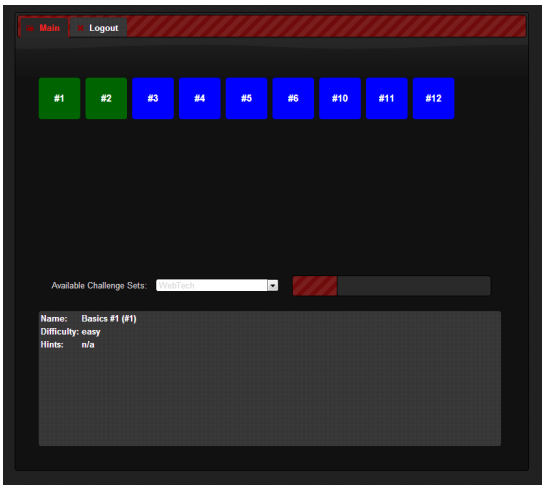

Figure 2: The overall progress of a certain learner.

To overcome non-self-explainatory instructions or to motivate the learner to work on a possibly less engaging exploit or vulnerability, the HackIt!-Framework wraps the challenges in a story that follows the user throughout the set and supplys him with information regarding the task.

To prevent frustration, the HackIt!-Framework will offer its users the ability to request hints by clicking on a button to point them in the right direction. These clues vary in their detailedness by starting from a very basic hint, like which subject has to be investigated, up to a detailed problem description and optional steps required to solve the challenge. In order to decrease the overuse of this optionally enabled feature, its usage can optionally cause certain penalties to the user. These penalties can range from a simple subtraction of points, the use of a modifying factor to the amounts of points the challenge normally would yield, up to the marking of the user's challenge as "solved with hints".

Since an important point of human psychology is motivation, the HackIt!-Framework allows users to compete against each other by providing a public scoreboard, on which they are able to see each other's progress. The position on this scoreboard is determined by the amount of challenges solved, their difficulty, and optionally the time required to solve it.

3.2 Setup and Usage

To ensure the security of the host system as well as an easy deployment of the HackIt!-Framework, it will be distributed as an almost ready-to-use virtual sandbox image for the virtualization software VirtualBox.

The framework itself can and should be hosted by the teacher. It provides a Web server which is capable of handling all the requests. There is no need for the students to install anything on their PCs. The challenges can (mostly) be solved by only using a web browser. Since this is a centralized client-server system, the administrator (the teacher) is able to assemble to challenges and to view the results.

4 RELATED LEARNING ENVIRONMENTS

Aside the HackIt!-Framework, there is the project HackThisSite[2], which also provides HackIt challenges to its users. This website offers tasks in different categories: basic, realistic, application, programming, phonephreaking, extbasic, JavaScript, steganography and IRC missions. Only the Basic, JavaScript, and Realistic challenges are comparable to the HackIt!-Framework. The other challenge types specialize on finding hidden messages inside images (steganography), abusing known Internet Relay Chat (IRC) vulnerabilities, writing programs to solve certain tasks within a short time limit which require decent times for humans (Programming) or reverse engineering programs to make them reveal their secret (Application). As these challenge types do not sensibilize the user to (Web) security, but rather in more advanced or more specialized topics, these kinds of challenges are currently not included in the HackIt!-Framework. But due to the fact that the HackIt!-Framework is completely modularized, these types of challenges could be easily created and added later on, as there is no restriction in the kind of challenge that the framework supports.

A different example is the website Happy-Security[3] which also offers various kinds of challenges: exploits, Flash, IRC, Java, JavaScript, cryptography, logic, programming, reverse engineering, steganography and PHP. The challenges of the different categories vary in their difficulty and can be created by the participants themselves. Most of the categories are outside of the scope of the HackIt!-Framework but especially the basic challenges, like

[2]http://www.hackthissite.org
[3]http://www.happy-security.de

the in the JavaScript or the exploit category, are compareble to those of the proposed framework. Still Happy-Security is difficult to use for educational purposes as it does not allow customization for courses of different difficulties or feedback on the learning progress for the tutor.

A possible drawback of the HackIt!-Framework in comparision to these websites is the community. In their forums, users are able to ask questions regarding the challenges, and others try to point them in the right direction without spoiling too much information. This kind of community feedback is replaced with the upcoming hint system in our framework.

The benefit of the HackIt!-Framework is its modularized core, allowing easily exchangeable challenge sets, user management as well as an easy inclusion of own and new challenges. This allows a quick growth without much effort, as most functionality is provided by the framework. Besides, the related systems do not offer the possibility to view or export the results of the (un-)solved challenges of the students.

5 CONCLUSIONS

This paper presents the development and features of the HackIt!-Framework, a Web-based framework for modular challenge set creation. It provides a solid base for further computer security learning content and enables to teach the security aspects of current website security as well as today's Internet risks.

The focus on modularity should ensure a flexible and persistence use, whereas the strong self-security measures allow a save deployment in educational settings. The possibility to adopt its challenges to the target audience's knowledge level make it a suitable choice for weekly exercises or smaller competitions.

5.1 Future Work

There are many ways to extend the HackIt!-Framework and enrich its features. The implemented challenges for this system do not cover the complete set of the main security issues. The next steps are to implement several ideas for challenges to have a good basis that covers most of nowadays security problems and to keep the framework and its content always interesting and feature rich.

The HackIt!-Framework is work in progress. Some described features are not deployed or fully tested at the current time. These are namely the scoreboard, the hint system, the story, the unique passwords, the flagging challenges as optional or required, and the order enforcement of challenges. The deployment and the testing of these features will be done in the near future to have the system fully functioning for the upcoming evaluation.
The authors plan to use the HackIt!-Framework as a method to teach students of a Web Technologies lecture how easy it can be to exploit self-implemented Web applications if the source code is not sanitized well. In this evaluation, not only the interface design and the ease of use will be evaluated but also if the students will be aware of the risks of implementing Web application when it comes to security issues. Last, but not least, it should be determined if the students like to play with those challenges and if it is easier for them to learn about possible counter measurements by solving quests instead of learning about those threads in a theoretical way.

REFERENCES

Apel, R., Berg, T., Bergner, N., Chatti, M. A., Holz, J., Leicht-Scholten, C., and Schroeder, U. (2012). Ein vierstufiges frderkonzept fr die studieneingangsphase in der informatik. In *Proceedings of HDI 2012*, Hamburg. Universittsverlag Hamburg.

Bergner, N., Holz, J., and Schroeder, U. (2012). Infosphere: An extracurricular learning environment for computer science. In *Proceedings of 7th Workshop in Primary and Secondary Computing Education (WiPSCE 2012)*, Hamburg. ACM.

Border, C. and Holden, E. (2003). Security education within the IT curriculum. In *Proceedings of the 4th conference on Information technology curriculum*, CITC4 '03, pages 256–264, New York, NY, USA. ACM.

Edge, C. and Stamey, J. (2010). Security education on a budget: getting the most "bang for the buck" with limited time and resources. In *2010 Information Security Curriculum Development Conference*, InfoSecCD '10, pages 29–35, New York, NY, USA. ACM.

Pothamsetty, V. (2005). Where security education is lacking. In *Proceedings of the 2nd annual conference on Information security curriculum development*, InfoSecCD '05, pages 54–58, New York, NY, USA. ACM.

Shabanah, S. and Chen, J. (2009). Simplifying algorithm learning using serious games. In *Proceedings of the 14th Western Canadian Conference on Computing Education (WCCCE '09)*. ACM.

Yang, T. A. (2001). Computer security and impact on computer science education. In *Proceedings of the sixth annual CCSC northeastern conference on The journal of computing in small colleges*, CCSC '01, pages 233–246, USA. Consortium for Computing Sciences in Colleges.