George Litvinov
gl2517

Topics in Software Engineering
Final Report

Practical Security Framework

Synopsis

I have implemented CTF-like course material for the fundamental security course at Columbia – *COMS W4181 Security I*. This project aims to introduce a practical component to the course in a gamified manner in order to improve student practical skill.

The project is composed of two parts – the web portal that was built on top of CTFd[1] (a basic open-source web portal for Jeopardy-style hacking competitions) and a set of challenges that were developed specifically for the Security I course.

Research Questions

Is it possible to develop and adapt a Jeopardy-style CTF structure into a classroom setting?
Yes the creation of such a framework is possible and was done for this project.

What are core requirements for such a framework to be adapted into a classroom setting?
After multiple discussions with Security faculty at Columbia, the adaptation of such a framework into a classroom would be contingent on multiple factors

- Ease of setup: The framework has to be easy to set up and to deploy. This was achieved by using docker containerization for both the framework as well as the challenges. Unfortunately, due to time limitations the challenge to web portal pipeline automation was not implemented, however a solution such as ctfcli[2] would allow for such automation to occur.
- Modularity: The web portal should allow for future improvements and challenges should be fairly easy to change and their parts should be reusable. As an open source framework, CTFd allowed for the development of this project, and it can be continued to be built on. For the challenges, container options in each of the challenges can be easily adopted in future development.
- Updatability: Both the web portal and challenges should be easy to update both in the short-run (mistake in code) and in the long-run (constantly changing technology front). Since all of the source code is given, the update and redeployment cycle is down to minutes. In terms of the long-run, the framework is simply a starting ground which can be built upon – although some of the challenges may have to change over time, the web portal can accommodate all needs in the long-run.

[1] CTFd on Github – https://github.com/CTFd/CTFd
[2] ctfcli on GitHub – https://github.com/CTFd/ctfcli

George Litvinov
gl2517

- Student-factor: The students should be able to easily set up everything they need for the course. Specifically for this reason the *basic* category of challenges was created that would allow students to be set on a level startinground.
- Scalability: The security course may grow to 100-200 students in the future, the framework should be able to handle them. This project was developed with that in mind: Google Cloud instances can be increased and decreased in terms of performance based on needs, furthermore docker containers can be given more or less computational power.

Deliverables
All of the deliverables can be seen on the Github Repo of this project. All of the installation and usage instructions can be seen there as well. The framework was designed to be intuitive and easy to use. I hope you get to try out some of the challenges too!

Please see the project demo here.

Self-Evaluation
Projects like this are very open-ended and can always be improved upon. However, I have definitely achieved my goal of developing a framework and building a strong foundation which can be built upon if there is a wish to use such a framework in a security course. I had to reduce the scope of the project in terms of the amount of challenges that were developed. The main difficulties in the project stemmed from containerization of hacking challenges. For example, for memory security challenges, the attacker should be able to overflow a buffer and break out of the program, but not the container itself. Only after a lot of experimentation was I able to set up the containers in such a way, that they remained secure where required and had a controlled behavior upon induced intended attacks.

Furthermore, it definitely took some creativity to develop the challenges, as well as niche skills. Writing purposefully insecure programs felt very weird to me personally, but I think made me a better programmer in the bigger picture. There was a lot of trial and error involved in figuring out how to make challenges work.

Overall, this was an immensely fun and thought-provoking research project that I am glad I was able to work on.