Using Open Source Intelligence to Visualize Industrial Controller Risk

Project Plan

Group:

sddec18-06

Client:

Grant Johnson

Faculty Advisor:

Manimaran Govindarasu

Team members:

Ethan Ball

Grant Foudree

Evan Kammermeier

Ryan Pals

Team email:

sddec18-06@iastate.edu

Team Website:

http://sddec18-06.sd.ece.iastate.edu/

Version 3.0

Last updated: 4/19/18

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Definitions

API: Application Programming Interface
CVE: Common Vulnerabilities and Exposures
IEEE: Institute of Electrical and Electronics Engineers
IETF: Internet Engineering Task Force
ICS: Industrial Control System
NVD: National Vulnerability Database
REST API: Representational State Transfer API (see API definition above)

1 Introductory Material

1.1 Acknowledgement

Team 06's Client: Grant Johnson
Team 06's Advisor: Manimaran Govindarasu

1.2 Problem Statement

Currently there is an abundance of information to be obtained regarding the security and vulnerability of control system devices, along with an abundance of data of publicly open and discoverable devices. For all this available information, there is no tool or way to correlate that data into something easier to use; this is the core of our project.

Our task is to create a tool which will query databases such as Shodan to see if a specific control system device is visible, it will then take that data and correlate it with information from the NVD and or the CVE database to check if that device has a known vulnerability. From this information it will run a risk assessment to see if the device is likely targeted, or possibly compromised.

1.3 Operating Environment

The operating environment for this project is the server/database we use to store the tool and other data we may use, such as a database. Therefore it should not be subject to any harsh weather conditions, and it should be properly maintained.

Data Collection Services - Linux Operating System with a Python runtime environment
Data Aggregation / Risk Likelihood Engine - Linux Operating System with a Python runtime environment and Database
User Interface - Linux Operating System, Web Hosting such as Apache Server or NGINX, User Browser (Chrome, etc)

1.4 Intended Users and Intended Uses

We are planning that the intended users for this toolset cover a wide spectrum of technical knowledge and skill sets. Therefore we must account for and design with these users in mind.

We have identified the possible users into various roles. By identifying the roles our users would play, we can better build a toolset that will accommodate their needs and uses.

These roles include:

- Researchers -- actors of moderate to advanced technical abilities interested in gathering information on possible vulnerable devices in order to find patterns and track issues towards a comprehensive solution.
- Owners -- generally nontechnical actors who are interested in seeing the security and potential vulnerabilities of their investments and devices.
- Operators of ICS devices / system administrator -- actors of moderate to advanced abilities interacting with ICS devices and are interested in their security and keeping them updated.
- Control center admin -- Nontechnical actor in charge of overseeing various systems and their reliability, would use tool to check if certain systems are likely compromised and inform superiors.
- Emergency Responders -- Technical actors watching industry sectors or territorial regions. Monitoring situational awareness to understand aggregate likelihood of an attack and the resulting impacts it would have in order to mitigate and respond.

1.5 Assumptions and Limitations

Assumptions:

- Server hardware and operating system environment system will be available through ISU ECpE and that these resources will be sufficient for the development.
- Working with the assumption that the free tiered access to Censys is enough
- It is anticipated that the NVD and CVE databases combined with ICS-CERT alerts will provide product information that can be correlated to shodan results. If varying levels of product information is discernible, the design will need to take into account a scaled likelihood based on minimal to maximal correlated device information.
- API limits will be enough for our project, though they may be a limitation

Limitations:

- How often we can query outside services (Shodan, Censys)
- Hosting limitations -- If the ISU ECpE servers are found to be insufficient, the sponsor will either arrange for different resources or modify scope to work with what is available.
- Likely not everything on Shodan will be utilized

1.6 Expected End Product and Other Deliverables

A functioning tool following the specified requirements which will determine the security posture of industrial control system devices. It will specifically answer these three questions:

- Is the device publicly visible?
- Is the device known to be vulnerable?
- Is the device likely compromised?

The final deliverable for this project will also include deployment guidelines, specifying the resources required to house it, and general use-cases of the tool. It will also include cost breakdowns for deployment based upon alternative deployment options.

We estimate completion and delivery of this project to be December 2018, the end of the Fall 2018 semester.

2 Proposed Approach and Statement of Work

2.1 Objective of the Task

The objective of our project is to make an auditing tool capable of determining the security posture of a utility company, as well as enumerating their publically-accessible devices on the internet. In order to accomplish this, we aim to design and develop a robust software program to perform device discovery and vulnerability analysis on a target network. The end product should be capable of auditing large and complex networks, as well as being usable by non-technical individuals.

2.2 Functional Requirements

FR.1: The tool can scrape online databases for vulnerabilities, such as NVD and CVE, and correlate the data with the scan results to identify vulnerable systems.

FR.2: The tool can interact with various websites, such as Shodan and Censys, to gather information on devices of interest and save the data in order to perform analysis later.

FR.3: The tool can generate an overall “risk score” to present the client with a simplistic metric on the security posture of their organization.

FR.4: The tool produces visually appealing graphs and maps representing the exposed devices and current risk of the organization.

2.3 Constraints Considerations

The tool is constrained by the limitations of the development team's access to the Shodan and Censys APIs. The current level of API access available to the team restricts the number of queries executed and amount of information that can be accessed in a given amount of time. The user-interface portion of the project must accommodate several different types of users of varying technical ability. The different categories of users who must be able to interact with the application requires that the team considers how to most effectively convey the necessary information to each group.

2.3.1 Relevant Standards and Specifications

The back end microservices of the application will have API endpoints exposed to other microservices in the system to facilitate the exchange of information between each microservice. These API endpoints will be developed and documented in accordance with the OpenAPI Specification. The OpenAPI Specification is a standardized way of describing REST APIs which is managed by the OpenAPI Initiative, a project of the Linux Foundation. The OpenAPI Specification uses definitions and standards specified in RFC 2119, RFC 6838, RFC 7231, and RFC 8174 from the IETF, so by complying with the OpenAPI standard the project is also implicitly complying with those definitions and standards. The team will uphold all aspects of the IEEE code of ethics when determining appropriate conduct for the project. In the case of any ethical concerns or uncertainties, the team will raise the subject with the project client and adviser.

2.4 Previous Work And Literature

2.4.1 Existing Tools

There are already a few tools that exist to solve this problem. The most well-known are Nessus for professional users and OpenVAS from the open source community. These tools correlate machines with known vulnerabilities like our project aims to do, they do not use intelligence sources such as Shodan and Censys; they actively scan the relevant IP ranges instead. they are active tools, while ours is passive. Our client believes that this is the key difference between our project and existing tools.

2.4.2 Relevant Literature

Our client recommended a handful of academic papers on this topic as background literature. In general, these papers do not discuss solutions, much less a particular solution similar to ours.

However, they do describe and analyze an infrastructure system riddled with vulnerabilities, and as such, they strongly support the case for the creation of this tool.

“Characterizing and Modeling Patching Practices of Industrial Control Systems” [1] describes typical patching behavior for internet exposed ICS devices. The paper outlines that even when critical vulnerabilities are exposed, patches are not always applied to the affected devices in a timely manner, further supporting the need for a tool capable of alerting ICS device operators to existing and newly published security vulnerabilities for their devices.

“An Internet-wide view of ICS devices” [2] and “Characterizing industrial control system devices on the Internet” [3] describe different ways to identify ICS devices on the internet. This information will be helpful for determining what results collected from Shodan and Censys are legitimate ICS devices and what results are false positives. Being able to accurately discern between ICS devices and other internet connected devices will be a key part of providing useful information for the end user; if too many false positives are returned then users may be overloaded with irrelevant devices, while failing to detect actual ICS devices could allow security vulnerabilities to go unnoticed.

2.5 Proposed Design

The project consists of implementing two major technical components: a backend infrastructure responsible for collecting device information, collecting vulnerability information, and correlating devices with vulnerability information, and a front end infrastructure responsible for handling user input from a web browser and displaying relevant device/vulnerability information to the user. Figure 1 contains a high level component overview of the proposed system. The backend infrastructure will be built around a microservices based architecture. This architectural pattern emphasizes implementing applications as a series of small, independent components that communicate via externally accessible APIs. The back end infrastructure will consist of five microservices: a microservice to collect devices from Shodan, a microservice to collect devices from Censys, a microservice to collect vulnerabilities from the NVD database, a microservice to collect vulnerabilities from the CVE database, and a correlation engine microservice to match devices with applicable vulnerabilities. Each microservice runs as its own independent process and will communicate with other microservices in the infrastructure via calls to externally facing interfaces.

The Shodan device collection and Censys device collection microservices are each responsible for going to their respective information sources online, querying that information source’s public API for potential devices of interest, parsing the results, and storing relevant devices in the database. The NVD and CVE vulnerability collection microservices take a similar approach, going out to their online sources of vulnerability information, collecting and parsing any new vulnerabiles, and storing that information in the database. Finally, the correlation engine microservice will match devices stored in the database with applicable vulnerabilities in the database in order to allow the front end interface to display relevant security vulnerabilities for the user’s devices. The correlation engine microservice will also be responsible for assigning a quantitative ranking of how critical the security situation for a given device is. Many different contextual factors like the number of applicable security vulnerabilities for a device, how recently that device was observed by Shodan or Censys, how long the device has been vulnerable, and other information points will be used by the correlation engine to calculate a unique ranking for each device, with a higher ranking indicating a more severe security situation.

The front end user interface for the project will be an Angular application accessed via a web browser. Angular was chosen over competing web application frameworks due to the fact that it is developed by Google, giving the team confidence that Angular will continue to be maintained in the future, and also due to the fact that Angular is widely used in the software engineering industry, increasing the amount of available support material and documentation that the team can leverage. The back end microservices will be built with Django and Celery, which were again chosen due to the fact that they are widely used in the software engineering industry. Prior team experience using Django also contributed to the team’s decision to choose this backend technology.

One possible design alternative for implementing the backend would be to create the infrastructure as a single, monolithic application. While it would be possible to utilize multithreading in a monolithic application to ensure that time intensive processes like requesting large amounts of data from external information sources do not block the main thread of the application, many of the other benefits provided by a microservices based architecture would not be realized. For example, the modular nature of a microservices based approach allows for the development of a new microservice that interacts with the rest of the system without requiring the modification of any existing microservices. As long as the new microservice interacts with the existing microservices in accordance with their public interfaces, no changes are needed in any of the existing components. To make a similar change in a monolithic architecture where all of the project’s source code is contained within a single application requires a more extensive knowledge of the implementation details of the whole system (increasing initial training time for new developers and making each new change more time consuming). Other benefits that a microservices architecture offers include faster time to production as each independent microservice can be deployed as soon as it is ready, and a more robust deployment environment as a critical exception in a microservice will only ever directly impact the single component where the error occurred. Therefore, it was decided to choose the microservices based architecture over the more traditional monolithic software architecture.

2.6 Technology Considerations

Building the project around a microservices architecture offers a number of strengths compared to a more traditional software architecture based on a single monolithic application. Utilizing a microservices architecture allows for a single microservice within the entire application to be restarted, modified, or removed from the overall system without impacting the operation of the other microservices. This gives the project an enhanced reliability in deployment over projects based on other architectural patterns, as an error in a single microservice will not directly impact other microservices. The microservices architecture also gives the project the advantage of being able to be modified and extended more easily in the future. Due to the modular, compartmentalized nature of the components inherent to a microservices based design, it will be easier to add in a new modular component or modify an existing modular component in the future than it would be to modify a monolithic application based on a different software architecture pattern.

Using Django to construct the back end components and Angular to construct the front end components allows the team to leverage tools which are widely utilized in the software engineering industry and which have fully featured, robust code bases. Building the project on these widely used technologies which will be continued to be supported by the vendor into the future ensures that there will be enough developers possessing the requisite skills to be able to take over the project and that the project will not be forced to migrate off of a deprecated technology platform in the near future. These technologies are also available to the group free of charge, allowing the project to be delivered to the client at the lowest possible cost.

The primary competition from existing products in the marketplace consists of vulnerability scanners such as Nessus and Rapid 7. These products already perform the task of scanning devices and matching the results with vulnerability databases. Their strengths consist of highly-accurate results, good reporting capabilities, real-time information, and high usability.

In addition to full-fledged scanner suites, other existing tools like Nmap have the capability of detecting the software and version running on a specific port of a device. However, one downside of Nmap is that it does not perform vulnerability correlations, which could introduce the chance that an ICS device operator is unaware of a published security vulnerability applicable to a device used by the operator.

2.7 Safety Considerations

Since we are working with utility companies who potentially have devices that control physical systems on the internet, we have to be careful not to disrupt them during our analysis. If we cause them to break somehow this could result in mechanical damage or even loss of life.

2.8 Task Approach

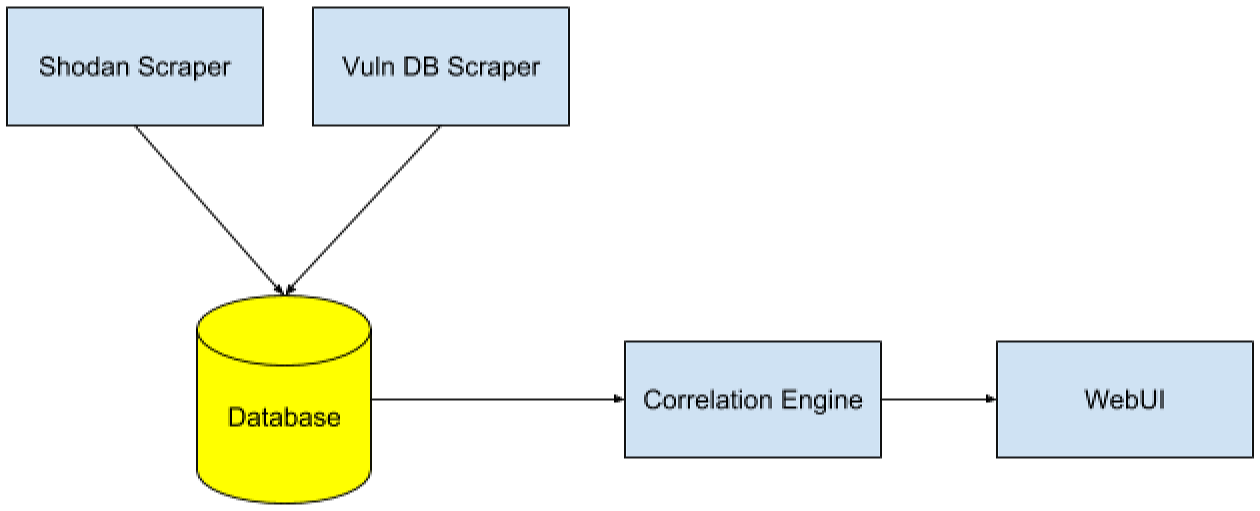

Figure 1: Task Approach Diagram

Similar to section 2.5, we plan to use four separate “microservices” and one centralized database to accomplish the task. The database is fed data from Shodan and vulnerability databases. This data is then interpreted by the correlation engine and the results are displayed via a web user interface.

2.9 Possible Risks And Risk Management

Title: API Limitations
Risk: Mitigate
Information: We are concerned with the API limits Shodan provides us with and hope that the limitations per day will not affect our testing and development. Another concern is that some of the vulnerability databases will not have a good API to interact with their data causing us to manually parse things out. This is dangerous because if they choose to change their format in the future, it will break our tool until an update can be released. Another concern is that services like Shodan do not update often enough to get good data or they do not discover all the devices. Since these services run scans at unpredictable times, some of the information can be weeks old which is not good.
Mitigation Action: In order to mitigate this risk we can rely on other sources in addition to our primary data source, using a combination of the data returned by both as our final source. This provides redundancy in our application, mitigating the risk of APIs changing or query limits.

Title: API Limitations
Risk: Accept
Information: The biggest concern is that simply getting a device’s version number is highly inaccurate for determining if a device is “vulnerable”. Distributions like Debian Linux do not

update version numbers every time an update is released, rather they engage in a backporting process in which security patches are applied but nothing else. Therefore, it is not uncommon to think something is “vulnerable” because of the version number when in fact it is not.
Mitigation Action: I am not sure what can realistically be done about this and remain a passive tool.

2.10 Project Proposed Milestones and Evaluation Criteria

Some of they key milestones in the project are getting a list of devices given an IP range from Shodan/Censys. We will verify this by checking the results from our code and comparing it to the results on the website. A second milestone is getting the version numbers for the services to match up with the vulnerability databases and find vulnerable devices. We will test this by inserting a version number that is known to be vulnerable in our scan results and checking if our logic performs the correct correlation.

2.11 Project Tracking Procedures

We plan to use Gitlab to track our progress with Gitlab Issues, as well as progress charts. We think that Gitlab issues will be a great tool as it allows us to assign tasks and deadlines. In addition to Gitlab, we also will meet bi-weekly to discuss the project’s progress both as a team, and with the client.

2.12 Expected Results and Validation

The desired outcome is that we have a functioning tool that meets the objectives outlined in section 2.1.

To confirm that our solution works at a high level, it would be nice to add a vulnerable device to the internet in the target IP range and confirm that the device shows up as vulnerable and red on the map frontend. All of the scanning and correlations should take place without interaction and all the user should notice is that a new vulnerable device popped up.

We will validate our design by completing the test plan and cases documented below.

2.13 Test Plan

FR.1: The tool can scrape online databases for vulnerabilities, such as NVD and CVE, and correlate the data with the scan results to identify vulnerable systems.

> **Test case:** For this requirement we want to test if the tool is able to gather data from vulnerability feeds/databases and make sure it correlates version numbers with possible vulnerabilities by inserting test data to check.

Test Steps:

1. Use what information we have on devices found to search vulnerability databases to see if any devices have known vulnerabilities.
2. Correlate any results from the vulnerability search with the devices.

Expected Results:
A list of separate, distinct devices with possible vulnerabilities.

FR.2: The tool can interact with various websites, such as Shodan and Censys, to gather information on devices of interest and save the data in order to perform analysis later.

Test case: Give the tool a IP range and have it gather information via Shodan/Censys, making sure all of the hosts complete successfully.

Test steps:

1. Use the tool to gather device information from sources (Shodan & Censys) within the given IP range.
2. Ensure devices discovered are relevant ICS devices.
3. Save these devices for correlation use.

Expected Results:
A list of devices which are visible to the internet and which have information relevant to an ICS device.

FR.3: The tool can generate an overall “risk score” to present the client with a simplistic metric on the security posture of their organization.

Test case: Add a couple devices to the database with varying levels of vulnerabilities and test to see if the risk score outputted by the tool reflects the severity of the vulnerabilities. Devices with a high vulnerability criticality should have a correspondingly high risk score.

Test steps:

1. Add mock devices to the database with varying vulnerabilities.
2. Use what information we have on devices to search vulnerability databases to see if any devices have known vulnerabilities.
3. Correlate any results from the vulnerability search with the devices, ranking them with a ‘vulnerability score’ based upon what information we can get from the device, and what vulnerability it has.

Expected Results:
Devices will have a correlation vulnerability score pertaining to the severity of their vulnerability tested.

FR.4: The tool produces visually appealing graphs and maps representing the exposed devices and current risk of the organization.

Test case: Inserting vulnerable devices into the database should result in the devices showing up in the user interface, as well as the tactical map in an organized, user-friendly, and visually appealing way.

Test steps:

1. Get devices results from the database.
2. Display these devices in a well organized and appealing format to show the results.
3. Links to outside sources (such as vulnerability sources) should be functional and accurate.

Expected Results:

Displayed devices within a given IP range showing the device, vulnerability score, and possible location.

Each of the tests listed are manual tests designed to ensure proper functionality. They would be conducted by using the tool itself to ensure expected output results are displayed when given a test input. Unit testing will be used on a small-time basis for the separate development of the frontend and backend as the project is in development.

3 Project Timeline, Estimated Resources, and Challenges

3.1 Project Timeline

Project development will be divided into four distinct phases, each corresponding to a new set of deliverables which answers one of the four questions specified in section 1.6: is the device publicly visible, is the device known to be vulnerable, has it been targeted, and is the device likely compromised. Each phase will consist of a number of two week agile sprints.

The phasing system allows for the team to focus on developing the most fundamental functionality required of a minimum viable product for the project before moving on to implement extra functionality which adds further features for the users. Specifically, phases 1 and 2 will implement core functionality related to identifying publicly accessible devices that are known to be vulnerable, while phases 3 and 4 add extra contextual information which can assist in decision making related to and long term management of ICS devices.

3.1.1 Phases of Project Development

The following phases of project development will be used to guide project planning activities and measure the current state of the project relative the the schedule:

Phase Number	Description of Work	Estimated Number of Weeks	Estimated Number of Agile Sprints	Comments
Phase 1	Implementing functionality to allow users to answer the question “Is the device publicly visible?” ● A back end microservice is constructed to scrape information from Shodan and Censys ● A database is established to store relevant device information gathered by the scraping utilities ● A front end web interface is constructed which allows the user to view a table of publicly visible devices with IP addresses within a user specified IP range. The interface will also optionally support geolocation of any displayed IP addresses if time permits implementation.	10 weeks	2 one month sprints	Extra time allotted to this phase for initial start up and training efforts
Phase 2	Implementing functionality to allow users to answer the question “Is the device known to be vulnerable?” ● A back end microservice is created to scrape information from vulnerability databases ● The database is modified to store vulnerability information in addition to device information ● The correlation engine which interprets vulnerability and device data is created ● The front end web interface is modified to display more advanced and detailed information about detected devices. Charts, graphs, tables, and alerts will be	12 weeks	3 sprints	Anticipated to be the first phase of the Fall 2018 semester

	added to the interface to give the user more detailed information about a given device and any applicable vulnerabilities and to provide aggregate statistics.			
Phase 3	One sprint set aside for contingency time and final polish work ● Any features not able to be implemented in phases 1 - 3 can be implemented in this sprint depending on client feedback and priorities ● Optional implementation of functionality to address final question: “Has the device been targeted?” if time permits ○ Investigate possible sources of information like IP blacklists ○ Implement necessary changes to front and back end infrastructure to support collecting and displaying targeting information ● Final documentation work carried out in anticipation of handing over the project to the client ● Finishing touches added to front and back end components to increase user experience and / or code maintainability	4 weeks	1 sprint	

Table 1: Description of different phases of project development

Please see figures 2 and 3 in the appendix for Gantt charts depicting the project schedule in the first and second semesters of the senior design course. The following milestone dates have been set for the completion of each phase:

Milestone	Date
Phase 1 Completion	Wednesday, April 25th, 2018
Phase 2 Completion	Friday, November 9th, 2018
Phase 3 Completion	Friday, December 7th, 2018

Table 2: Identification of project milestone dates

Sprint Number	**Phase Number**	**Start Date**	**End Date**
1	1	Wednesday, Feb. 28, 2018	Tuesday, Apr. 03, 2018
2	1	Wednesday, Apr. 04, 2018	Wednesday, Apr. 25, 2018
3	2	Monday, Aug. 20, 2018	Friday, Sept. 14, 2018
4	2	Monday, Sept. 17, 2018	Friday, Oct. 12, 2018
5	2	Monday, Oct. 15, 2018	Friday, Nov. 09, 2018
6	3	Monday, Nov. 12, 2018	Friday, Dec 07, 2018

Table 3: Timeline of Project Agile Sprints

Sprint Number	Front End Work to Complete	Back End Work to Complete
1	● A new Angular project has been created and can be run locally on a user's computer ● The web interface allows users to enter an IP address or a range of IP addresses to search for ● The web interface will display any entries in the database matching the user's IP address(es) ● The front end is able to query the database containing device information	● An API for all applicable backend microservices is drafted and documented ● A microservice has been created to collect and parse all entries of interest from Shodan ● A database capable of storing parsed devices from Shodan is created ● The Shodan scraper microservice is able to store parsed information in the database
2	● The Angular application is modified to query the database for both Censys and Shodan results ● The web interface of the Angular application is modified to display both Censys and Shodan results in a consistent manner ● Raw human-readable context	● A microservice has been created to collect and parse all entries of interest from Censys ● The database is modified to store entries from Censys ● The Censys scraper is able to store parsed information in the database

	information that a user could employ to determine how likely a search result is an exposed ICS device is added to the web interface ○ For example: If a specific firmware version or device model can be determined then display that with the device's results ● Unit testing has been implemented for the Angular application ● Geolocation of matching IP addresses is added to the web interface	
3	● The front end web interface of the Angular application is modified to be capable of displaying relevant NVD entries for a device ○ Note: Actual display of NVD data on the front end will not be accomplished in this sprint as the correlation engine microservice has not yet been implemented on the backend ● Aggregate statistics and metrics about all detected devices are added to the user interface in various visual formats like tables and graphs	● A microservice is created to parse and collect relevant NVD entries ● The database is modified to store NVD entries ● The NVD scraper is able to store the parsed entries in the database
4	● The Angular application is modified to query the correlation engine microservice for any applicable NVD information for the currently selected devices ● Raw human-readable information that was used by the correlation engine microservice to match an NVD entry with a device is provided when possible	● A correlation engine microservice is created to match NVD entries with any applicable detected devices ● The correlation engine can accept queries from the frontend to attempt to match any vulnerability with a given device and return results in a format consistent with the API defined in

		sprint 1
5	● The Angular application is modified to query the correlation engine microservice for any applicable CVE information for the currently selected devices ● The web interface of the Angular application is modified to display human-readable information that was used by the correlation engine microservice to match a CVE entry with a given device ● Alerts are displayed in the web interface for the user when a displayed device is matched with a sufficiently severe vulnerability ● User configurable alert settings are implemented to allow users to control how/when they are given alerts	● A microservice is created to parse and collect relevant CVE entries ● The database is modified to be able to store CVE information ● The CVE scraper is storing information in the database ● The correlation engine microservice is modified to consider and return CVE information stored in the database when matching devices with potential vulnerabilities
6	● Documentation in source files and on the wiki is checked to ensure accuracy and comprehensiveness ● Any key functionality not implemented in sprints 1 - 5 is implemented ● Final finishing touches and polish are applied to the frontend ● The web interface of the Angular application is modified to display any external indicators of device compromise like presence of a specific IP address on a blacklist of known spam addresses	● Documentation in source files and on the wiki is checked to ensure accuracy and comprehensiveness ● Any key functionality not implemented in sprints 1 - 5 is implemented ● Final finishing touches and polish are applied to the backend ● Any necessary microservices are created to pull information indicating targeting of a device and store that information in the database ● The database is modified to store this new targeting information

3.2 Feasibility Assessment

The project will involve combining data from a variety of sources, processing the data, storing it efficiently, and then displaying it to the user in a web interface. There exists the risk that integrating one or more of these separate components into a full stack application providing all of the functional requirements identified in section 2.2 will not work as anticipated. Care must be taken in designing and implementing the various public interfaces of each module to ensure the modular components can be successfully integrated. The use of existing popular application frameworks like Angular and popular application development architectures like the microservices architecture will help to contribute to project feasibility by ensuring that development is carried out with industry standard tools according to industry standard practices.

There also exist multiple anticipated challenges in phases 3 and 4 of the project. The objective of phase 3 is to implement functionality to determine if a given device has been targeted by malicious actors. The exact sources of data to use for making this determination have not yet been decided upon. Finding comprehensive, timely sources of data which can be effectively accessed by the back end data gathering modules may prove to be to be difficult, rendering phase 3 unable to be implemented in the currently scheduled 4 week implementation window. Phase 4 of the project is concerned with providing an assessment of how likely a given device is to have been compromised. This assessment may have to be based largely on the context-sensitive targeting information gathered in phase 3. If the role of context-sensitive information is large, it may be difficult to implement logic which can automatically generate some sort of relative ranking or quantification of how vulnerable a device is. The application may have to either ignore this context-sensitive information, providing potentially less accurate assessments to the user, or display all of this information to the user for manual analysis, increasing workloads on the user.

3.3 Personnel Effort Requirements

The following development activities will be carried out in the four phases of the project:

Phase 1 Task	**Number of Developers**	**Estimated Development Hours / Developer**	**Documentation Hours / Developer**	**Total Task Time (Hrs.)**
1.1) Creation of back end microservice to collect information from Shodan	2	40	2	84
1.2) Creation of back end microservice to collect information from Censys	2	40	2	84
1.3) Creation of database to store information gathered in tasks 1.1 and 1.2	2	32	2	68
1.4) Creation of web interface to query and display information stored in database created in task 1.3	2	32	2	68

Phase 1 Total				**304**
Phase 2 Task	**Number of Developers**	**Estimated Development Hours / Developer**	**Documentation Hours / Developer**	**Total Task Time (Hrs.)**
2.1) Creation of back end microservice to collect information from vulnerability databases	2	32	2	68
2.2) Creation of back end microservice to correlate publicly accessible devices identified on Shodan and/or Censys with known vulnerabilities collected in task 2.1	2	40	2	84
2.3) Modification of the database database created in task 1.3 to store information gathered in task 2.1	2	24	1	50
2.4) Modification of web interface created in task 1.4 to query the correlation engine created in task 2.2 about vulnerable devices	2	24	1	50
Phase 2 Total				**252**
Phase 3 Task	**Number of Developers**	**Estimated Development Hours / Developer**	**Documentation Hours / Developer**	**Total Task Time (Hrs.)**
3.1) Research to find appropriate sources of information which indicated malicious targeting of a device	2	14	2	32
3.2) Creation of back end microservice to store information from sources identified in task 3.1 in the database modified in task 3.3	2	23	1	48
3.3) Modification of the database database created in task 1.3 to store information gathered in task 3.2	2	7	1	16
3.4) Modification of the correlation engine created in task 2.2 to support the new information gathered by the microservice	2	7	1	16

created in task 3.2				
3.5) Modification of the front end web interface to support displaying querying the correlation engine for information collected in task 3.2	2	7	1	16
Phase 3 Total				**128**
Phase 4 Task	**Number of Developers**	**Estimated Development Hours / Developer**	**Documentation Hours / Developer**	**Total Task Time (Hrs.)**
4.1) The correlation engine is extended to produce a relative description of how likely a device has been compromised based on all sources of information stored in the database	4	24	1	100
4.2) Modification of the front end web interface to support displaying the likelihood of device being compromised assessment added to the correlation engine in task 4.1	2	11	1	24
Phase 4 Total				**124**

Table 3: Personnel effort estimations by task

3.4 Other Resource Requirements

The project will require external hosting resources to maintain the team's documentation and GitLab instance. Both of these services should be provided by the Electrical and Computer Engineering Department's Electronics and Technology group. Any virtual machines needed by the team to host various components of the project should also be able to be obtained from the Electronics and Technology group.

3.5 Financial Requirements

API access to both Censys and Shodan is offered on a tiered model where users can pay certain levels of subscription fees to gain greater access to the systems. The project is currently relying on the free tiers of access to interact with both sites. Should the free tiers prove to be inadequate, the option of buying a paid subscription may need to be explored. The exact cost can not be estimated in advance as both sites offer multiple subscription tiers at different prices.

4 Closure Materials

4.1 Conclusion

Currently, there are many ICS devices that control necessary infrastructure. Unfortunately, these devices aren’t always secured correctly. There are a few resources available that gather data about exposed systems such as Censys and Shodan. There are also some databases such as the NVD that contain known vulnerabilities which may apply to these systems.

This project aims to help operators of ICS devices better understand the cyber security risks associated with their devices.

We will achieve this by correlating exposed systems from Shodan and Censys with vulnerabilities from the NVD and CVE databases with a server running on a University provided server. Our application will use this information to assists ICS operators in making informed decisions about whether they need to improve the security of their devices, and how to go about doing this.

By following the steps outlined in this proposal, we can help improve the security and reliability of the utilities and infrastructure we all rely on in our everyday lives.

4.2 References

[1] Brandon Wang, Xiaoye Li, Leandro P. de Aguiar, Daniel S. Menasche, and Zubair Shafiq. “Characterizing and Modeling Patching Practices of Industrial Control Systems”. Proc. ACM Meas. Anal. Comput. Syst. 1, 1, Article 18. June 2017.

[2] Mirian et al, "An Internet-wide view of ICS devices," 14th Annual Conference on Privacy, Security and Trust (PST), Auckland, pp. 96-103. 2016.

[3] Xuan Feng, Qiang Li, Haining Wang and Limin Sun, "Characterizing industrial control system devices on the Internet," IEEE 24th International Conference on Network Protocols. Singapore. pp. 1-10. 2016.

4.3 Appendices

Pictured on this page: Figures 2 and 3: Gantt charts for project phases 1 - 3 and sprints 1 - 6

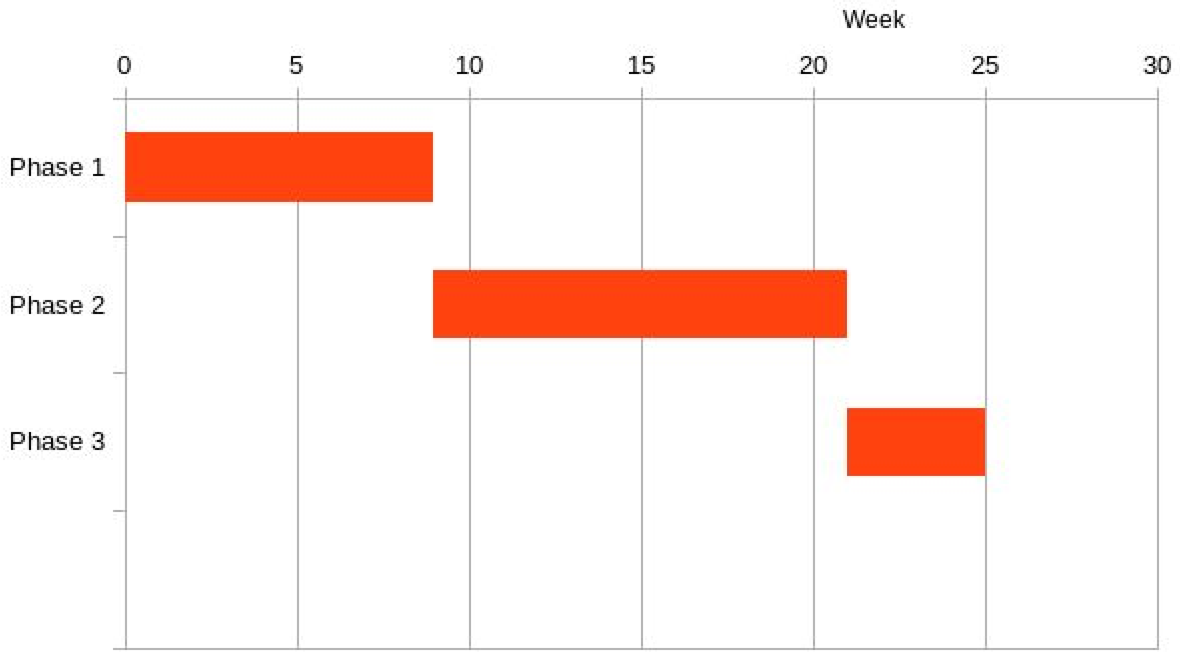

Figure 2: A phase level Gantt chart of the project timeline

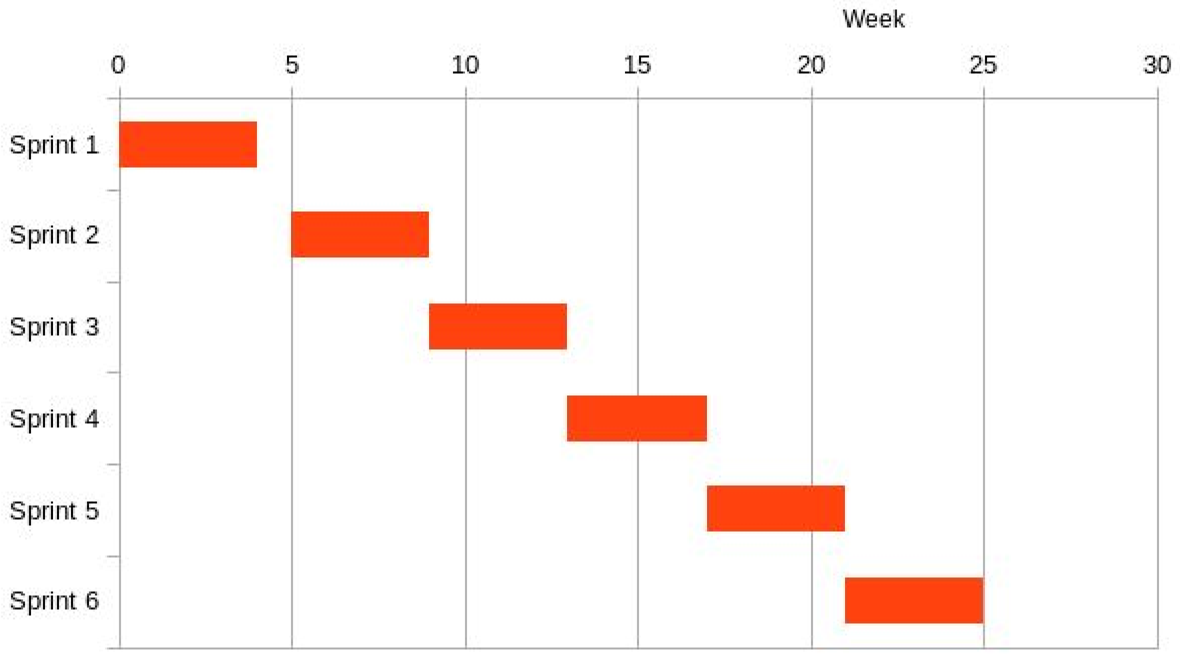

Figure 3: A sprint level Gantt chart of the project timeline