2021

Building a competitive platform for cyber security and computer science technical challenges

Zachariah Pelletier

Building a competitive platform for cyber security and computer science technical challenges

Abstract

To answer the question of whether creating a competitive environment drives students' engagement while completing class objectives, two EMU Honors students designed and built a web application that creates such an environment. This system allows students to complete assignments that are weighted on a point system and compare scores on an anonymous "Leaderboard". This system attempts to emulate a competition environment for objective-based learning and is designed to be used for Information Security lab assignments similar to a Security competition environment. Although the primary proof-of-concept labs for this project for this project are in the Computer Science and Information Security realms, such a competitive platform can be used for a variety of disciplines and subjects.

Degree Type

Open Access Senior Honors Thesis

Department

Technology Studies

First Advisor

James Banfield

Second Advisor

William Sverdlik

Third Advisor

Phil Rufe

Subject Categories

Computer Sciences

BUILDING A COMPETITIVE PLATFORM FOR CYBER SECURITY AND COMPUTER SCIENCE TECHNICAL CHALLENGES

By

Zachariah Pelletier

A Senior Thesis Submitted to the

Eastern Michigan University

Honors College

In Partial Fulfillment of the Requirements for Graduation

with Honors in Computer Science – Applied and Information Assurance/Cyber Defense

Approved at Ypsilanti, Michigan, on this date April 21, 2021

James Banfield 04/01/2021

Supervising Instructor (Information Assurance/Cyber Defense): Dr. James Banfield

William Sverdlik 04/01/2021

Supervising Instructor (Computer Science – Applied): Dr. William Sverdlik

James Banfield 04/01/2021

Honors Advisor: Dr. James Banfield

4/6/2021

Department Head: Phil Rufe

Ann R. Eisenberg, Ph.D. April 21, 2021

Honors College Director: Dr. Ann Eisenberg

Abstract

To answer the question of whether creating a competitive environment drives students' engagement while completing class objectives, two EMU Honors students designed and built a web application that creates such an environment. This system allows students to complete assignments that are weighted on a point system and compare scores on an anonymous "Leaderboard". This system attempts to emulate a competition environment for objective-based learning and is designed to be used for Information Security lab assignments similar to a Security competition environment. Although the primary proof-of-concept labs for this project for this project are in the Computer Science and Information Security realms, such a competitive platform can be used for a variety of disciplines and subjects.

Introduction

Information Security and Computer Science assignments are notoriously technical and can sometimes be very tedious. As a student in both disciplines, I have experienced many challenges, programming assignments, security case studies, and other in-depth technical challenges that were often hard to complete due to motivational drain and burn-out. Many of my personal empirical observations and discussions with other students in both of my programs have led me to believe that I am not the only student experiencing this and looking for a way to motivate my education in this way. A common and popular way for many people to get engaged in certain new topics within the Computer Science or IACD fields is to attend conferences and other group events where experts and enthusiasts of similar technical areas come together to engage in collaborative and competitive technical challenges.

This environment is largely popular because it combines a combination of comradery among team members and competitive motivation among the competitors. In the Computer

Science realm, these conferences and events usually take place in the form of "Hackathons" where teams or individuals engage in a multi-hour to multi-day competitive event to create an application or solution to a common problem. In the area of Cyber Defense, professionals and students that attend conferences often join to participate in CTF (Capture the Flag) and other competitive security challenges that involve a variety of concentrations including Digital Forensics, Penetration Testing, Social Engineering, Red Team/Blue Team offense/Defense, and many more.

This process of creating a competitive environment where there is a point-based scoring system and weighted elements for competitors to use to compare individual standing is known as "gamification". Gamification works by allowing optional anonymity and allowing people to compare their personal scores and achievements among their peers. From this, two queries arise that are somewhat interrelated: "Does gamification really drive user engagement?" and "Is creating such an environment going to benefit some users more than others?". Both of these questions must be kept in mind when creating such a system. In order to allow for future research surrounding the topic of gamification in an academic setting, I worked with one other IACD Honors Student to design, create, test, and deploy a fully scalable web application that allows professors in our department to do just that. We have come to call this application Neuralabs.

There is no doubt that this project is a large undertaking that in its entirety will take more than 150 hours to complete. Even with two students working on this project for a year-long duration, there is still a lot of ground to cover and these students will attempt to accomplish what usually takes multiple years and several teams of professionals working full time. To date, both students have worked an average of 10 hours a week for the last two semesters as well as time spent throughout the Summer totaling around 400 hours of work per student. Even with this large time allotment, there is still a lot of work to do. It is our hope that our faculty advisors use the progress we have made as a launching point to conduct further research in this area and utilize our system. We hope that other students in our department get the chance to expand upon and improve our system and conduct research surrounding the use of our system in an academic setting.

Related Work

Using CTFs for an Undergraduate Cyber Education

Faculty members at the United States Airforce academy have been focusing on competition-based learning since 2010 when students formed an official team centered around competing at both national and international Cyber Security conferences. The authors say that "Through our experience in fielding a cyber team, we have grown to appreciate students' significant learning from these competitions. Most notable is the increased motivation level of some students. Students would put in many hours trying to solve a CTF challenge, doing research on the web and implementing possible solutions. Their motivation was significantly greater than for their traditional class assignments. The learning also helped them with their regular coursework." (Carlisle et al, 2015). The authors' empirical observations in this case support a more productive stance for gamification.

Given the success in driving student engagement during isolated competitions, the faculty responsible for organizing the team for CTF events took over the Cyber Security curriculum and converted it into a "CTF-Style classroom" (Carlisle et al, 2015). In this paper, the authors introduce both the details of exactly how they implemented such a curriculum and then how the program impacted their students as well as issues they faced during implementation. This implementation was achieved by incorporating both a set of fifteen Cyber Security based courses into the required core set of classes for academy freshmen and giving a set of laboratory-based hands-on assignments throughout the program. Immediately following this, there is a focused Cyber training elective that is chosen by each student. These concentrations consist of different focused Cyber Security topics that are delivered through a CTF-style architecture.

The unique feature of the second phase is that the program is entirely student-led with a select few of the rising Juniors and Seniors providing mentoring and support services while the Sophomores work through the security challenges gain exposure to different Cyber Security paths. With the completion of these assignments, points are assigned to students that increase proportionally to the relative difficulty of the assignment. All students are aware of each other's scores via an anonymous scoreboard system, with an additional incentive to score higher being the acquisition of a Cyber Badge that can be worn on the student's uniform as a symbol of accomplishment. At this point, students have the opportunity to continue their Cyber Security education by declaring the Computer and Network Security major and help to continue expanding the gamified offerings.

The outcome of the incorporation of this program was measured in feedback from students and the engagement and general depth of questions asked/answered. Additionally, the authors noted that "…by the time our cyber students reach their junior and senior years they have developed a true desire to keep learning and experimenting." (Carlisle et al, 2015). The first major obstacle that was encountered by those who created and developed the program was the necessity for every feature of competitive environment to be carefully designed and be centered around meaningful learning objectives. These difficulties were in a large part solved by developing a consistent framework through which all challenges would be created and taken.

This allowed instructors to have granular control over what conditions surround each challenge and the unique problems that are given to each student.

One finding that was especially interesting to me was that students who competed in the CTF environment asked less questions on average than students in traditional academic settings, but their questions were more informed and focused. In the future, the authors hope to expand the CTF program further, branching out to new topics and enhancing pre-existing coursework with CTF challenges. This case study of introducing the organization and delivery of academic material in this format is generally supportive of the use of gamification for academics, specifically in the Cyber Security realm.

Increasing Student Intrinsic Motivation and Self-Efficacy Through Gamification Pedagogy

The authors of this work aim to present their findings from research surrounding using Gamification Pedagogy as an alternative learning method for students completing System Administration and Cyber Security tasks. The study conducted relied on qualitative data taken from student feedback from 96 students in the IACD program at Eastern Michigan University. This method of applying Gamification to common academic tasks is a subset of Experimental Learning Theory (ELT). The study that was conducted was centered around understanding the perspectives of the students participating in the study. The authors wanted to understand how Gamification of required academic material affected the student's intrinsic motivation as well as their self-efficacy.

The setup of this study was as follows. Students in both classes that were used for this study were first given a traditional set of didactic exercises in networking. They were asked to complete a set of problems using traditional algebraic computation to find potential IP addresses in a network range. At this point, the traditional exercise was complete and the students were asked to supply their answers as well as feedback regarding the relative motivation and intrinsic engaging nature of the exercise. In a separate exercise, the same group of students were asked to complete a gamified exercise where they were given an interactive real-life network. They were given the IP address of that node and asked to calculate the subnet mask, a task which is common in computer network to organize and compartmentalize networks. At this point, the students were asked to scan the network using a common network scanning tool called NMAP.

Another major difference between this exercise and the last one that the students completed was that the whiteboard in the room was used to tally the number of students who had completed the exercise and discovered all of the necessary pieces for the assignment to be complete. As mentioned earlier, the responses to the different exercises were measured in terms of motivation and effort put into the task by the students using qualitative analysis of response provided by those students. The authors found that the effect that gamification had on this task was that "Competition has a constructive effect on participation and learning that will result in higher learning through social pressure to achieve" (Banfield et. Al, 2014).

The Seed Project

The SEED project is an online platform that provides and curates hands-on labs for computer and information security topics. The system was created by Wenliang Du, a professor of Electrical Engineering and Computer Science at Syracuse University. This system uses segmented virtual machines to deliver content covering topics such as Software Security, Network Security, Web Security, System Security, Cryptography, and Mobile Security. Labs on this system are built inside of files that can be loaded into virtualization software so that users can download an entire operating system that comes pre-loaded with all of the software needed to complete the given lab. Also included in each of these modules are an instruction set (usually in the form of a PDF file), additional informational videos covering the topic which are given via Udemy and other online learning platforms, and additional reading from online sources on the given topic.

This system allows its users to complete labs at their own desired pace, allowing the user to complete it within their own timeframe, or to use it as instructional material for an actual class in an academic setting. This process is assisted by the presence of an estimated timeline for completion, either self-paced or in a supervised environment, as well as an estimated difficulty so that users can gauge their ability to both complete the exercises and retain the material presented within them. Another nice feature of this system is that it was created as an open-source project at a university, which has allowed students, faculty, and other members of the community to contribute to it in a meaningful way. In this respect, this project is truly open source[1].

Project Design and Architecture

The initial design of this project was conceptualized to be a large-scale web application that can be used by many people at once. In traditional Computer Science terms, a web application is a shared application that is used by multiple users at once (known as “clients”) and can be accessed through a web browser. This technique of delivering content is very popular and has been used with great success at providing a shared solution to common workflows. The operational part of the system will exist on a web server that is configured to allow users to connect to it from any location that allows for a connection to the internet. In addition to building such a system from the ground up, our team also decided to exercise many of the Security concepts that we have learned in our program and fully deploy the system to a physical server

[1] Open-source software can be contributed to and used to collaborate with the general public due to its publicly accessible design. More information can be found at https://opensource.com/resources/what-open-source.

(through our department) as well as test it for Security vulnerabilities and harden it against potential threats.

This project uses a popular programming framework called *Flask* based on the programming language Python. This framework allows developers to create a web framework that displays styled templates to the user. We decided to build out the front end of our application using JavaScript, JQuery, and a front-end styling library called Foundation. These are all common programming and markup languages used for web-based applications. Our web application would utilize a database system used to store persistent data for our users regarding lab material, scores, lab attempts, and user information regarding relationships to individual courses. The software we decided to use for this purpose is called MongoDB and it is a very popular NoSQL non-relational database solution. This framework was chosen for both its efficiency and its ability to process and transfer web data in a common format. The individual components and how they interact are shown in figure 1 of appendix A.

Our goal was to design the application in such a way that it could be easily scaled in size or be distributed among several servers if needed. Due to feasibility and equipment constraints for the project, we only had access to one server on which we stored all of the interacting pieces of the system, but we tried to design the application in such a way that the physical equipment that supported the software could be switched out very easily. Overall, the design of the project balanced efficiency of computing resources with usability and speed of the program for the end user. Careful consideration went into which tools we used and how elements of the system were constructed in order to maximize all of our goals for the usability of the final product.

Project Development

In the context of Software Engineering, “Development” is the task of programming and working to build the program itself. This process is also known as “Coding” and can involve teams of up to hundreds of software developers on one project. As with any other complicated project, such as building a building, the team that carries out the construction must be well organized and coordinated in order to successfully complete a project within the desired timeframe. There are many different techniques and systems that are used to build and validate a piece of software as well as choose which features are to be implemented in a certain sequence. This methodology is very important when starting a large project to ensure that consistent progress is made in a methodical way. For this project, we decided to use a programming methodology known as the “Agile” method.

This methodology is described as “an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.” (Atlassian, 2021). This is a more free-flowing form of development that relies more on writing program code that is modular and easily modified so that code can be written quickly and be frequently changed as the architecture of system evolves. The primary facets of this system are that the software developers are free to implement the desired features in any form they wish as long as the program that is written can be used by other features that other team members are working on. This allows developers to quickly build a program while keeping a high-level view of the overall project. This does not mean that high-level planning does not occur during this process, it is just that the overall plan is somewhat temporary and will often change as the requirements and validation of the project changes.

There are many implementations of this methodology, but a common one that we have chosen to use on this project is known as the SCRUM system. This is a term that comes from the sport Rugby and is used to describe the start of play. There is a SCRUM master who organizes the team and makes high-level decisions but does not have any more power than any other member of the team. It is a very non-hierarchical system and works well with software that has loose requirements or requirements that change often. The primary tool we used for project management was a platform called Trello which is a visual planning tool that can be used to create tasks with statuses that can be organized into lists. This system allowed our team to break desired features into manageable tasks and select certain programming processes to be prioritized.

The SCRUM system breaks the development process into periods of time that have a set timeline known as “Sprints”. These sprints are usually 2 weeks long and have certain selected development tasks associated with them. During the planning of a given sprint, the team members work with the client (in this case, our faculty advisor dictated what we worked on and when) to prioritize what work should be done during each sprint. During the initial planning process and throughout the development process, tasks were created to cover each and every desired feature and are automatically put into a list known as the “Development Backlog” which is the default bucket that tasks are put into that are not planned for development soon. When a new sprint is started, tasks are selected from the backlog due to relative importance and sometimes necessity if, for instance, the given tasks is a pre-requisite for another task that is a high priority.

At the beginning of each sprint, each team member often volunteers to take on certain development tasks and the work is divided on a purely voluntary basis. If all tasks are completed during a given sprint and team members are without work, they can simply take a high priority task from the backlog and continue work. If all of the tasks that were selected for development for a sprint hadn't been completed, they would be moved back to the backlog and re-evaluated for development during the next sprint. During the development process, our team used common programming principles such as modularity (code objects can be reused), efficiency, readability (it is easy to understand code that someone else has written), and other common standards used in our field. Both students worked very well together on this project and were able to accomplish most development tasks within the given time frame.

In order to complete all of the code implementation on this project, our development team of two needed a method by which to collaborate and work on the program together. A common solution for this is the use of a "version control repository", or a common location to store code that is accessible over the internet and can be used to collaborate. In our case, we used version control software called Git. Our code was hosted as a private repository (not accessible to anyone without explicit rights to see the code) on a website called GitHub. Utilizing this software, we were able to push and pull code updates from this repository, and merge pieces of code together when we were working in similar areas of the application. This allowed us to collaborate virtually and manage changes to the codebase.

It is important to note that the software development process took a majority of the time during this project. This is due to the reliance of this project on the quality of our software implementation, and our limited resources in terms of time constraints to work on the project. At the end of the project, there will be a few development tasks left in the backlog to be accomplished, but all of the desired functionality is recorded so that future students who choose to participate in this research can continue where we left off. All critical tasks that contribute to the core functionality are to be completed this semester.

The resulting application had two different access levels of authorization that could be assumed when using it. Users could either be assigned as base-level users, or administrators. The primary difference between the two levels of access were simple: the administrators could create new courses, instructors, manage access codes, and add tags to the system as well as delete labs while the base-level users could only create community (non-course specific) labs and take labs that they were given access to. The entry-point of the application was the login page[2] where a user could enter their login credentials to gain access to the app. If the user had never used the app before, they could click on the *sign up* link which would bring them to the registration page[3] and allow them to enter the required information such as their username, email, school (if chosen), and password.

Following the login/registration process, the user is brought to the index (landing) page of the application, known as the Dashboard[4]. The dashboard serves as the access point to get a snapshot of the current state of the application. For administrators, this where they could see all of the labs they have created and allows for navigation to modify the data in the application. For regular users, this is place where they could resume labs they had started but not yet finished, start new labs, and see their stats as they compare to other users. The relative standing of each user is shown according to the number of points they have collected, based on the number of labs they have finished and their relative difficulty. The profile page[5] is the area of the application

[2] See figure 2 in appendix A for the user interface of the login page
[3] See figure 3 in appendix A for the user interface of the registration page
[4] See figure 4 in appendix A for the user interface of the dashboard page
[5] See figure 5 in appendix A for the user interface of the profile page

that is used to edit any user information after registration. Information can also be accessed here relating to any other members that have been added to the users' "Friend list" and all account privacy information. This page is consistent for both regular users and administrators.

Another source of general information on the current status of a user can be found on the leaderboard page[6] of the application. This page serves as the main source of information regarding the gamification aspect of the system. Here, the user can identify their current standing against other users who have completed labs. This is shown as a leaderboard showing the number of points and the current level obtained by each user. This information can be anonymous as each user is registered under a pseudonym username and can choose whether or not to share this information with others. This information can also be found on the main dashboard page which is updated with the current level and number of points a user has collected every time a new lab has been completed. The dashboard page also includes a radar graph showing the frequency of completed labs that require certain skills.

The lab creator and editor page[7] is available to all users regardless of authorization level. It is important to make a distinction between certain types of labs that are created. Official courses are managed by administrators and are assigned to instructors, however any user can create a lab regardless of whether or not they are an instructor of a course. These "General" labs created by non-affiliated users are added to a "Community" bucket of labs where users can take each other's labs, but do not have to be a part of a course to do so. The administration console[8] is the primary tool used by administrators to monitor and moderate the application. Through this

[6] See figure 6 in appendix A for the user interface of the leaderboard page
[7] See figure 7 in Appendix A for the user interface of the lab creation and editing page
[8] See figure 8 in Appendix A for the user interface of the administration console

page, any admins on the system can create new official tags to be used in the creation of labs, create new instructors, create new schools, and officiate courses on the platform.

The lab management console[9] is primarily used for visualization of created courses, sorting courses by course (that the user is an instructor of), and creating new labs/editing labs that have already been created. This console is an all-in-one portal to managing labs that the user of the platform has created. Through this page, the user can use sorting mechanisms to find specific labs that are associated with individual courses and view all of the static information about the lab itself including the number of points the lab is worth, the name of the lab, and all of the lab information.

The page for taking the lab itself is only accessible to users that are enrolled in the course that the lab is associated with, or all users if the lab has been published to the *Community* bucket. This page is known as the lab attempt page[10] and is the vehicle through which users make actual lab attempts and take the lab itself. This allows students to access the material that is uploaded when a lab is created and submit answers for each page of the lab to obtain points. This interactive lab-taking session is saved on each page submission. Finally, after a lab is submitted, the user is taken to a lab completion page[11] where they are given the time that the lab was submitted and the number of points that were accumulated as a result.

Architecture of the Scoring Engine

The scoring engine is possibly the most integral part of the system due to its central role in automatically scoring the lab attempts of users. Scoring Engines are commonly used in CTF and other security challenges to automatically accumulate points for a given team depending on

[9] See figure 9 in Appendix A for the user interface of the lab management console
[10] See figure 10 in Appendix A for the user interface of the lab attempt page
[11] See figure 11 in Appendix A for the user interface of the lab completion page

whether or not the qualities it was set up to monitor are satisfied. For instance, if a security team is defending a server ecosystem and are supposed to keep their website hosted on one of the webservers accessible regardless of attacks from opponent teams, the scoring engine would consistently ping and test the web server to ensure that it is fully operational. According to the uptime of the scoring engine specifications, points can be given to an individual or team.

Our implementation of the scoring engine in our application is similar to this aforementioned application. In our case, we wanted to make our scoring engine as generic and extensible as possible so that our users could utilize as many methods as possible for grading their labs. Our initial implementation was a simple static check with a question and an answer. Our scoring engine would check each attempt's answers against those provided upon lab creation. Our scoring engine was smart enough to account for discrepancies in answers such as differentiation in capitalization. In the future, we hope that the scoring engine can be extended to allow for different methods of scoring for labs. It was anticipated that instructors would be able to embed flags inside of files, create scripts for automatically grading more complicated labs, and allow for instructors to upload their own scoring engine implementations.

The architecture and design of this system was purposely designed with the future expansion of the module in mind. This was done through a combination of using object-oriented programming principles to create a universal system of scoring labs and creating an initial scoring technique that is its own separate "module". It is our hope that future modules can be made and essentially plugged into the scoring engine to enable many different scoring techniques. The only requirement of the scoring module is that it sets a local variable representing the total number of points achieved through scoring. This can be done however the author likes, as long as it is returned in a consistent format and utilizes the pre-existing scoring standards.

Deployment to Production Server Environment

After the application was developed to a point where it was ready to be deployed onto a piece of physical hardware, a donor server was procured for the project from the School of Information Security and Applied Computing. This equipment was taken by one member of our team and hosted at their place of residence so that it could be configured and managed without access to campus facilities due to the ongoing COVID pandemic. We were able to engage both of our team members in this process by setting up remote access to another computer on the home network that hosted the server. This allowed both team members to work side-by-side on the deployment process as if they were in the same physical space.

The server that was used for this process was a Dell Poweredge 610[12] that had an adequate amount of hard drive space for deploying our proof of concept. Our team installed the latest version of Ubuntu Server as the base operating system and configured the environment to allow incoming web connections through the firewall and allow for remote access via SSH (Secure Shell). At this point, our team could install and configure the relevant web server architecture and platforms that we would need to allow the server to be accessed securely by the outside world. Flask documentation recommends using an external server option. They say "While lightweight and easy to use, Flask's built-in server is not suitable for production as it doesn't scale well. Some of the options available for properly running Flask in production are documented here. If you want to deploy your Flask application to a WSGI server not listed here,

[12] See figures 12, 13, and 14 for pictures of the server hardware used.

look up the server documentation about how to use a WSGI app with it. Just remember that your Flask application object is the actual WSGI application." (Flask, 2021).

The WSGI application we decided to use for this purpose was a framework called Apache. Apache is commonly used to host web applications and other web solutions through its wide use and ability to be run on most modern operating systems. Apache was downloaded fairly easily onto our Ubuntu server operating system that we had installed on the server. After downloading Apache, we completed some routine configuration tasks in order to set up Apache such as allowing the service to be accessible through the firewall and ensuring that all of the required directories were present on the system. We also configured the Apache service to start on server startup.

Finally, all relevant code to the system was transferred to the server and MongoDB was installed as a dependency. After running the software and configuring it for deployment in our production environment, the system was tested to ensure that it could be reached remotely through a web browser and that it could handle a certain load of users using the service at the same time (this process is known as load testing).

Penetration/Security Testing of Production Deployment

After the system was deployed to the production environment, another security process was undertaken by the team which is known as Penetration Testing. This security testing procedure is used to find potential vulnerabilities in software and the systems they are deployed onto using common hacking methodologies. This process uses some of the same hacking tools that are used by malicious hackers to exploit systems and tries to simulate a real-world environment as closely as possible.

Upon completion of development for the Neuralabs web application, both developers took on a different role in order to perform security testing and full evaluation of the deployed web application in a production environment via grey-box penetration testing. All testing procedures were performed in accordance with the Open Web Application Security Project (OWASP) and followed MITRE vulnerability assessment standards. The scope of the testing is as follows:

- Testing of the server configuration including open ports, running services, and security patches.
- Password cracking check on login.
- Authorization checks for endpoints.
- Vulnerability scans of the entire system.
- SQL Injection checks on inputs.
- XSS evaluation.
- Token manipulation checks.
- Malformed URLs.

The first step we took to try to identify general areas of vulnerability on both our application and the hardware it was deployed to was to scan both the server we used for deployment and the web application itself. In Cyber Security, the term Vulnerability Scan refers to taking a list of known vulnerabilities that exist in the systems or services involved and detect if the software or configurations that present this vulnerability exist on the device under test. A vulnerability entry is commonly known as a "CVE", or Common Vulnerability and Exposure. The primary tool we used for performing vulnerability scans on our web application and deployment hardware is a widely used tool called "Nessus". Nessus allowed us to enter specific information about the scans that we performed, such as the IP address of the target we were scanning, what ports to scan, methods by which services should be discovered, and even login credentials to our application so internal pages could be tested.

The nice thing about using Nessus is that it has a nice user interface that takes the form of a web application itself. This interface allows the security tester to have full control over the tests that are being run, schedule tests to be run on a regular basis, and export full reports summarizing any of the scans to various formats[13]. Vulnerabilities that are found during a scan are measured on the following scale (ranging from highest to lowest): Critical, High, Medium, Low, Info. Additionally, each vulnerability is assigned a CVSS (Common Vulnerability Scoring System) score which allows the relative importance of each vulnerability to be assessed and weighed against others. This relative importance also helps prioritize which vulnerabilities should be remediated (or fixed) first and allows the security tester to focus on specific areas of the tech stack.

After high-level vulnerabilities had been identified and cataloged, our team moved to attempt to exploit each of these vulnerabilities as a proof-of-concept. This both demonstrates the urgency of fixing these security flaws and allows the security testers to confirm that the vulnerability does indeed exist. As this process is completed, each tester took very specific and precise notes regarding what we did, how we went about doing it, and other environmental conditions that were present when the vulnerability was exploited and proved out. These notes are very important and form the basis of our final deliverable for this portion of the project. Finally, after all security flaws had been identified and thoroughly tested, our team moved to the final step of the penetration testing process: reporting the findings of the testing.

[13] An example of the reports generated from Nessus can be found in Appendix A, figure 15.

Our format followed industry standard guidelines and consisted of the following sections in order: Table of Contents, Details, Tools Used, Executive Summary, Vulnerability Scans, Exploits, Conclusion. Within each of our vulnerability scan and exploit sections, we mainly followed the following format of reporting information: Relative Priority, Description of the threat, Proof of Concept, Remediation and Status. Typically, security testers are not the ones to actually fix and patch the security flaws they find. This is sometimes seen in organizations that do not have the resources to hire a separate set of specialists for security patching, but in this case, we only had two team members to complete all phases of the project, so we completed each phase of the process ourselves.

Through our security testing process, we found several vulnerabilities of interest. The first major configuration vulnerability that we encountered was an insecure version of Apache that we had installed. According to the Nessus compiled database of vulnerabilities, the latest version of Apache, and also the most secure version with the most security patches is version 2.4.46. We had installed version 2.4.36 on our server which according to the CVE database of exploits, contained several proxy issues, URL manipulation vulnerabilities, and execution of code with escalated privileges (CVEDetails, 2019). Our proof of concept was to initiate a simple URL manipulation attack where we entered consecutive forward slashes (/) in the URL and were able to assess information about the structure of the backend of the application, as well as various endpoints that existed on the system.

The second major vulnerability that was discovered through the penetration testing process was that there were vulnerabilities that existed in the Linux Kernel itself in the version of Ubuntu server that we installed (Ubuntu, 2021). We verified which version of Ubuntu was installed, and also determined with security patches were applied at the time of scanning. This revealed that

Ubuntu was not fully up to date, and that there were pending security patches that needed to be updated.

After finding and documenting all of our vulnerabilities and their respective exploits, our team moved to transition into the final step of deployment for our web application: hardening and patching our environment. This step is largely guided by both external CVE findings based on community-sourced exploration and the report that was generated through the penetration testing and security testing step. This document allows security engineers and IT administrators to examine the remediation steps laid out by the security team and implement fixes for these security flaws accordingly. In this case, we used this document to patch security flaws and harden our deployment server.

Server Environment Hardening

The last step of deployment that our team engaged in was to patch all of the security vulnerabilities that were discovered during the security testing phase. These were done in accordance with the penetration testing report that was created as a result of the security testing done by the team before. The first step our team took to fix security patches through this process was to upgrade and update all existing operating system and third-party packages using the package manager apt (Ubuntu, 2021). After both of these operations were complete, our team checked all appropriate versions and package information to ensure that everything was up to date and all applicable security patches had been applied. At this point, work could begin to secure the Apache service we had running on the server to ensure that all web application related processes were patched and secured. The first step we took towards this end was to stop the Apache service from running so that we could make modifications to the configurations. Next, we updated Apache to the latest version and made sure that all security patches were applied.

Finally, we worked on securing the application itself. We wrapped the entry point to the script that launched the Flask app in an Apache configuration module so that all of the relevant Apache security features were applied. Next, we configured the startup of the application itself to be in production mode (a Flask deployment mode that prepares the app to be served to clients) and generated a security secret key based on a 2048-bit RSA private key that was generated on an air-gapped computer and transferred to the server via USB flash drive. Lastly, we modified the firewall rules to only allow SSH access to the machine via OpenSSH and to allow all connections to or from the port utilized by the Apache service[14]. Note that these rules allow access to port 22 used for SSH from any address. This was allowed simply for configuration purposes and should be only allowed from a specific maintenance address after it is exposed to a larger network.

Future Work

Although a lot has been accomplished through the completion and presentation of this project, there are still many applications and enhancements that can and should be made to this system. Luckily, during our weekly team discussions, my colleague in the department and I captured most of the feature considerations we brainstormed to our Trello board. We were not able to finish every single task we captured during this process, but the list of considerations lives on in digital format. The combination of this and our fully integrated Git repository will allow us to pass this project on to our department should any other students show a desire to work on it.

Even though we were only able to finish the technical implementation and testing of this application during our time on this project, the research portion and findings of this project

[14] An image of the firewall rules can be found in Appendix A, figure 16

remain open-ended. There are many applications of this project, and empirical data can be captured that could provide a basis for understanding the relation between student motivation and engaging in a competitive space. I would like to encourage students in our department as well as other areas that have an interest in understanding student motivation to complete coursework as it relates to engagement (especially for highly technical tasks) to continue research in this area and to use this project as a launching point.

References

Deployment options. (n.d.). Retrieved March 26, 2021, from https://flask.palletsprojects.com/en/1.1.x/deploying/

Carlisle, M., Chiaramonte, M., & Caswell, D. (2015). *Using CTFs for an Undergraduate Cyber Education* (pp. 1-6, Tech.). Washington DC, Maryland: USENIX.

Banfield, J., & Wilkerson, B. (2014). *Increasing Student Intrinsic Motivation and Self-Efficacy Through Gamification Pedagogy* (4th ed., Vol. 7, Ser. 2014, pp. 291-298, Publication No. ISSN-1940-5847). Littleton, CO: ERIC. (ERIC Document Reproduction Service No. EJ1073237)

Du, W. (n.d.). Seed project. Retrieved March 26, 2021, from https://seedsecuritylabs.org/about_us.html

What is open source? (n.d.). Retrieved March 27, 2021, from https://opensource.com/resources/what-open-source

Vulnerability details : Cve-2019-10097. (n.d.). Retrieved March 28, 2021, from https://www.cvedetails.com/cve/CVE-2019-10097/

Vulnerability details : Cve-2019-0220. (n.d.). Retrieved March 28, 2021, from https://www.cvedetails.com/cve/CVE-2019-0220/

Vulnerability details : Cve-2019-0211. (n.d.). Retrieved March 28, 2021, from https://www.cvedetails.com/cve/CVE-2019-0211/

USN-4887-1: Linux KERNEL Vulnerabilities: Ubuntu security notices. (n.d.). Retrieved March 28, 2021, from https://ubuntu.com/security/notices/USN-4887-1

Package management. (n.d.). Retrieved March 28, 2021, from https://ubuntu.com/server/docs/package-management

Atlassian. (n.d.). What is agile? Retrieved March 31, 2021, from https://www.atlassian.com/agile

Who is the OWASP® Foundation? (n.d.). Retrieved April 01, 2021, from https://owasp.org/

Appendix A

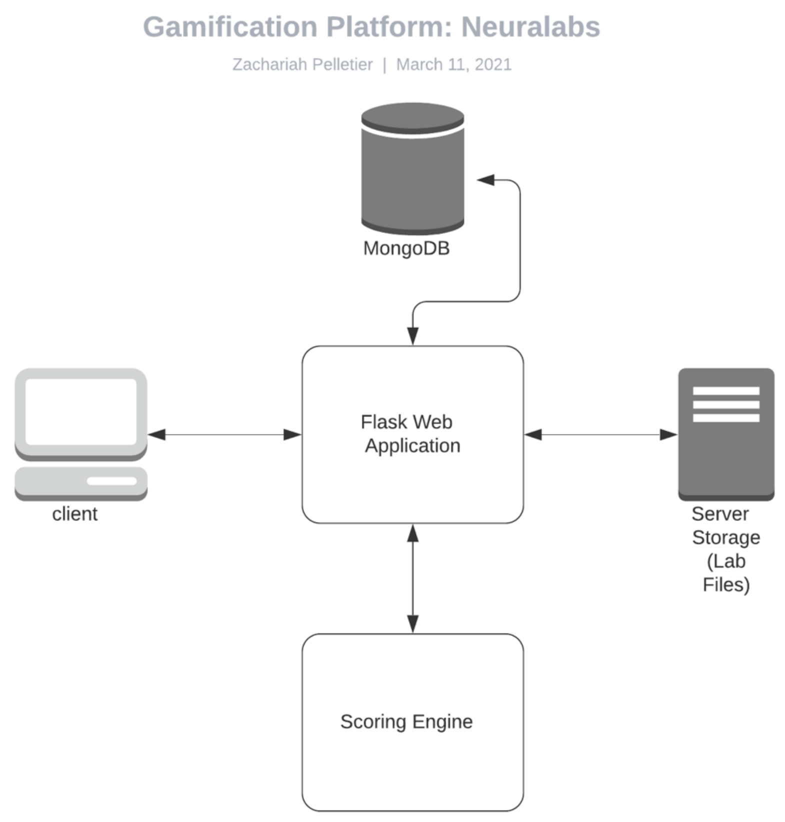
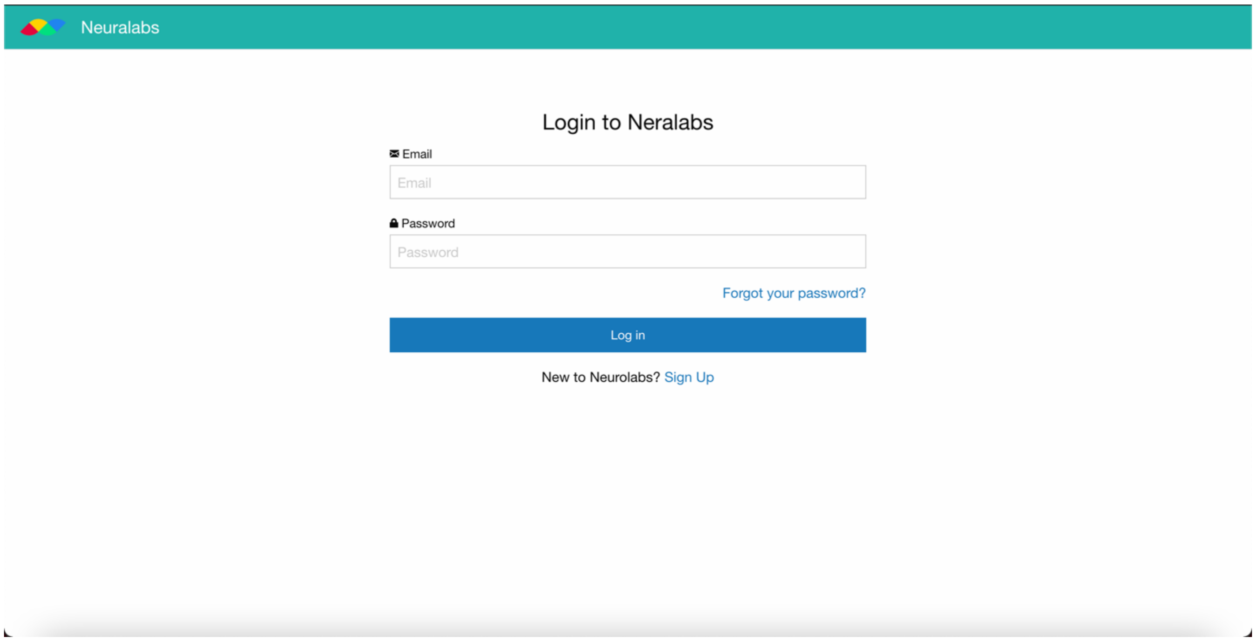

Gamification Platform: Neuralabs
Zachariah Pelletier | March 11, 2021
MongoDB
client
Flask Web Application
Server Storage (Lab Files)
Scoring Engine

Figure 1: Neuralabs Architecture Diagram


Neuralabs

Login to Neralabs

Email: Email

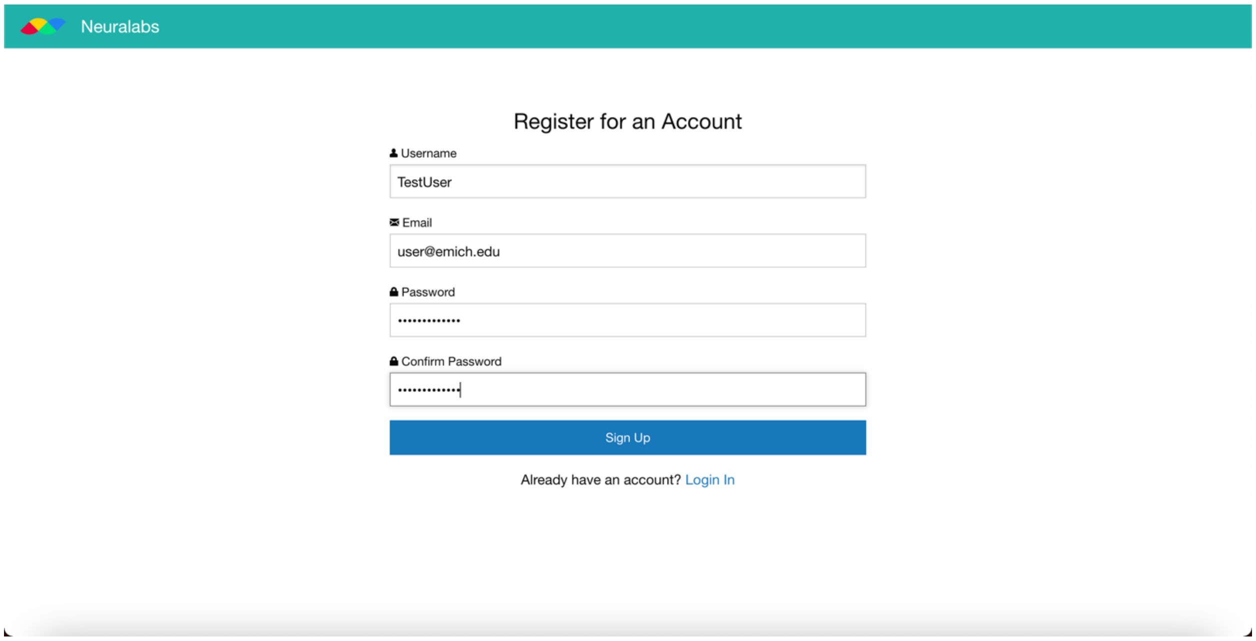
Password: Password

Forgot your password?

Log in

New to Neurolabs? Sign Up

Figure 2: User Interface - Login Page


Neuralabs

Register for an Account

Username: TestUser

Email: user@emich.edu

Password: •••••••••••••

Confirm Password: •••••••••••••

Sign Up

Already have an account? Login In

Figure 3: User Interface - Registration Page

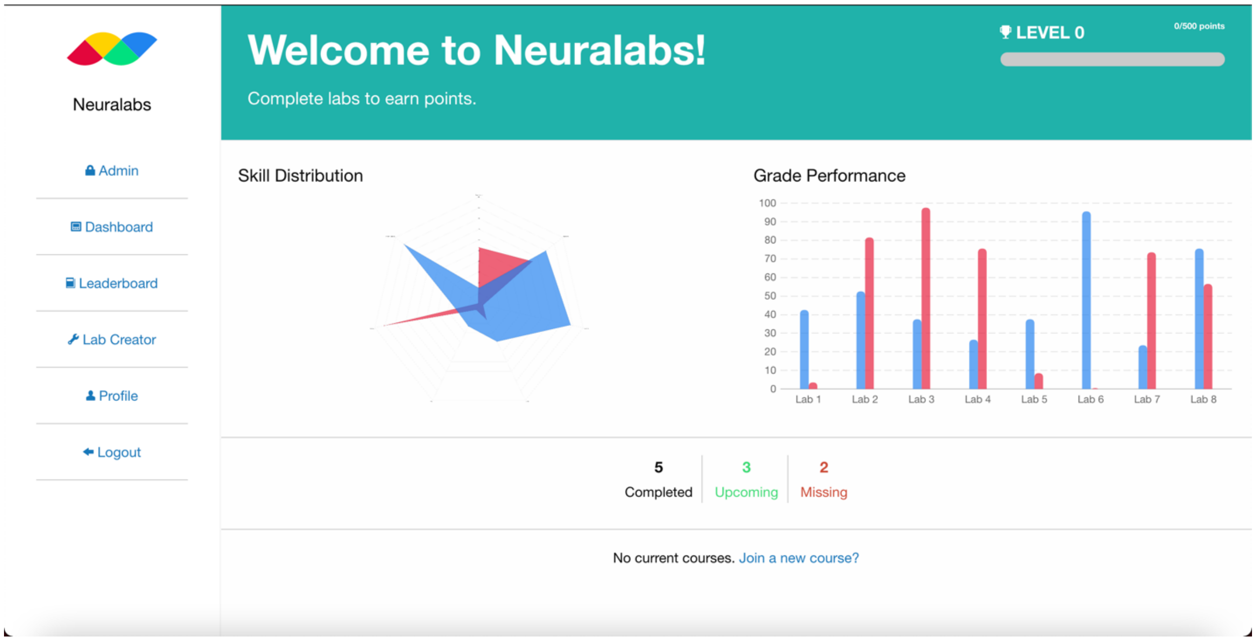
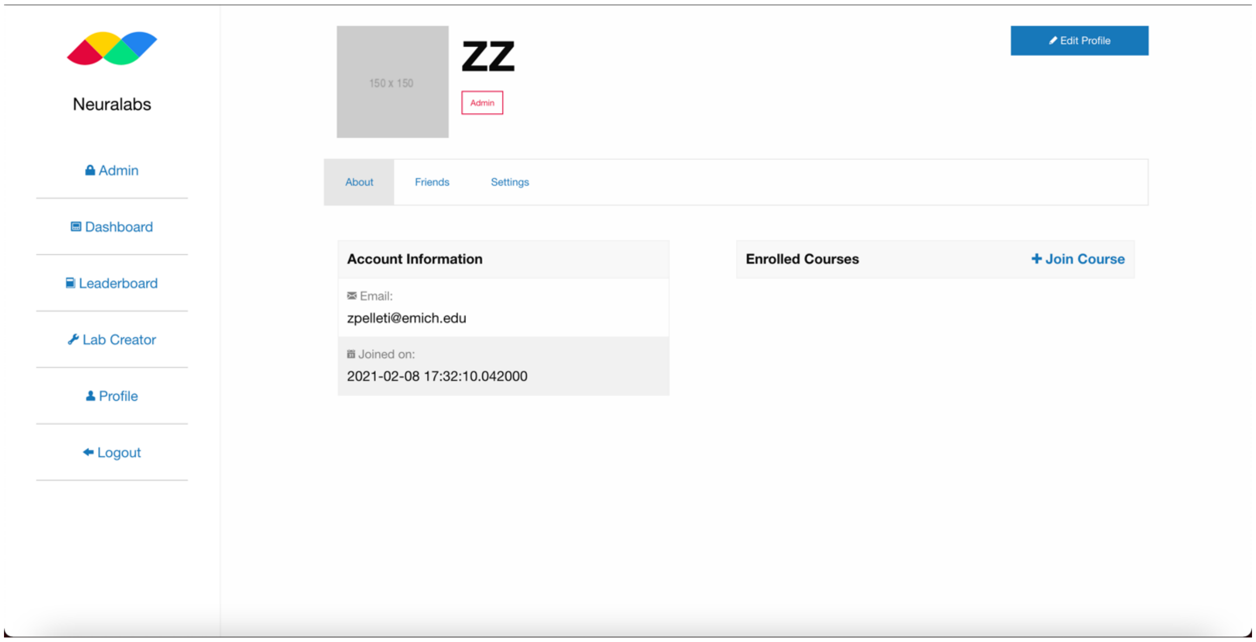


Figure 4: User Interface – Dashboard

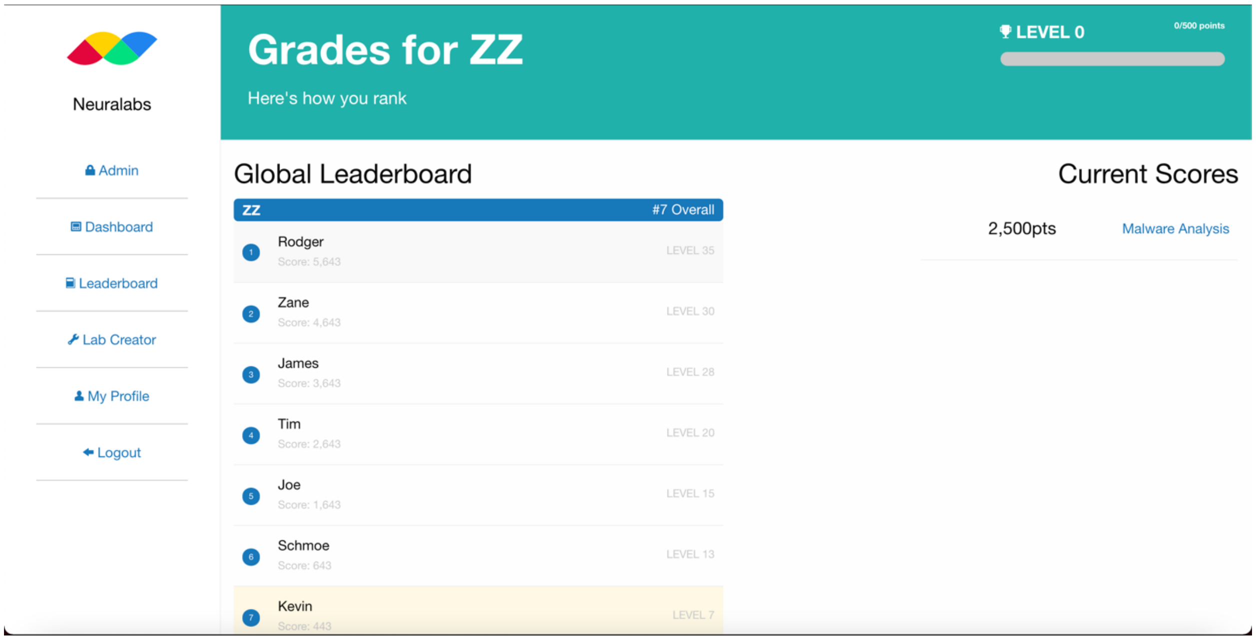


Account Information

Email:
zpelleti@emich.edu

Joined on:
2021-02-08 17:32:10.042000

Figure 5: User Interface - Profile Page



Neuralabs

Admin
Dashboard
Leaderboard
Lab Creator
My Profile
Logout

Grades for ZZ

Here's how you rank

LEVEL 0 0/500 points

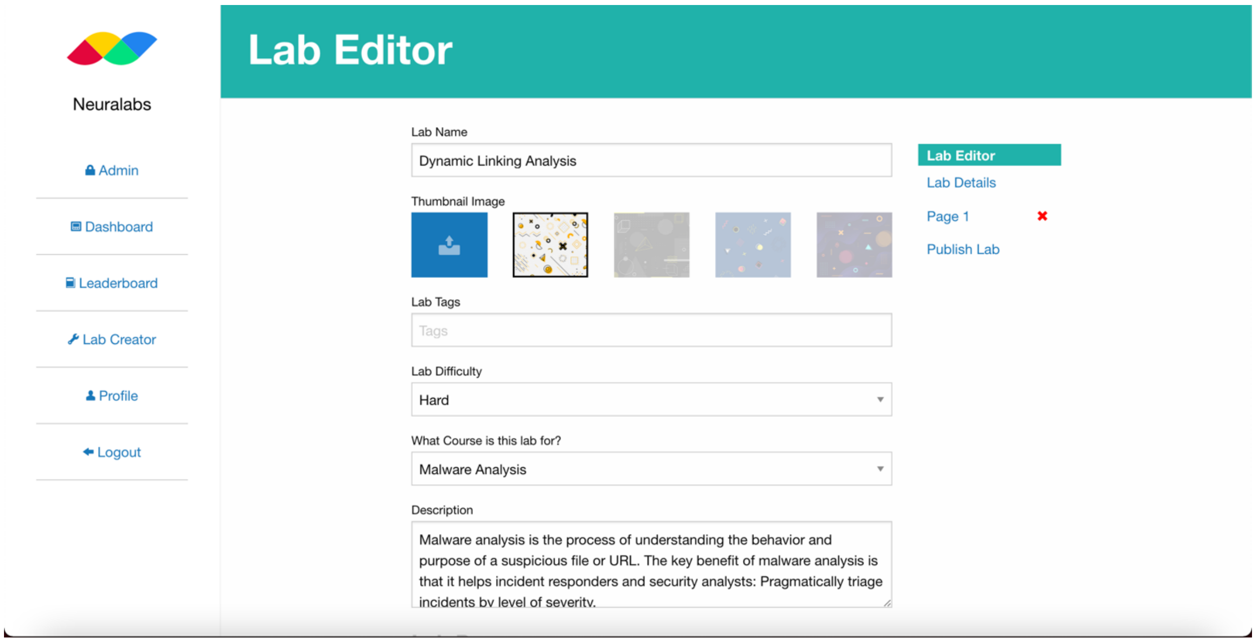
Global Leaderboard

ZZ		#7 Overall
1	Rodger Score: 5,643	LEVEL 35
2	Zane Score: 4,643	LEVEL 30
3	James Score: 3,643	LEVEL 28
4	Tim Score: 2,643	LEVEL 20
5	Joe Score: 1,643	LEVEL 15
6	Schmoe Score: 643	LEVEL 13
7	Kevin Score: 443	LEVEL 7

Current Scores

2,500pts	Malware Analysis

Figure 6: User Interface - Global Leaderboard



Neuralabs

Admin
Dashboard
Leaderboard
Lab Creator
Profile
Logout

Lab Editor

Lab Name: Dynamic Linking Analysis

Thumbnail Image

Lab Tags: Tags

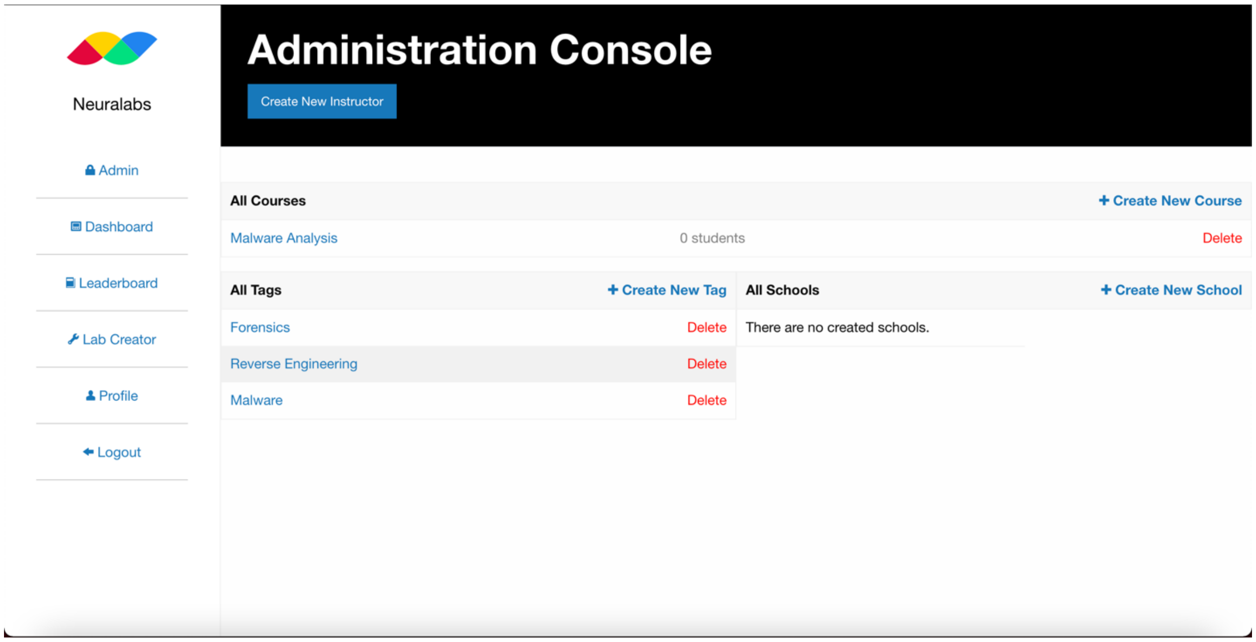
Lab Difficulty: Hard

What Course is this lab for?: Malware Analysis

Description: Malware analysis is the process of understanding the behavior and purpose of a suspicious file or URL. The key benefit of malware analysis is that it helps incident responders and security analysts: Pragmatically triage incidents by level of severity.

Lab Editor
Lab Details
Page 1 ✖
Publish Lab

Figure 7: User Interface - Lab Creator/Editor



Administration Console

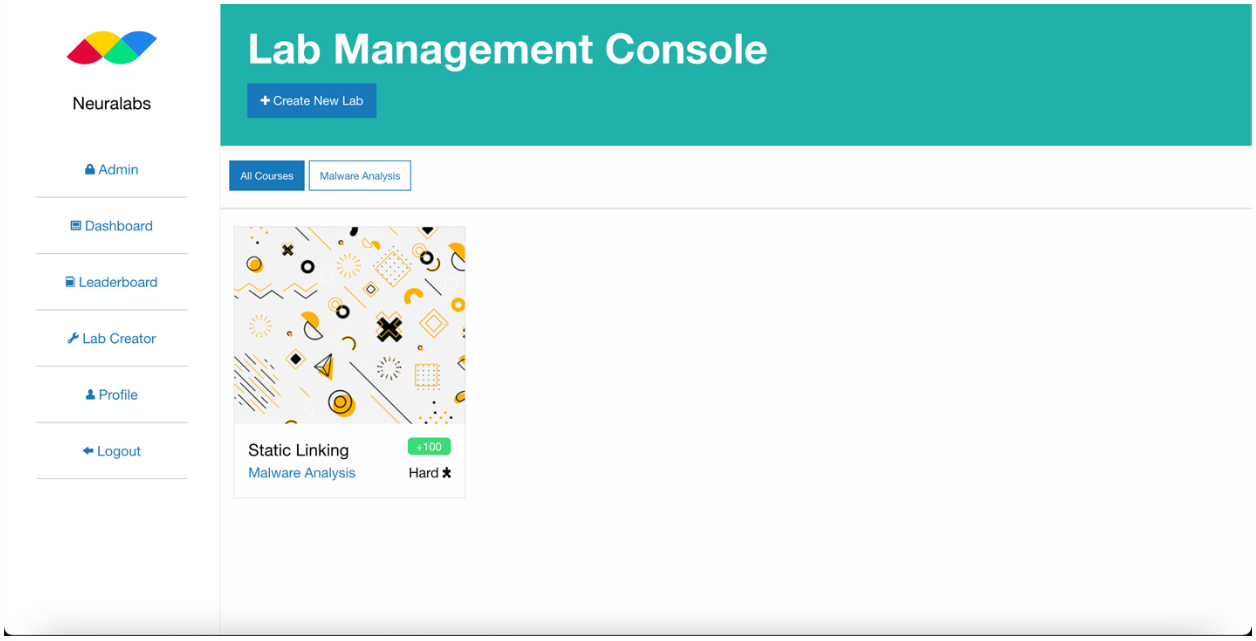
All Courses		
Malware Analysis	0 students	Delete

All Tags	
Forensics	Delete
Reverse Engineering	Delete
Malware	Delete

All Schools

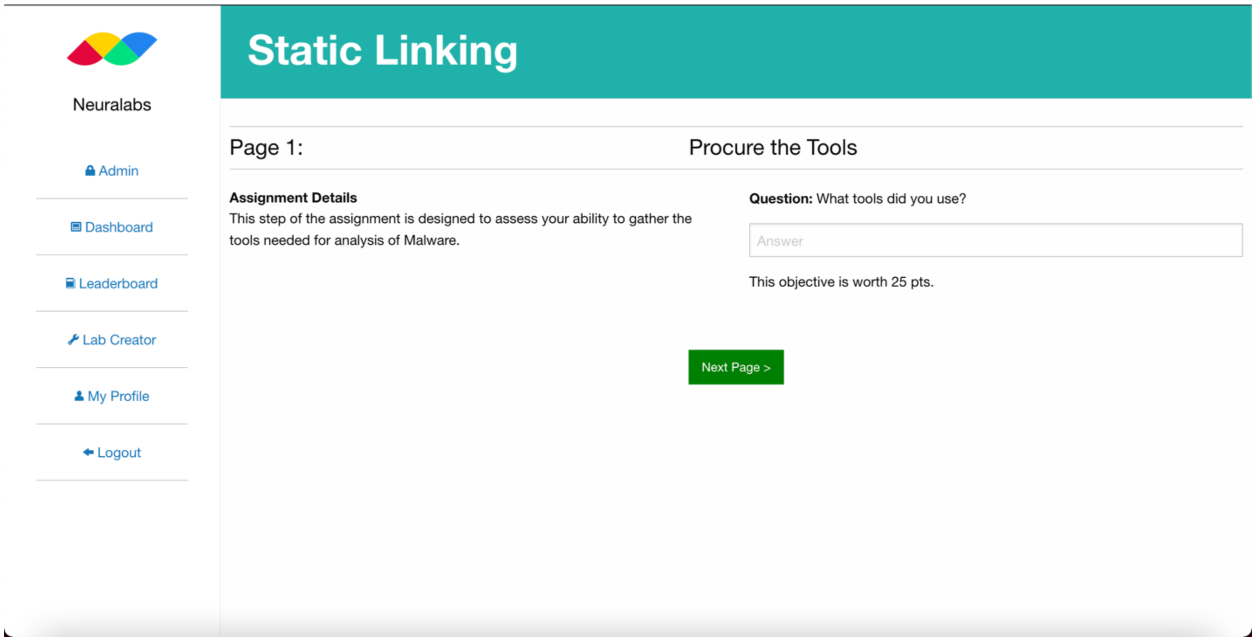
There are no created schools.

Figure 8: User Interface - Administration Console



Lab Management Console

Figure 9: User Interface - Lab Management Console



Static Linking

Page 1: Procure the Tools

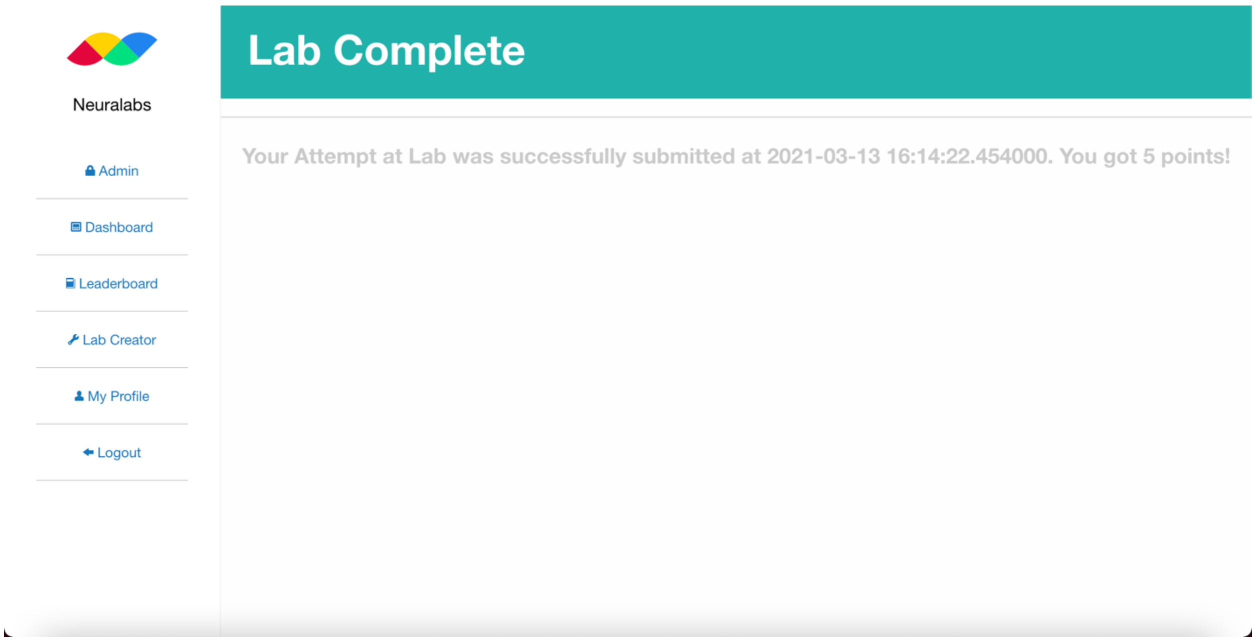
Assignment Details
This step of the assignment is designed to assess your ability to gather the tools needed for analysis of Malware.

Question: What tools did you use?

Answer

This objective is worth 25 pts.

Figure 10: User Interface - Lab Attempt Page



Lab Complete

Your Attempt at Lab was successfully submitted at 2021-03-13 16:14:22.454000. You got 5 points!

Figure 11: User Interface - Lab Complete Page

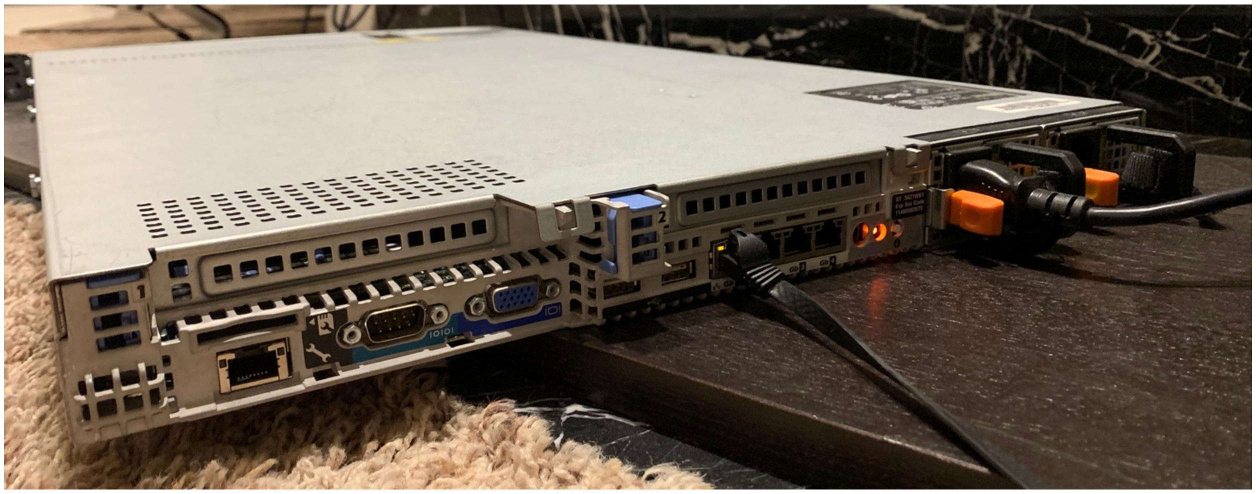


Figure 12: Server Rear View

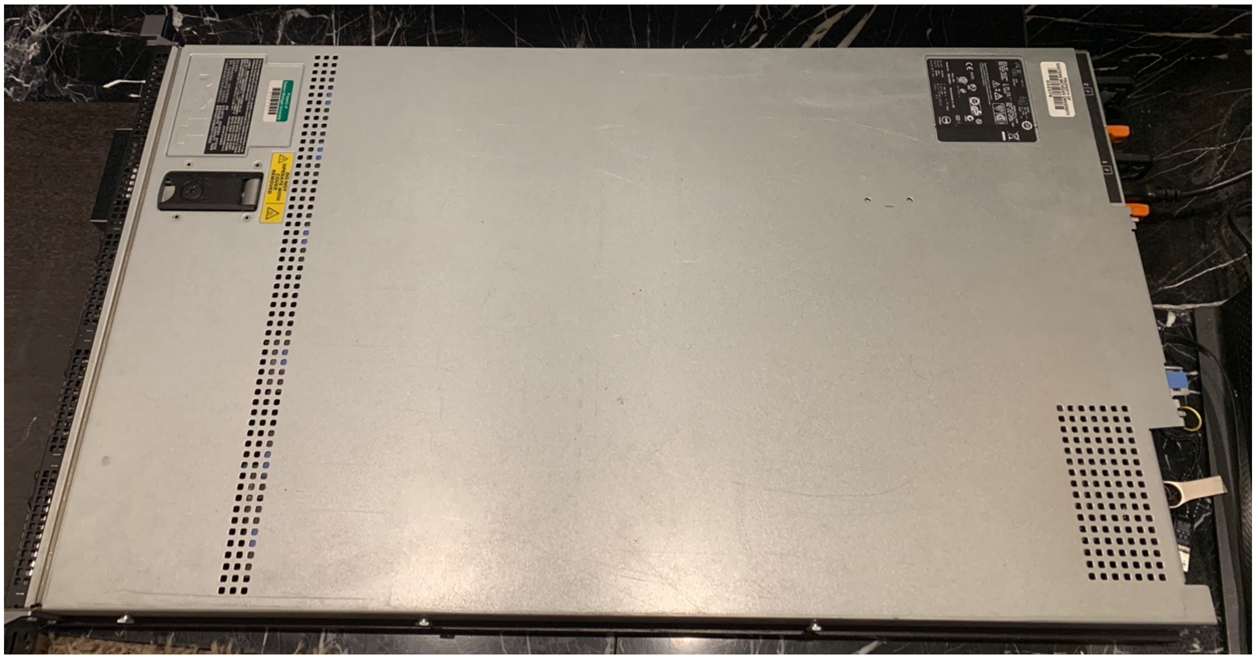


Figure 13: Server top View

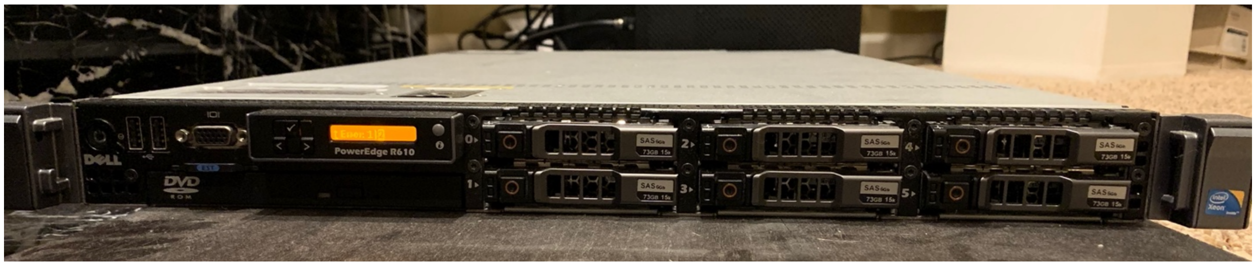
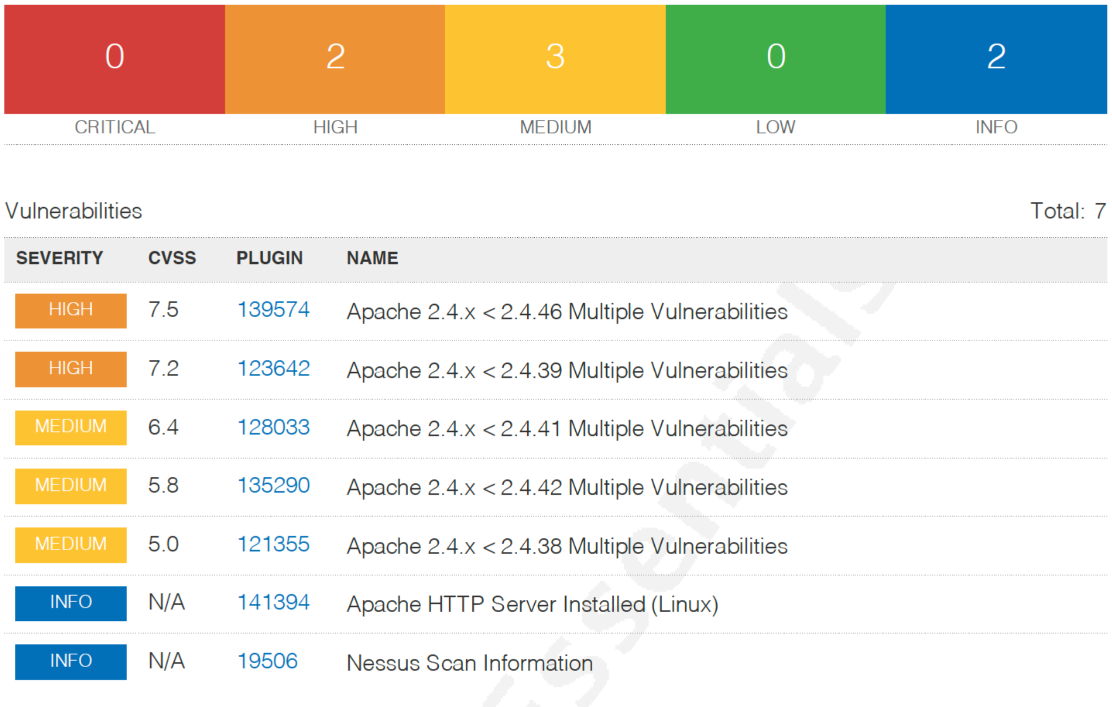


Figure 14: Server front View

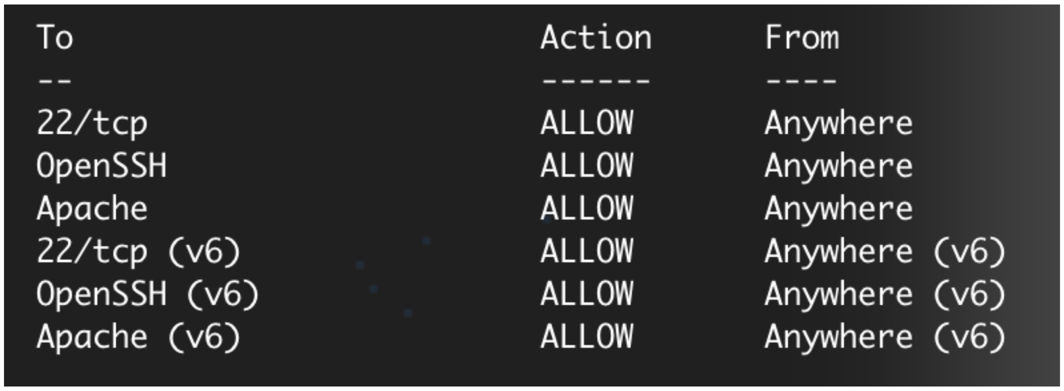


CRITICAL	HIGH	MEDIUM	LOW	INFO
0	2	3	0	2

Vulnerabilities Total: 7

SEVERITY	CVSS	PLUGIN	NAME
HIGH	7.5	139574	Apache 2.4.x < 2.4.46 Multiple Vulnerabilities
HIGH	7.2	123642	Apache 2.4.x < 2.4.39 Multiple Vulnerabilities
MEDIUM	6.4	128033	Apache 2.4.x < 2.4.41 Multiple Vulnerabilities
MEDIUM	5.8	135290	Apache 2.4.x < 2.4.42 Multiple Vulnerabilities
MEDIUM	5.0	121355	Apache 2.4.x < 2.4.38 Multiple Vulnerabilities
INFO	N/A	141394	Apache HTTP Server Installed (Linux)
INFO	N/A	19506	Nessus Scan Information

Figure 15: Example Report Generated from Nessus



```
To                        Action      From
--                        ------      ----
22/tcp                    ALLOW       Anywhere
OpenSSH                   ALLOW       Anywhere
Apache                    ALLOW       Anywhere
22/tcp (v6)               ALLOW       Anywhere (v6)
OpenSSH (v6)              ALLOW       Anywhere (v6)
Apache (v6)               ALLOW       Anywhere (v6)
```

Figure 16: Firewall Rules on the Hardened Server