Simon Owens
Senior Project Planning 2018

Problem Statement and Background – Due October 1st

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. Setting this up is unique for every place that wants to do this: running vulnerable virtual machines would not be able to support a class size of one hundred students or resources could be wasted if too much infrastructure was allocated. Services would have to be setup, systems updated, and users would have to be added/removed. Students will also frequently crash their target computers which requires constant troubleshooting and resetting. If students are allowed 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 does not give students unique answers to submit. Setting up a victim and attack virtual machine takes several hours to do and doesn't directly help students learn security. Vulnerable machines from the internet also don't have unique answers, so one student could do the exercise, email it to the rest of the class, and the instructor would have no way of knowing who did the exercise. 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 on the internet. This is where my project 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 by comparing submitted hashes. Professors can also view how long students spent on their exercises, and all of the commands they sent. If the class isn't ready for hands on exercises, the instructor can easily create their own multiple choice exercises for students to complete. Hosting this application takes minimal resources and can scale easily to the class size. You can adjust the load balancer easily through the GUI based on how many students there using the system. The GUI and exercises will work seamlessly for all class sizes.

Requirements and Specifications

These requirements and specifications deliver the functionality that professors and students need in order to learn security at a rapid pace. The main goal of this application is to securely deliver a portal to professors and students to interact with virtual machines.

1. GUI interface for students to login, launch exercises, revert machines, and submit answers.

This GUI will have two main components: a grading page for professors and a page for students to interact with their exercises.

2. Each exercise has a unique hash based on user, exercise, and admin private key

This helps translate security exercises into grades for students. This feature helps prove that the student did their own work.

3. GUI interface for teachers to login and view answers of students

This interface should easily display which students have submitted answers and if any of their answers match each other. Since each student should have a unique answer, this will catch cheating. Professors should be able to assign grades within seconds.

4. There must be at least 3 web exercises

This allows users to immediately start practicing upon download. No additional configuration needs to be done in order to start learning. Web security is extremely relevant in today's industry.

5. There must be at least 3 desktop application exercises

This allows users to immediately start practicing upon download. No additional configuration needs to be done in order to start learning. Desktop application security is still relevant but less common in security jobs.

6. The application should only allow a student to launch one exercise at a time

This limits the resources the application consumes. Students can only work on one exercise at a time, so they should be restricted by the application.

7. The application should be multi-threaded with locks on critical functions

Users should never have to wait for an action server side to complete before issuing other actions. This makes the application feel nice and smooth.

8. Buttons pressed should give a "waiting" sign, not receive input, and have a "kill" button.

Spinning up some machines may take a minute, so the user should be able to cancel their actions at any time. They should also be alerted that the server is attempting to complete their action.

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.

Students that learn more about security may be tempted to try attacking this application for fun or to even change their grade. OWASP ZAP is a common web scanning tool which will help detect vulnerabilities during each build.

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. New users may not use the application if they need special infrastructure to run the application. If the application takes too long to setup, or takes troubleshooting based on the system, users may also lose interest.

11. There must be an nginx proxy in front of the application for scalability

Some environments may have two hundred students which could make the web application slow. Using nginx allows for static files to be delivered faster, and allows administrators to spin up more applications to meet the amount of users.

Simon Owens
Senior Project Planning 2018

Architecture Design

This will be a Django project that interfaces with docker to launch virtual machines.

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 and commands
- 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

Architecture diagrams can be found on this page: https://github.com/so87/Security-Lab-Manager

Development Plan

Agile development

Development Prerequisites

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

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?