Simon Owens
Senior Project Planning 2018

Problem – Due September 14th

Universities desire to teach software security because of the industry demand for secure coding and Security Engineers. The best way to prepare students is with hands on experience seeing, exploiting, and patching vulnerabilities. Setting up a practice area for students with multiple computers is expensive and requires management. Students will also frequently crash their target computers which requires constant troubleshooting and resetting. If you allow them full permission to the infrastructure to troubleshoot their own problems, they could do nefarious things or even break the infrastructure. Students could host their own virtual machine, but this takes time away from class, requires computing power, and doesn't give students unique answers to submit. Even if all of these efforts were planned, supported, and managed there are not any solutions that translate student exercises to grades for professors. Professors could take the time to create tons of exercises and vulnerable virtual machines but there are already hundreds of great resources available. This is where my application comes in – the Security Lab Manager. It takes these vulnerable exercises others have already made, adds a unique hash for every exercises, per student, and manages them so students can attack, destroy, and reset their machines. Professors have a nice interface to view competition of student exercises and be notified if any students cheat. Hosting this application takes minimal resources and can scale easily to the class size.

How do we build a solution that allows students to?

- Start working immediately with relatively no client-side setup
- Launch and reset exercises
- Submit unique answers to each problem

And allows professors to?

- Install/setup easily
- View student submissions
- Use visualization infrastructure that uses ½ GB of ram per student

My Security Lab Manager is a collection of Docker services working together to virtualize this environment: a proxy, web front-end, back-end, and a database. An administrator can download the project and install the application with one click on either Windows or Centos7 running Docker - the installer only has enter the master password for the application. The administrator can then visit the IP of the host computer via HTTPS to login and start creating users. Once student's login, they will be able to view various exercises and start them. Starting an exercise will launch a light-weight Docker container. This container will have a unique hash in the root directory based on: the teacher's password, student's name, and exercise name. Students can then begin attacking the virtual machine to uncover the hash. If students crash the virtual machine, they can simply restart it with one click. Once students complete the exercise, they can submit their unique hash to the application. Teachers can then view student's progress and be alerted if any hashes submitted are the same. If students wish to add any new exercises, they just have to enter: the exercise's name, where it should be grouped, and the Docker image name.

How does my solution scale, stay up to date, and remain secure?

- Using Dock as the visualization image allows users to easily add new security exercises. I don't need to spend the time making new exercises since other professionals already make things like WebGoat, Bricks, and Damn Vulnerable Web Application.

- Using an Nginx proxy and Docker containers allows the administrator to scale the application's performance easily. This application could support anywhere from 5 to hundreds of users via load balancing and redundancy.

- The continuous integration Jenkins build will detect if a base container breaks functionality upon any update. A failed build on the development branch will not push to production so stable releases can always be used. Before any code can be added to production, all tests must pass, and there must not be any Sonarqube vulnerabilities, code smells, or bugs. Snyk and Dependabot do scans against the project for common vulnerabilities and my dependencies.

- All requests to web application front-end come through Nginx via HTTPS so attackers cannot snoop on traffic or execute remote vulnerabilities easily since Nginx has a great security program.

- A vulnerability assessment will be done against the system to ensure none of the OWASP top 10 exist in the web application

What are some of the trade-offs in my design?

- Students will be sending malicious traffic across the network at this Security Lab Manager. This could potentially violate any University policies.

- This application can launch Docker containers with full permissions. If the main application was compromised the attacker could use resources of the host machine and pivot onto other targets.

- The Security Lab Manager must be centrally hosted and have computing power to support the class size

Requirements

1. GUI interface for students to login, launch exercises, revert machines, and submit answers
2. GUI interface for teachers to login and view answers of students
3. Each exercise has a unique hash based on user, exercise, and admin private key
4. There must be at least 3 web exercises
5. There must be at least 3 desktop application exercises
6. The application should only allow a student to launch one exercise at a time
7. The application should be multi-threaded with locks on critical functions
8. Buttons pressed should give a “waiting” sign, not receive input, and have a “kill” button.
9. The application must be developed securely with static analyzer and must undergo scanning from OWASP ZAP. This application should be difficult to exploit or DOS.
10. This application should be extremely easy to setup. Every time the project is updated, Jenkins will run a Sonqarqube scan to ensure no new findings have been added. Then it will do a full build on a bare Centos7 system and run all tests to ensure functionality. If all tests pass, the production build will be updated. Anyone wanting to use this application should just have to download my repository and run a build script in bash. All dependencies will be installed.
11. This application can manage KVM machines. Users can put exported virtual machines from virtual box in here and they will show up for all students

Architecture Design

This will be a Django project that interfaces with docker to launch virtual machines. Architecture drawings have been made on scratch paper so far.

Development Prerequisites

- Jenkins and Sonarqube
- Testing Driven Development
- Agile principles
- Django framework knowledge

Development Plan

Agile development

What I will be doing on a daily basis

- Test driven development
- Working 2 hours per day during the week
- Am I learning and working efficiently?

- Does what I'm doing add value? Does it look and feel nice?
- What are my biggest roadblocks?
- How much am I actually accomplishing per sprint?

How should I track this and report status?

- I will put this in TFS
- I will give you a demo and summary bi-weekly of my progress. You will give me feedback and Ill update my backlog

Senior Project Documentation Requirements

What do I have to document for Fall?

- Engineering notebook
- Proposal Overview, Problem Statement and Background
- Requirements and Specifications
- Oral Presentation
- Ethics
- Mini posters
- Official Proposal

What do I have to document for Spring?