Team Number: 3

Names: Tyler Michels Lester Cheng Wesley Adams Adam Podgorny

Emails: tjmichels4455@ku.edu l520c507@ku.edu w809a093@ku.edu ns41388@ku.edu

Team Meeting time: Tuesday 6:00-7:00

Lab Meeting time: Monday 1:15 - 1:30

Contact: Adam Podgorny

Project Sponsor: KUISC [www.kuisc.com](http://www.kuisc.com) , Chris Seasholtz and Ellis Springe

**Project Description (150-250 words)**

* Why is the project being undertaken? Describe an opportunity or problem that the project is to address.
* What will be the end result of the project?

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. We will create a new front-end and back-end to ensure compatibility between our components and of the package on multiple operating systems. 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**

* 3-5 specific and measurable objectives per semester for first & second semester
* Completion date for each first semester milestone
* Estimated completion date for each second semester milestone
* Both implementation and documentation milestones
* Gantt Chart

*Semester 1:*

 Documentation

* Requirements Specification (10/15)
* Front end use diagram(s) (10/22)
* Back end use diagram(s) (10/22)

Implementation

* Learn python (ongoing)
* Begin back end coding (11/01/17)
* Begin front end (12/10/2017)

*Semester 2:*

 Documentation

* Final Project Design (2/4)
* Final Project Video (4/
* Final Poster/Quad Chart (5/2)

 Implementation

* Completion of back end modules (2/26)
* Completion of front end (2/26)
* Completion of middle layer (2/26)
* Completion of integration (3/12)
* System tuning (3/12)

**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:* As team lead, my main task is to be the liaison between the KU Information Security club and our senior design team. I have also helped oversee the design of the system as a whole to make sure that each layer interacts with the others in the correct fashion. For implementation I am working with Adam to develop the back end control functions that will communicate with the middle layer.

*Adam:*My focus will be on an overall structuring and documentation, as well as approaching things from the Linux aspect of things. This includes handing the backend scripts vis a vis communications between the OS via Bash, and the back-end via Python. My task is then to appropriately handle those various tasks output such that they may be sent further up the backend to be processed and interpreted down the line. If necessary to test this, I may be deploying Kali Linux against a mock system setup.

*Lester:* My focus will be developing functionality to take the back-end data and make it available for the front-end. Since my work is highly dependent on the progress of the other pieces, I will be assisting with front-end development for now.

**Github link**

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

Owner of the Github Account is Tyler Michels [tjmichels4455@ku.edu]

**Final Project Design** *(this I have pulled from the Project Proposal but it will need to be updated, specifically with regards to the front end)*

* How the software works
* 1,000-1,500 words
* 3-5 graphs, charts, or illustrations

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 scoring engine software that we are creating will be quite different than most other software being designed for this course. The software will be launched either on the host machine being set up, or a windows vm in order to check for the different items being scored for the competition. Once the settings for scoring are saved, the software will run a series of configuration commands in order to change the state of the machine, and use a webpage to display the current score of the machine. The front end of the software is being developed with java, and will use a variety of different back end tools to check if the services/policies have been changed.

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 specified increment to determine the status of the key indicators of the system which are relevant to the scores. This may be the up/down/unseen status of an FTP daemon, the status of a ssh daemon, 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. As the deployment OS used in these is Ubuntu, these scripts are keyed to the SystemD suite. Implementation has taken the form of a singular task class which behaves different based on parameters. These may then check for SystemD process status, arbitrary process execution via the ‘ps’ tool, ports status, and other details as needed. Regardless of the specifics, these are executed, and report back to the back-end controller, and this data is collated into a JSON blob, or other format via direct Java calls, and sent to the middle layer.

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 workload 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.

**Ethical Issues**

* 100-500 words per issue

*(Taken from Project Proposal)*

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 skill set, 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.

**Intellectual Property Issues**

* 100-500 words per issue

*(Taken from the Project Proposal)*

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 commercial 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.

**Change Log**

* Front end implementation tweaked due to changes in language being used for the frontend ui. Also minor changes to how software will be ran, and the context for the score checking.
* Work plans updated for all to reflect how the work has been going and is planned to continue going.