**C-Sploit: A Serious Game for Identifying Software Vulnerabilities in C Computer Code**

Nathaniel W. Flack, Air Force Institute of Technology, Wright Patterson AFB, OH, 45433

Abstract

The world is becoming more defined by software. Methods of transportation, the places we live and work and even our social interaction are more and more defined by software. This fact necessitates that program managers, project leads, and software developers be able to identify vulnerabilities within software. This paper proposes a serious game that will help beginner and intermediate programmers practice the task of reviewing computer code in the C programming language for security vulnerabilities. Additionally, players will become more familiar with the types of exploits that may be used against software and how to pair vulnerabilities with potential exploits. Finally, I will show how this game could be used in the context of a secure software design classroom to effectively practice code review and learn the potential exploits that might be used to attack their code.

Introduction

In 2011, Marc Andreessen a venture capitalist and technology investor, said, “Software is eating the world.”[1] He said this in light of the increasing success of many software companies. In recent years, the world of software has moved to the Internet of Things (IoT) which boils down to the addition of software and Internet connectivity to things we use every day, to include refrigerators, vehicles, thermostats, lightbulbs, and just about everything else [2]. This software boom, along with the accessibility of IoT devices through the Internet, both for users and attackers, carries significant cybersecurity risks making the development of secure software more important than ever.

These changes will require more attention to be placed on the education of software designers, developers, and testers as well as project managers in awareness and application of secure software design techniques. In response to this need, this paper proposes a serious game that will build familiarization and deeper awareness of the need and assist in designing and developing secure software.

Serious games are an emerging teaching tool that is growing in popularity. Their effectiveness outside of the classroom is evident in studies from the United Kingdome [3] and the United States [4] and research continues to refine the models for creating and using serious games for educational purposes. In an educational setting, serious games generate environments for collaborative learning where knowledge is gained through discussion and negotiation between individuals and groups [5].

While awareness of the need for secure software has been growing in recent years, new avenues of engagement, such as this game, could be a key factor in continuing and enhancing the conversation. Games could also help to engage a new generation of software designers, developers, testers, and users to think about software security.

Literature Review

Serious games have shown to be effective in many areas of study and a wide-range of benefits. A 2016 review of the literature shows 143 papers with high quality evidence concerning the positive outcome of games. The authors write, “The most frequently occurring outcomes and impacts were knowledge acquisition/content understanding and affective and motivational outcome.”[11]

Although there are numerous serious game studies, fewer have been performed in a classroom setting due to multiple factors. One recent study shows that students who played serious games show a higher self-reported knowledge and an increase in “higher-quality” learning, although test scores showed no difference from traditional classroom instruction [6]. Another study analyzing the effect of games in English as a second language vocabulary learning showed that games promoting interactive learning led to greater learning over non-interactive games and traditional teaching methods [7].

It is important to note that several studies show that males outperform females in serious games studies [7][8]. This is an important aspect to keep in view as the computer programming related fields are already male-dominated. Analysis of this phenomenon is present in the literature since 1985 [9] [10].

Many serious games have been developed for the computer science education world, specifically for introductory computer programming classes. However, no mainstream serious games exist, nor are there any studies that examine the impacts on serious games for secure software design.

A serious card game created by Adam Shostack in 2014, called Elevation of Privilege, is an example of a game designed to teach software security. His game seeks to draw software developers into the practice of threat modeling using a card-based game that is easy to learn. His results show initial success in several contexts that drove discussion about and interest in threat modeling. The game was mostly used as a training tool within Microsoft [13].

Game Components

The game proposed in this paper uses coding examples in the C programming language and known vulnerabilities to enhance the player’s ability to identify secure and insecure computer code as well as match vulnerable software and with the correct exploit.

This serious game is a simple, points-based card game designed to help players review computer programming code, identify known vulnerabilities, and build awareness of existing exploits. The game is a card-based game called “C-Sploit” for 2+ players or teams. The game could also be used by a single individual to improve their skill in identifying vulnerabilities in code.

The game components consist of three different types of cards. First, code snippet cards show small samples of C code that may or may not contain known vulnerabilities. The backside of each code snippet card reveals if the code is secure or not and identify the possible exploits that could be used against the example code. Code snippet cards will need to be large because they contain a lot of information and need to be viewable to all players. As the game matures, it would be ideal to have two different decks of code snippet cards, one for beginners and an expansion deck (of code snippet cards) for experienced software designers, developers, and testers.

The code snippet cards must be developed using both secure and insecure code, which requires relevant examples of both. The book *24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them* by Howard, LeBlanc, and Viega’s [12] is primarily used for the exploit cards and definitions and code snippets. The code on the code snippet cards will need to be simple so that players can easily identify what the code is meant to do and decide if it is secure or not. Therefore, tweaks may be made to the code samples provided in Howard et al, and other coding tutorials and sources may be used.

Details of the security of the code snippet will be detailed on the back of the card. This will include a short description of why the code is vulnerable and a list of possible exploits that could be used to successfully attack the code. This information will determine how each round is scored depending on each card played.

The first version of the game will have secure and insecure code snippet cards in the following categories:

1. Allocating memory with user input
2. Printing user input to the screen or log
3. Performing math functions with user input
4. Error handling and catching exceptions
5. Requesting admin credentials
6. Executing user input with system calls
7. Building SQL queries with user input
8. Creating and using temporary files
9. Handling usernames and passwords
10. Public Key Infrastructure
11. Encryption

﻿void main (){

[“path” provided by user]

char buf[20];   
char prefix[] = "http://";

strcpy(buf, prefix);   
strncat(buf, path, sizeof(buf));

}

Figure 1. Example Code Snippet Card

This code is vulnerable to a buffer overrun attack on “buf”

The problem here is that strncat() has a poorly designed interface. The function wants the number of characters of available buffer, or space left, not the total size of the destination buffer

Figure 2. Backside of Code Snippet Card in Figure 1

The second kind of cards are exploit cards which have known exploits or attacks derived from multiple sources. Each player will have four of these cards in their hands. The third type of card, is a version of the exploit card, but instead of a specific exploit the cards will say either “The code is secure” or “the code is vulnerable.” Each player will have one of each of these cards in their hand in addition to the four other exploit cards. All exploit cards will have the same back to simplify design, but also to mask a players move so that other players or teams do not know what has been played thus far.

Here are several exploit cards that will be created for use in the first version of the game. Each has one or more of the code snippets that it can potentially exploit.

1. Buffer Overrun Attack
2. Format String Attack
3. Integer Overflow Attack
4. Capturing Leaked Information
5. Elevating privilege
6. Elevating Privileges
7. Command Injection
8. SQL Injection
9. Stealing Login Credentials
10. Intercepting Encryption Key
11. Breaking Encryption Algorithm

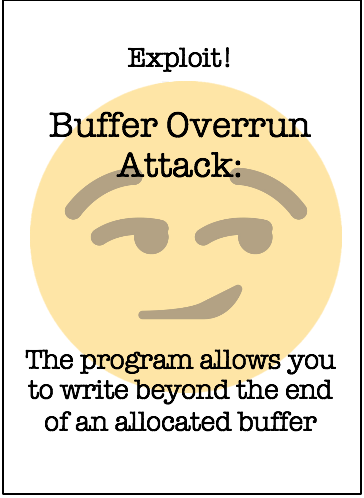


Figure 3. Exploit Card Sample (back and front)



Figure 2. Secure/Vulnerable Exploit Cards

Game Rules

The game’s theme centers around a software company writing code for military weapons systems that has run into tough competition with other companies trying gain more of the market share. There are also growing concerns that foreign nation states are increasing the rate and complexity of their attacks, specifically against software. You have just been selected to head an elite team of code reviewers analyzing code excerpts from your company along with that of your competitors. You and your team must determine if the code is secure or not and if it is vulnerable, try to exploit it with the tools at hand. You are the last line of defense; your company and your country need you.

The objective of the game is to gain as many points as possible after ten rounds of play. Players will either play the game as individuals or teams. Teams can be as big or small as needed depending on the size of the group. Teams should be spread out, if possible, to allow for open team discussion without giving away their game decisions to other teams. The game needs at least 2 individuals or teams to be played, but can support many players if enough cards are present. If in a classroom setting, the code snippet cards could be projected so that all players can see the code clearly. Then players or teams could play the exploit cards in a table in the middle or front of the room. To simplify the game rules, teams will be referred to as a “player.”

One code snippet card will be played for each round of the game. Every player will examine the code and play one exploit card (face down in the game space). This card could be an actual exploit to attack the code or one of the secure/vulnerable cards. Once all players have played a card for the round each player reveals their exploit. To enhance the effectiveness of the game to teach the learning objectives, each player should explain their play. Once the discussion is over, the code snippet card is flipped over revealing the answers. Points are awarded to players based on the following scale:

- Correctly identified code as secure (1 point)

- Correctly identified code as not secure (1 point)

- Played “the code is secure” card incorrectly (-2 points)

- Played “the code is vulnerable” card incorrectly (-1 point)

- Played a card containing an exploit that was incorrect (-2 points)

- Played a card containing an exploit that was correct (3 points)

After each round, all players take back their secure/vulnerable cards (if played) and all other exploit cards are put in the discard pile. All players draw two exploits cards and then discard as necessary (one or two exploit cards) to return to a six-card hand consisting of four exploits plus the pair of secure/vulnerable cards. The additional step of drawing and discarding exploit cards rewards players that can identify more versatile exploits that might work against multiple code snippets.

As part of the design multiple exploits could be successful against multiple code snippets and code snippets should either be secure or vulnerable to more than one exploit. This will allow for multiple exploits to score points in a single round.

The game ends after ten rounds have been completed. The player with the most points after ten rounds is the winner. If there is a tie, the player with the most successful exploits (sinister face) will declared the winner.

Methodology

There are three target audiences for this serious card game. The first, and primary audience, is students learning the concepts of secure programming and secure software design. Second, are code developers, who could benefit from the discussion of secure coding principles and potential exploits that could be used to attack their code. Third, those that frequently review code for quality assurance or security testing.

The game is built to address three learning objectives. First, to teach familiarity with exploits that could possibly be used against software. Second, to improve the ability of players to identify if a snippet of code is vulnerable or not. Third, to strengthen the player’s ability to pair a software vulnerability with the possible exploits.

Because every code snippet will have an explanation on the back of the card of why it is or is not secure along with a list of exploits that could be successful against the code this game will be most beneficial in an environment where discussion can happen at the end of each round. This will create interaction between the players/teams and should increase learning as the game is played.

It is expected that new vulnerabilities and exploits will be created and discovered after this game has been created, therefore players are encouraged to add exploit and code snippet cards to their decks as well as award points

There is randomness involved in the game as players can only keep four exploit cards in their hand. This means that some players may know how they would successfully exploit a vulnerable piece of code, but might not have the exploit available in their hand at that time. This is true to the real world as vulnerable code is not exploited every time.

The game also favors those who can accurately identify secure or insecure code as well as choosing exploits that may be more useful given their experience and knowledge of specific exploits.

There is a strong component of risk and reward in the game. The number of rounds is set at 10 to build excitement and encourage some risk toward the end of the game for those players that might be behind in the points total. The risk occurs because cards containing specific risk yield higher points, both positive and negative, for correct and incorrect plays respectively.

A possible experiment is to teach this game in a classroom setting for a course focused on software security. This could be very effective in an undergraduate program where students would most likely have taken a C or C++ programming class, but still be mostly unfamiliar with software security, vulnerabilities, and exploits. Before playing the game students could be survey concerning their current knowledge of software security, C programming, and exploits. Then, after learning and playing the game students could be asked about their experience and if the game contributed to their understanding. Additional post-game survey questions could focus on the fun factor in the game and if it elicited additional conversation about software security. If possible, this research should be compared to a control group of students in the same class that were not given the opportunity to play the game, but were taught the same concepts. A test or another assessment technique could be used to determine any difference in the quality of learning with the serious game.

Future Work

There are many areas where serious game design and development could add to the current education and awareness strategies. The abstraction needed by computer programmers seem to make them amiable to learning through serious games.

Further work on this card game could include the addition on new exploit and code snippet cards. New vulnerabilities and associated exploits are an unfortunate reality in today’s world. Therefore, blank cards should be included in each C-Sploit game so that players can make their own exploits, write their own code to be included in the game, and try to make the game as relevant to their context as possible. Also, the game could use additional elements to make it more fun and exciting. While the C programming language is used, it could be played using pseudo-code that could be more easily understood by a wider audience.

There are also further studies that could provide insight into the value of this game and similar games. Another possible experiment could be to play the game with students familiar with programming techniques, but not required to take a secure software design class. This experiment would test the interest of the students in secure software design both before and after the game in order to measure its effectiveness in growing interest in this important field.

Conclusion

Serious games have provided a unique and engaging tool for educators and practitioners to build awareness on important issues. C-Sploit is a simple card game seeks to raise awareness concerning secure software design and provide a fun context to discuss the complexities of computer code, secure design, exploits, and vulnerabilities.

Bibliography

[1] Andreessen, Marc. "Why Software Is Eating the World." *The Wall Street Journal*, August 20, 2011. Accessed October 3, 2018. https://www.wsj.com/articles/SB10001424053111903480904576512250915629460.

[2] Daley, John. 2017. “Insecure Software Is Eating the World: Promoting Cybersecurity in an Age of Ubiquitous Software-Embedded Systems.” *Stanford Technology Law Review* 19 (3): 533–46. http://search.ebscohost.com.afit.idm.oclc.org/login.aspx?direct=true&db=a9h&AN=123284244&site=eds-live.

[3] Dunwell, Ian, Sara de Freitas, Panagiotis Petridis, Maurice Hendrix, Sylvester Arnab, Petros Lameras, and Craig Stewart. 2014. A game-based learning approach to road safety: the code of everand. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 3389-3398. DOI: https://doi.org/10.1145/2556288.2557281

[4] Kato, Pamela M, Steve W Cole, Andrew S Bradlyn, and Brad H Pollock. 2008. “A Video Game Improves Behavioral Outcomes in Adolescents and Young Adults With Cancer: A Randomized Trial.” *Pediatrics* 122 (2): e305 LP-e317. http://pediatrics.aappublications.org/content/122/2/e305.abstract.

[5] Lameras, Petros, Sylvester Arnab, Ian Dunwell, Craig Stewart, Samantha Clarke, and Panagiotis Petridis. 2017. “Essential Features of Serious Games Design in Higher Education: Linking Learning Attributes to Game Mechanics.” *British Journal of Educational Technology* 48 (4): 972–94. doi:10.1111/bjet.12467.

[6] Bakhuys Roozeboom, Maartje, Gillian Visschedijk, and Esther Oprins. 2017. “The Effectiveness of Three Serious Games Measuring Generic Learning Features.” *British Journal of Educational Technology* 48 (1): 83–100. https://doi.org/10.1111/bjet.12342.

[7] Zhonggen, Yu. 2018. “Differences in Serious Game-Aided and Traditional English Vocabulary Acquisition.” *Computers & Education* 127 (December): 214–32. doi:10.1016/j.compedu.2018.07.014.

[8] Riemer, Valentin, and Claudia Schrader. 2015. “Learning with Quizzes, Simulations, and Adventures: Students’ Attitudes, Perceptions and Intentions to Learn with Different Types of Serious Games.” *Computers & Education* 88 (October): 160–68. doi:10.1016/j.compedu.2015.05.003.

[9] Kiesler, S., Sproull, L., & Eccles, J. S. (1985). Pool Halls, Chips, and War Games: Women in the Culture of Computing. *Psychology of Women Quarterly*, *9*(4), 451–462. https://doi.org/10.1111/j.1471-6402.1985.tb00895.x

[10] Lockheed, M.E. Sex Roles (1985) 13: 115. https://doi.org/10.1007/BF00287904

[11] Boyle, Elizabeth A., Thomas Hainey, Thomas M. Connolly, Grant Gray, Jeffrey Earp, Michela Ott, Theodore Lim, Manuel Ninaus, Claudia Ribeiro, and João Pereira. 2016. “An Update to the Systematic Literature Review of Empirical Evidence of the Impacts and Outcomes of Computer Games and Serious Games.” *Computers & Education* 94 (March): 178–92. doi:10.1016/j.compedu.2015.11.003.

[12] Howard, Michael, David LeBlanc, and John Viega. *24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them* (2005).

[13] ﻿Shostack, Adam. 2014. “Elevation of Privilege: Drawing Developers into Threat Modeling.” *USENIX Summit on Gaming, Games, and Gamification in Security Education*, 1–15.