Team Name: Team 3
Names: Tyler Michels, Lester Cheng, Wesley Adams, Adam Podgorny
Emails: tjmichels4455@ku.edu, l520c507@ku.edu, w809a093@ku.edu , ns41388@ku.edu

Team/GTA Meetin: Monday 1:00 - 1:15
Team Meeting day and Time: Tuesday 6:00-7:00
Contact: Adam Podgorny
Project Sponsor: KUISC www.kuisc.com , Chris Seasholtz and Ellis Springe

Project Description (150-250 words)
This project is being undertaken at the behest of the CyberPatriot group and it's sponsors to more fully encourage, develop, and support the various cybersecurity clubs and competitions across the nation. One such tool that is used frequently is a scoring engine, which allows the progress tracking of the various teams to their objectives. While solutions are in place for the Windows side of these, such competitions often involve Linux machines, where the scoring engine solutions are more sparse. Team 3's task to to rectify this by creating a scoring engine package that may be deployed for such competitions, possibly alongside the Windows versions. The end result of these endeavours will be a flexible backend package which allows for scoring teams during cybersecurity competitions in the format often used by the CyberPatriot clubs for practice. This is currently slotted to integrate with the frontend interface already used for these competitions. As we are in close and regular contact with our sponsors at KUISC, these requirements are likely to evolve as time goes on.

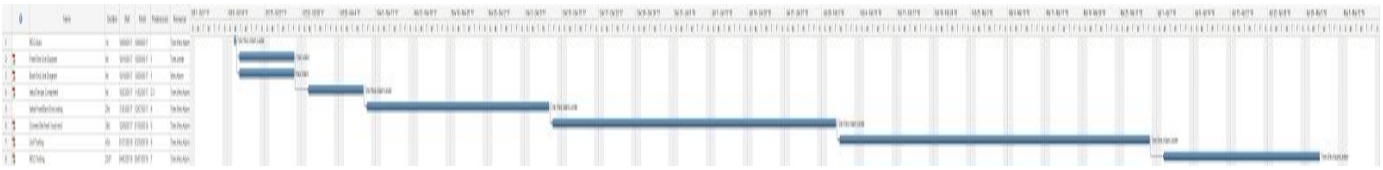
Project Milestones
Semester 1

1. Requirements Specification (10/15)
2. Front end use diagram(s) (10/22)
3. Back end use diagram(s) (10/22)
4. Initial design completed (11/1)
5. Initial front end and back end coding completed (12/7)

Semester 2

1. Front end/back end connected (01/30)
2. Unit testing (01/30 - 3/30)
3. Requirements testing (04/01 - 05/01)

Gantt Chart:



Link: https://drive.google.com/file/d/0B2PMOv9_1QHBZ2FYSERSVTR0ZU0/view?usp=sharing

Project Budget

Due to the nature of our project, we will not be requiring any monetary support for our development. Any hosting resources will be provided by the KUISC, and programming the application will be done in software that is offered for free. No special training or additional resources will be required.

Work Plan:

Tyler: Most of my focus will be on the front end part of the scoring engine, as well as some of the python features that will allow the scoring engine to interact with the different linux machines. I plan to work on a little bit of everything, but most of focus will be on the front end of the software.

Wes: Most of my focus will be on the back end part of the scoring engine. I plan on developing the the features that are actively scanning and checking the systems that are being scored.

Adam: My focus will be on an overall structuring and documentation, as well as approaching things from the Linux aspect of things. This may include hosting infrastructure and learning/deploying Metasploit and Kali to validate this. This role necessarily includes a bit of involvement in the backend of things as well, including databases is necessary.

Lester: My focus will be developing functionality to take the back-end data and make it available for the front-end. I plan to work on a bit of everything where needed as well if any other piece of the project needs an extra set of hands.

Github link:

https://github.com/0xmonkey/CS-Linux-Scoring-Engine

Preliminary Project Design

The scoring engine software that we are creating will be quite different than most other software being designed for this course. The software will be installed on its own virtual machine (likely Windows), and be in contact with a variety of different machines spread across various networks. The software will generate a web page that will be used to display the various current scores, and allow each of the teams to log in and see their team specific information. The software running on the virtual machines will keep track of all of the different scoring categories: uptime, team injects, bonus points, point deductions/penalties, etc.. These will all be displayed and updated in real time on the website that is shown to the various teams. Below, there will be a much more in depth description of the various components used/generated by this software.

To start off, it's important to understand the environment that this software will be used in and the type of people that will be utilizing this software. This scoring engine is being designed for Cyber Patriot, the national youth cyber education program, which operates on a national level and allows high school students to compete in cyber security competitions set up and

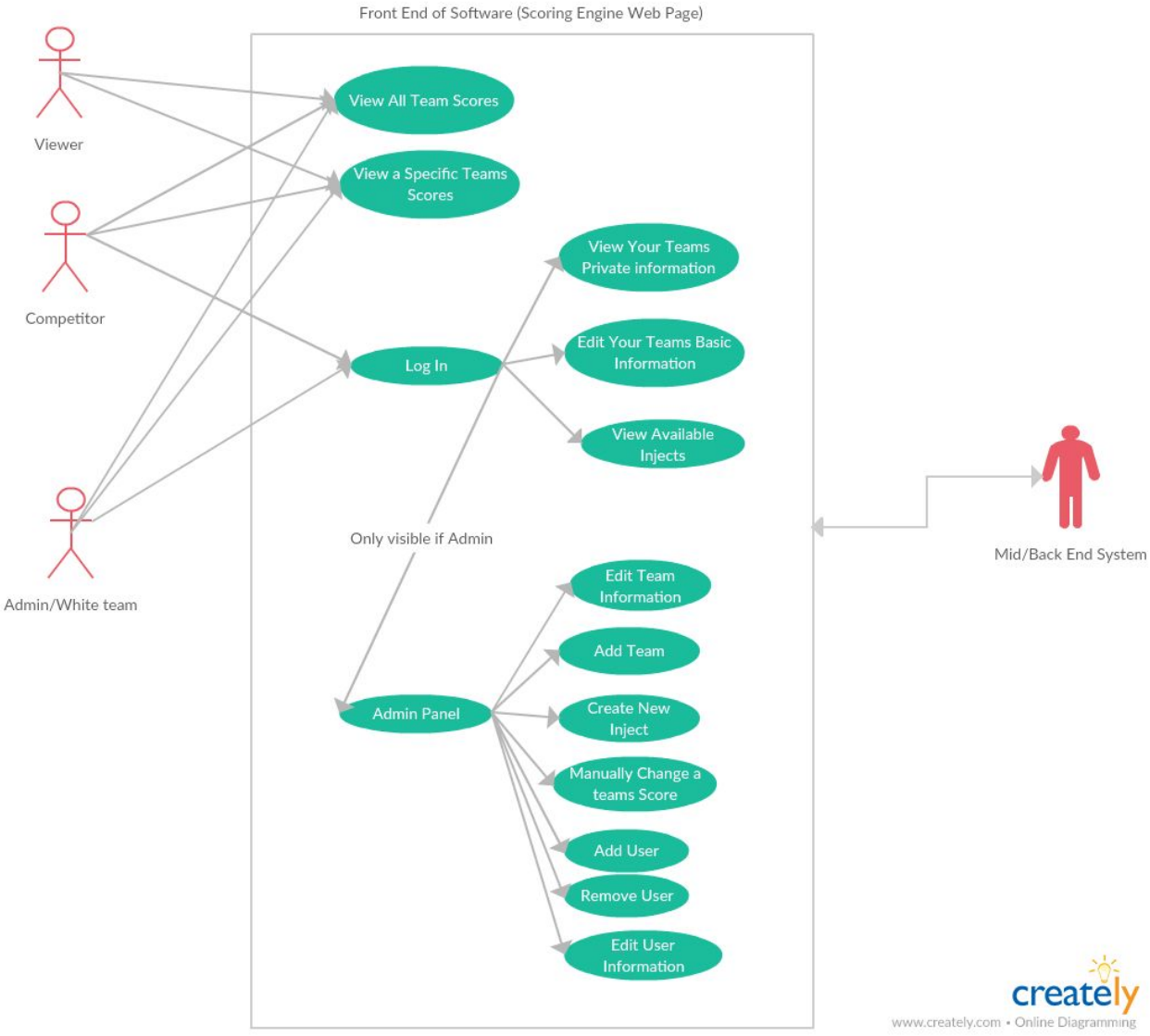
hosted by this organization. These competitions are run in a virtual environment located on a group of servers that the organization utilizes for the purpose hosting the competition. Each of the teams will be given their own virtual network, and are tasked with setting up and configuring a variety of virtual machines to the competition specifications. These machines include things like FTP servers, Web Servers, Mail servers, Firewalls or routers, and various other services that you would likely see in a corporate environment. The scoring engine is responsible for scanning each of these various services on every machine in each team's network to check if they have been configured correctly and are responsive. It should be noted that not every service should be active or open on every machine, and the scoring engine will take this into account. This check contributes to the uptime portion of the scoring engine. The scoring engine will be positioned in the virtual network such that it can communicate with all of the various teams in order to determine if their services are up and running.

The middle layer will act as a controller which takes the data output from the back end and adapts it to be more manageable and presentable to be used on the front end. The final design of this layer will be highly based on the needs of the front end and the back end. The idea of this layer is to offload work from the other two layers, allowing them to concentrate on gathering all the information we can and creating a user friendly design without having to worry about how they will interact with each other. This helps streamline the development process and makes future maintenance and support easier. This also means that should any of the back end or front end need to be changed, they can be easily replaced and plugged back into the middle layer. Additionally, by implementing a middle layer, we can more easily create unit tests for the whole project and add some security protections.

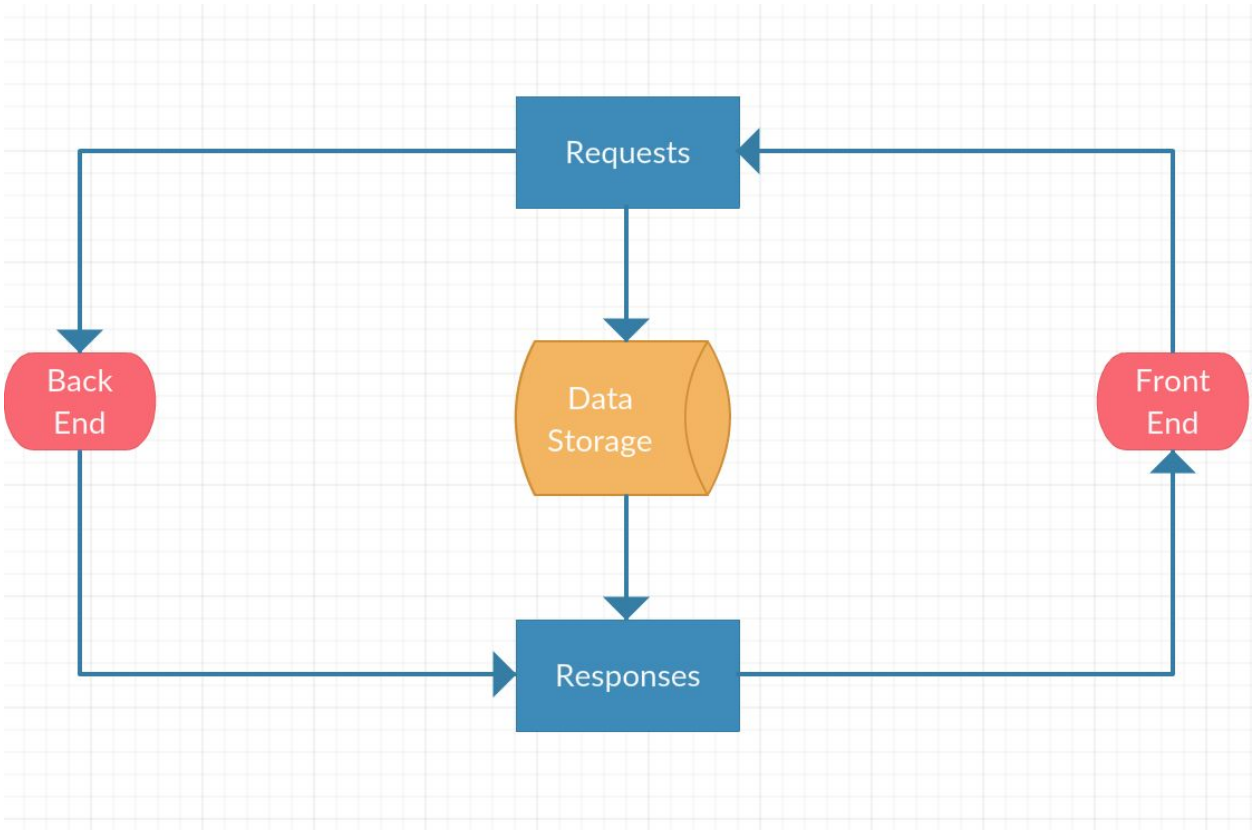
The back end will be a task based process that will run scripts in a certain increment to determine the status of the key indicators of the system which are relevant to the scores. This may be the status of an FTP daemon, the status of a ssh session, and so on. After each task is executed, it will report back to the processes central controller, which then will broadcast back to the engine middle layer for score evaluation and process. These scripts will need to be an integration of both Python and bash scripts in order to be able to work directly with the kernel if the process in question requires it. Depending on implementation, SystemD or Upstart may also be polled if necessary. The stubs for these tasks may themselves be modularly designed classes in Python, which means custom service checks could be added later as necessary. There are a variety of options for ensuring certain connections. This may be reading raw data from the ip or ss tools, or an analysis script of netstat. This extends as well to VPN addresses so long as the host controller IP is known. Along with checking the status of certain services such as FTP and SSH, the back end will also be scanning for prohibited file types - which can change from competition to competition. This can be done via bash or powershell scripts that are implemented alongside the Python system controller. Since the file types can change from competition to competition, there should be a check before the script is called to determine which file types are prohibited. This check could be incorporated into the front end and passed as a parameter to the file checking system call.

As a final note, this project is a combination effort of members of the KU Information Security Club from both the Lawrence and Edwards campuses. The Edwards campus team is only working until the end of the fall semester, while we (the Lawrence group) are working through the end of this academic year. As such, the work load has been split according to the time available to them. In an effort to maintain professionalism in our work, we have decided to keep our repositories separate until such time as it is deemed necessary to finalize the project. This step will be noted in our github repository and we should have their documentation available to view if required.

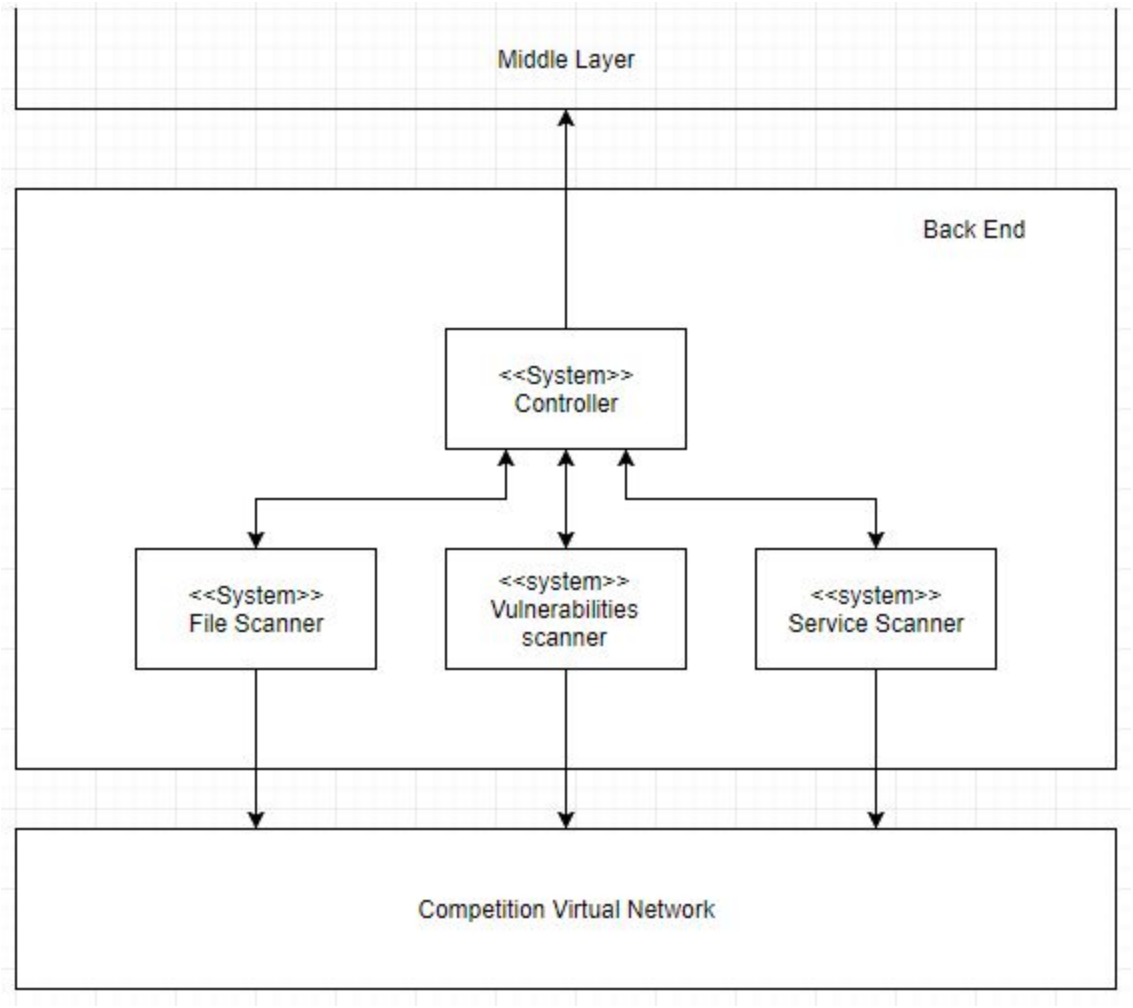
Front end use case diagram



Middle layer architecture diagram:



Back end context diagram:



Ethical and Intellectual Properties Issues:

The Ubuntu distribution used is freely available for download, and the relevant license clauses permit free use for endeavours such as this, there is little consideration to make from that particular aspect. As we are going to install our software into Ubuntu systems, and potentially use images of this setup on virtual machines, it would be a relevant concern as to whether such deployment is permissible under the Ubuntu license. We see no reason that it would not be, as it's licensure is not such that commerical use software cannot be deployed with it. As such, our use cases should certainly be acceptable.Id

As the Cyber Patriot organization will come to hold the IP rights to the software during and after development, it is theirs to decide how it will be subsequently licensed back out. As our team itself does not directly involve lawyers, we judge this consideration to be best made those who will be in charge of distribution.

Necessarily, adaptations and inspiration may be drawn from the source code of the Windows version of the components being developed. As that was voluntarily given to our team by the organization handling the scoring engines, there is no conflict in that subsequent reuse and adaptation.

From the ethics standpoint, several considerations may be made. This software is being developed as an aide to cybersecurity competitions, which in turn are educational events designed to promote learning of the information security skillset, and broadly raise awareness of that realm in general. However, as an accessory to cybersecurity training, necessarily questions of malicious use inevitably would come. As offensive exploits themselves are not being developed, the ethical consideration of potential harm as a result of this project is severely reduced. Harmful consequences that would come about a result of this software package would be limited to fringe cases of misuse, not intentional operation. As such, according to standard use cases, there are no ethical troubles inherent to this software.

Change Log:

There have been no changes made to our initial project description. We have had a clear idea and outline for our project specifications since we obtained the project idea which means our initial goals and project specifications are not going to see much change at this stage.