Nathan Flack

Secure Software Design

Serious Game Idea Submission

I am going to develop a game that will help players become more familiar with C code, identify known vulnerabilities, and theoretically exploit code using existing vulnerabilities. My game idea is a card-based game called “C-Sploit” for 2+ players. The game could also be used by a single individual to improve their skill in identifying vulnerabilities in code. The game will have 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 identifies the possible exploits that could be used against the example code. The second kind of cards are exploit cards which have known exploits or attacks derived from multiple sources. Each player will have 4 of these cards in their hands. The third type of card are a type of 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 4 exploit cards.

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 front of the player). This card could be an actual exploit to attack the code snippet or state that the player thinks the code is vulnerable or not by playing the appropriate card. Once all cards are played for the round each player reveals their exploit. For greatest effect, each player should explain why they played what they did. Once this is completed, 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 the cards are cleared to the discard pile. All players take back their secure/vulnerable cards and draw an additional two exploits cards. Each player must then discard one or two exploit cards to return to a six-card hand (4 exploits plus the secure/vulnerable pair 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.

For this scope of this class, I don’t believe I will be able to create all the code snippet and exploit cards, but I hope to make some sample cards to better explain the game play and types of cards. This could also be a helpful study tool for students preparing for exams in the class.

Great initial design.

So if I understand correctly, your lesson objective is to provide experience in discovering vulnerabilities in software and your target audience is experienced programmers? I like the design as it stands; however, I’m curious if there’s a way to create graduated cards so that players that are not familiar with these vulnerabilities can learn about them through the game. Thoughts?

How many code snippets and exploit cards are you expecting to make? With a design this far along already, I would think you’d be able to create quite a few during the time remaining. Thoughts?

My only concern is whether the game is fun. Whether fun or not, it sounds like it will work well. However, is there any way that you can increase the fun factor?

What sort of a literature review do you expect to need to do? The Elevation of Privilege paper will likely be included. I’d also like to see something about serious games and using them in the classroom. Lastly, you’ll need some reference to the code snippets and exploits. While the Howard book provides an overview, you may want to consider also looking at examples on the web. Consider phrack.org and blackhat.org as resources. For what you’re doing, they may be better sources than scholarly papers. Looks great. I’m here if you have any questions or want to brainstorm.

Response:

Yes, build experience in identifying the differences vulnerabilities within software that can be identified in the code itself. Through repetition players will be able to quickly identify code and know if it is reasonably secure or contains obvious errors.

Target Audience: Students in secure programming and software design course, secure s/w programmers, individuals who frequently review code such those in quality control and management positions.

I would like to design the game with a beginner card deck and then add an expert card deck that would have the same errors but be harder to spot and/or exploit. This will create an entry level games for players that don’t have a lot of programming experience (or aren’t as familiar with C code) as well as experienced programmers.

This game could also be very helpful as a review game for students before an secure s/w design class

Literature Review:

Elevation of Privilege.

CyberWar 2025.

**EMERGO: A Methodology and Toolkit**

This paper analyzed a specific framework for creating a scenario-based game for use in the classroom. While the authors were creating and evaluating serious games for the classroom setting, which is the same for my game. It focused solely on scenario games. However, the article provided a list of questions that any game creator could ask in thinking through a new game design as well as evaluating their game following trial runs.

Nadolski, Rob J., Hans G. K. Hummel, Henk J. van den Brink, Ruud E. Hoefakker, Aad Slootmaker, Hub J. Kurvers, and Jeroen Storm. 2008. “EMERGO: A Methodology and Toolkit for Developing Serious Games in Higher Education.” *Simulation & Gaming* 39 (3): 338–52. doi:10.1177/1046878108319278.

2.1.5. Anxiety and effort caused by serious games

A proper amount of, rather than excessive, player anxiety and mental effort could strengthen the effectiveness of vocabulary learning. The restricted serious game “Augmented Reality” provided students with effective learning because the serious gaming led to slight player anxiety and mental effort, while the free serious game system could merely produce flow experience for students ([Hsu, 2017](https://www-sciencedirect-com.afit.idm.oclc.org/science/article/pii/S0360131518301970" \l "bib15)). Children who suffered from autism spectrum disorder could improve their vocabulary learning and keep vocabulary in mind longer because of the slight player anxiety and mental effort ([Khowaja & Salim, 2018](https://www-sciencedirect-com.afit.idm.oclc.org/science/article/pii/S0360131518301970" \l "bib17)).

**7.6. SPOTTING THE SIN DURING CODE REVIEW**

C/C++ developers need to pay the most attention to integer overflows. Now that many developers are better about checking sizes when directly manipulating memory, the next line of attack is on the math you use to check what you’re doing. C# and Java are next. You may not have the issue of direct memory manipulation, but the language lets you make nearly as many mistakes as C/C++ allows.

One comment that applies to all languages is to check input before you manipulate it! A very serious problem in Microsoft’s IIS 4.0 and 5.0 web server happened because the programmer added 1 and then checked for an overly large size afterward—with the types he was using, 64K −1+1equals zero! There is a link to the bulletin in the section “Other Resources” in this chapter.