Security Lab Manager: Virtual Security Training for Universities

Simon Owens

Computer Science

University of Evansville

April 2019

ABSTRACT

The Security Lab Manager is a web application that manages vulnerable virtualization machines for users to practice hacking on. Once hosted, users can login, choose an exercise, and start hacking in their own virtual environment. Administrators can login to view completed exercises and send grades to users. New exercises and machines can easily be added or imported.

LIST OF FIGURES AND TABLES

INTRODUCTION

Learning modern security practices is difficult and time consuming. Much of this time can be spent setting up safe environments to practice and reproduce vulnerabilities in. Some security exercises may not work because of different operating system protections, configuration settings, permissions, and patch version. The Security Lab Manager eliminates these problems by having a variety of exercises that can be setup in seconds. This application has a convenient interfaces that allows users to start, stop, and restart exercises, and submit answers when they are complete.

Users can practice exploiting web vulnerabilities, modern desktop C++/Java applications, and can import security exercises others have created. Administrators can create multiple choice questions to get users introduced to certain topics before doing hands on exercises. For each exercise, there are guides for how to code securely and prevent the vulnerabilities they just exploited.

Static analyzers, fuzzing, CI/CD, and proper programming practices were used for the development of this application. This makes the application production ready to be updated and easily deployed locally on a desktop, server, or in the cloud with AWS.

PROBLEM STATEMENT AND BACKGROUND

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 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.

There are a variety of problems when students are responsible for setting up their own environment and exercises. These exercises require virtualization software to run on because of software compatibility issues and risk of harming the student's computer. 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 does not directly help students learn security. Just getting one exercise to work might require installation and configuration of: an operating system patch, DLL, library, application, networking, firewall settings, registry settings, and anti-virus rules. This configuration usually requires 4GB of RAM, and a couple CPU cores on top of the student host OS. This which may be impossible for some students and extremely slow for others. Vulnerable machines from the internet also do not 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 this project comes in – the Security Lab Manager. It takes these vulnerable exercises others have already made and manages them so students can attack, destroy, and reset. 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. The GUI and exercises will work seamlessly for all class sizes. Professors have a nice interface to view competition of student exercises and be notified if any students cheat.

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. 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.

3. Students should be able to start, stop, cancel, and revert their security exercises

As a student, they should always know what action is currently processing, and have the ability to cancel.

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

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

5. The application should be multi-threaded

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

6. There must be at least 3 web security 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.

7. There must be at least 3 desktop application security 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.

8. The application must be developed securely with static analyzer and must undergo scanning from OWASP ZAP (a common web application scanning tool)

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

9. This application should be extremely easy to setup, updated, and have documentation

Administrators should only have to download and run one command to install the application. Administrators should also get reports on any issues, vulnerabilities, and code quality on the download page.

10. Each exercise must be uniquely identified for each student.

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

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.

DESIGN

Overview

This will be a Django project that interfaces with Docker to launch virtual machines. Figure – 1 show the architecture for how users can login and reach exercises.

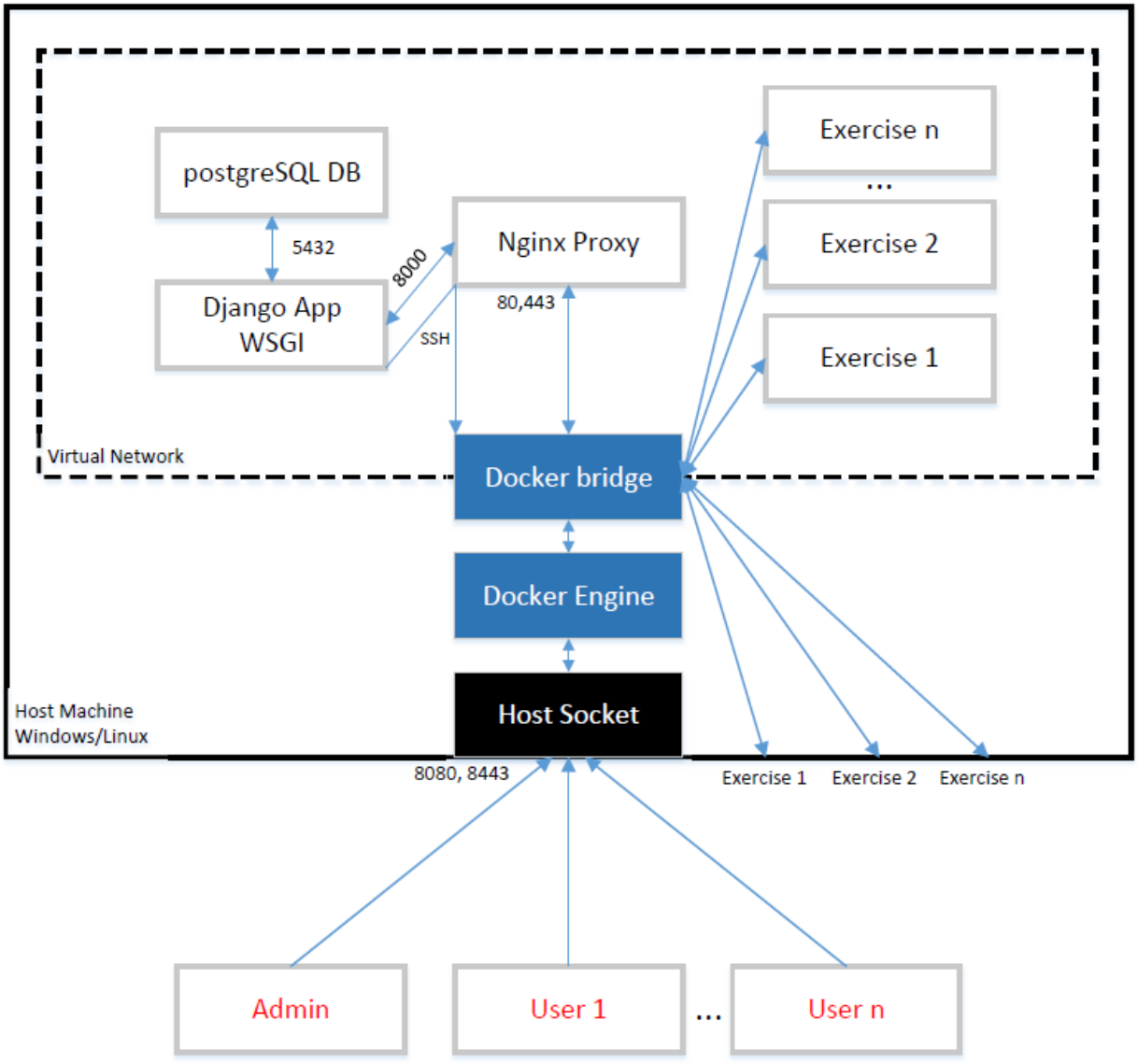

Figure 1 – Host Architecture

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.

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

Graphical Interface

The login page for users and administrators is the same.

Figure 2 – Login Page

Users will be directed to the below page where they can launch exercises and submit their answers. Users can see all of the different sections, with all of the exercises associated with them.

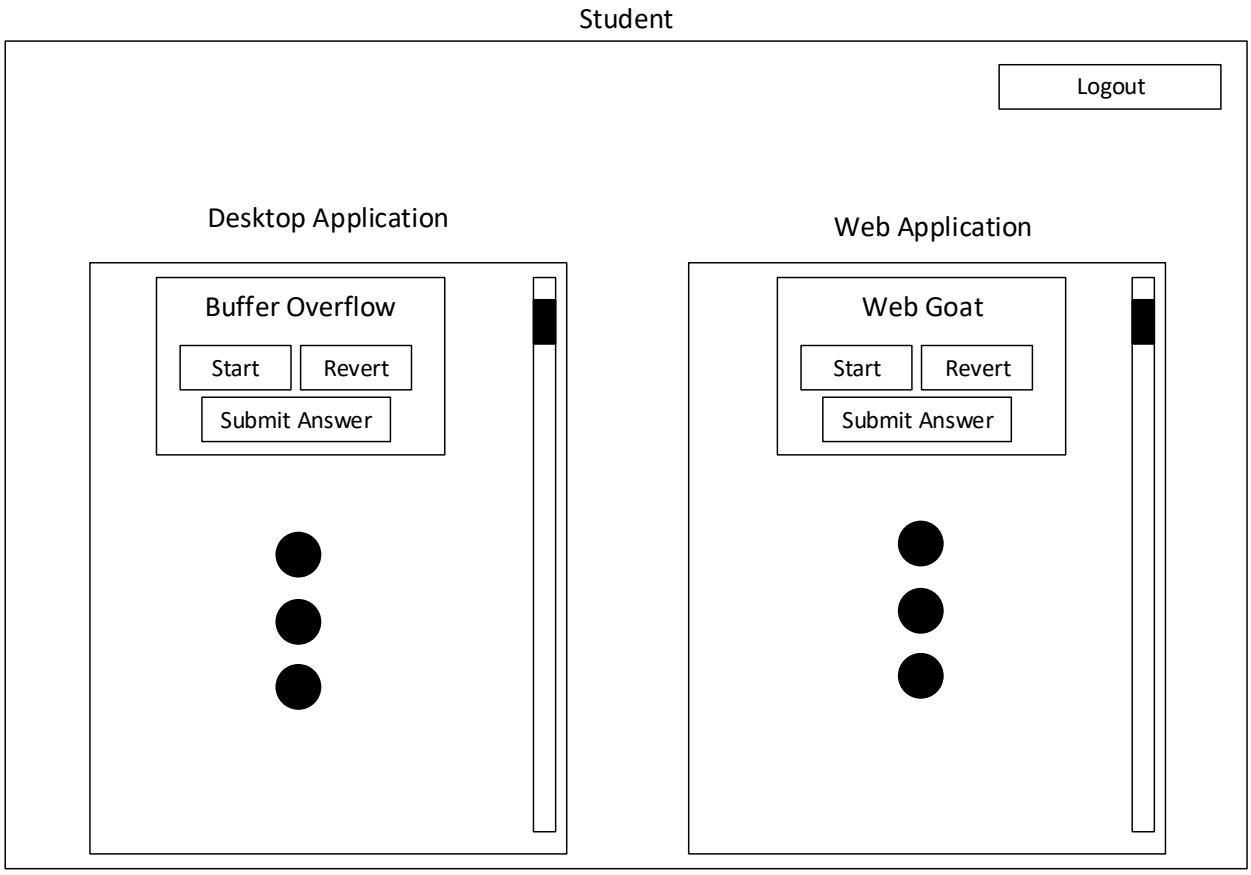

Figure 3 – Student View

Administrators have an entirely different view – they can manage the performance of the application and see which students completed their exercises.

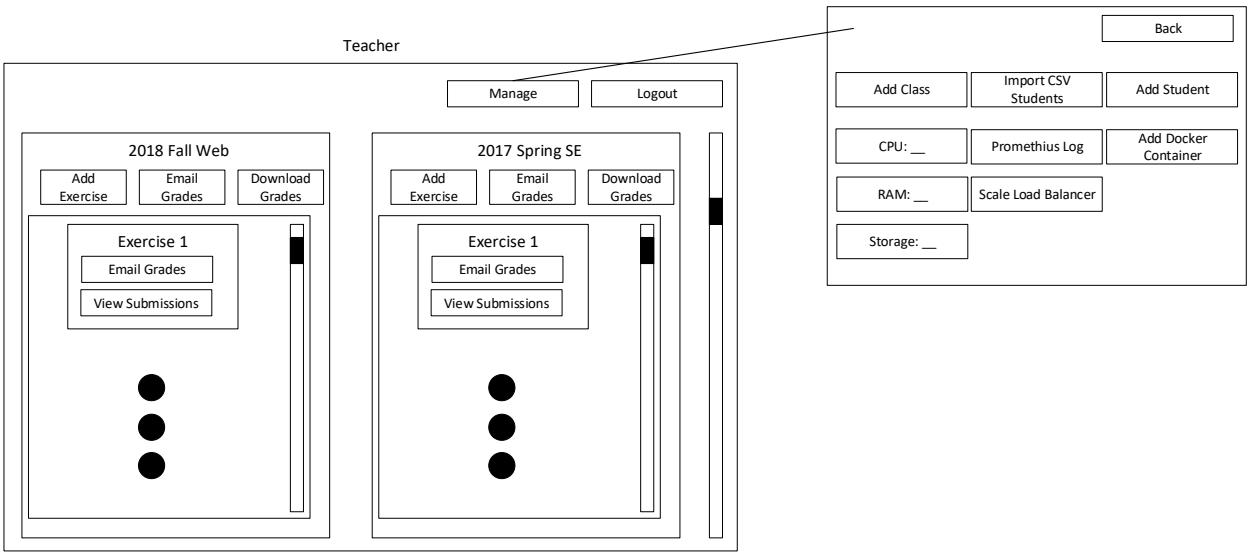

Figure 4 – Administrator View

They can easily scale the application, add users, and send out grades to students.

Database

more analysis is needed

DEVELOPMENT PLAN

Development Prerequisites

Before development of this application can start, a strong knowledge of CI/CD, Agile principles, Web Security, and the Django framework are required. These skills allow the secure delivery of what best fits the needs of the product sponsor's needs. In order to understand these practices, the following books have been read.

- Scrum – by Jeff Sutherland
- The DevOps Handbook – by Gene Kim
- Test-Driven Development with Python – by Harry J. W. Percival

Agile – Sprint 1

The goal of this senior project is to accomplish all of the requirements already laid out. The requirements may be removed or added once the velocity of development is understood. There will be a weekly demo on the new functionality added. Feedback will then drive the work for the next week until the end of development. Requirements will be broken down into tasks – which will be less than 4 hours and have a value weight associated with them. These tasks will be managed through Waffle IO on the github repository page. All of the hours will be tracked in the engineering logbook located at csserver.evansville.edu.

Continuous Integration and Continuous Delivery

This project is developed using CI/CD via Jenkins. This allows the application to easily manage dependencies, vulnerabilities, and enables easy contribution. Below is a diagram for how the application gets developed and deployed.

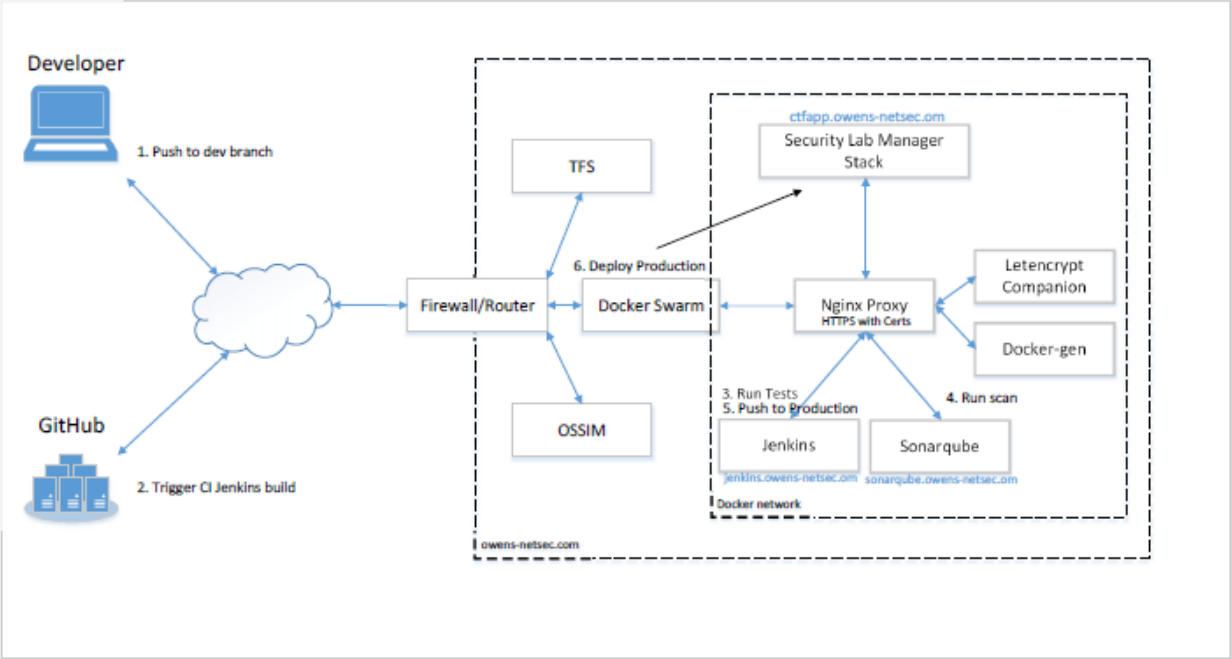

Figure 5 – Development and Deployment Process

- 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 web scan is done against the system every build to ensure none of the OWASP top 10 exist in the web application

REFERENCES

APPENDIX – A