Scout: Network Asset Discovery

by

Jack Brown, Chase Hood, Joseph Hu

Submitted to

the Faculty of the School of Information Technology

in Partial Fulfillment of the Requirements for

the Degree of Bachelor of Science

in Information Technology

Jack Brown	4/12/2020 Date
Chase Hood	4/12/2020 Date
Joseph Hu	4/12/2020 Date
Bo Vykhovanyuk, Faculty Advisor	Date

University of Cincinnati

College of

Education, Criminal Justice, and Human Services

April 2020

Table of Contents

List of Illustrations

Tables:

Figures:

1.0 Abstract

Many of the top companies in the world use a SIEM to collect and monitor network events to ensure their environment is secure. Looking at the top SIEM solutions, these products struggle at having all necessary capabilities out of the box. Our team developed Scout, Scout provides a missing value which will minimizes setup in environment's and reduce security event response times, thus preventing security breaches. Scout is able to scan and discover all relevant and important asset details within an environment and integrate them directly into your companies SIEM. Rapid7 claims, "…connecting your tools and automating security processes can save up to 83 percent of the time spent manually fetching data and other triage and incident response tasks.". Staying on budget is something that all companies need to keep in mind, and our solution, Scout, will be able to save companies thousands, if not more.

1.1 Problem Statement:

Most SIEM's (Security Information and Event Management) in today's market all experience a similar issue that causes analyst to spend an excessive amount of time searching through logs and events for simple information. In an article by LogRhythm, a security intelligence company, "analysts are spending too much time trying to understand which threats are real because they're performing investigations across multiple platforms […] they're wasting valuable time and resources on manual, repetitive tasks instead of focusing on more critical activities.".

The most common issue with these SIEMs are the lack of correlation they have in environments when it comes to pinpointing their source host. Depending on what data source the log is pulled from, an analyst rarely will have all the necessary information readily available to determine which system may be infected. For example, you may get an IP, hostname, or user but never all 3 together. With IPs changing and nonconventional naming methods for hostnames this can cause analyst to spend valuable time searching through other data source logs attempting to pinpoint the user, current IP, Hostname, and MAC address to make sure they have the right system. The time wasted doing this research is valuable as every second is important in sending out alerts or acting on the compromised machine.

The Research that is needed to conquer our problem was finding a tool that would allow us to scan our internal network at work and be able to set up a pipeline to be sent over to Logstash so we could visualize our information in Elastic. We needed to learn how to code in a language that would allow us to configure an API and make changes to delete stale data and post new scans that would be operated daily.

1.2 Solution:

Our solution is to create a tool that will pull all this information into one spot, correlate it, and have it readily available for the analyst. Today's modern SIEM require the need of span ports, dedicated credentials, or privileged network access in order to discover assets such as products from AT&T's AlienVault and Sophos' Endpoint.

With Scout, we will be able to find and fingerprint hardened assets across segmented environments with a single agent to measure and access an entire enterprise that will integrated into a SIEM. To do this we will pull logs from a variety of data sources such as routers, AD server, and workstations. With these logs we will import and format them through Syslog which then feeds into our SIEM. This is where we store all our relevant information regarding the source systems. Once all IPs, MAC addresses, and hostnames are correlated correctly we will feed them back into our SIEM. This will provide analysts with accurate up to date information on which system the security event relates to minimizing the time it requires to resolve a security alert.

Our solution is seen to be better than other's because we have created a python script that allows us to easily import the API from Rumble which scans our internal network and pushes that data into Logstash where we can index each parameter as required to display to our customers and analysts. Other asset scanners will just look within a network and push the data but not remove the stale data, Scout does this but removing old data through our script and ingests the daily scans into Logstash as new information.

1.3 User Profile:

We have determined that the only End-User that will be using this tool will be system administrators or security analysts. Our User Profile table provides the potential users, software and experiences that they have had prior to using Scout.

User Profile Form
1.4 PROJECT: Scout – A network asset extraction solution
1.5 POTENTIAL USERS: - Proficient groups/individuals within security and/or networking - Non-proficient groups/individuals within security and/or networking
1.6 SOFTWARE, INTERFACE, AND RELATED EXPERIENCE: This project will be goaled towards groups or individuals who work within the realm of IT, specifically security and/or networking. The features in our project will attract clients who want the simplest way to view network and extract network assets that will be integrated to a SIEM. Related experience is based solely on how much of the network the individual wants to extract such as the whole network or subnets of a partial entity.
1.7 EXPERIENCE WITH SIMILAR APPLICATIONS: Individuals have had experience experimenting with Wireshark to look through packets and determining what IPs were associated with the information given from the log. Users may also have used OpenVAS to scan through a network and find vulnerabilities. Regardless of experience, a user will be able to utilize our project and easily discover assets that can be imported to our SIEM.
1.8 TASK EXPERIENCE: User may have had experience performing tasks such as investigating logs, configuring a network, managing access points for routers and/or modems, and doing network scans.
1.9 FREQUENCY OF USE: The setup of the product will be run once. For actual use and the purpose of a demonstration, the product can be set to run daily/weekly/monthly.
1.10 KEY PROJECT DESIGN REQUIREMENTS THAT THE USER PROFILE SUGGESTS: - Simple and Interactive UI - Responsive interface - Quick set and start - Easy network discovery

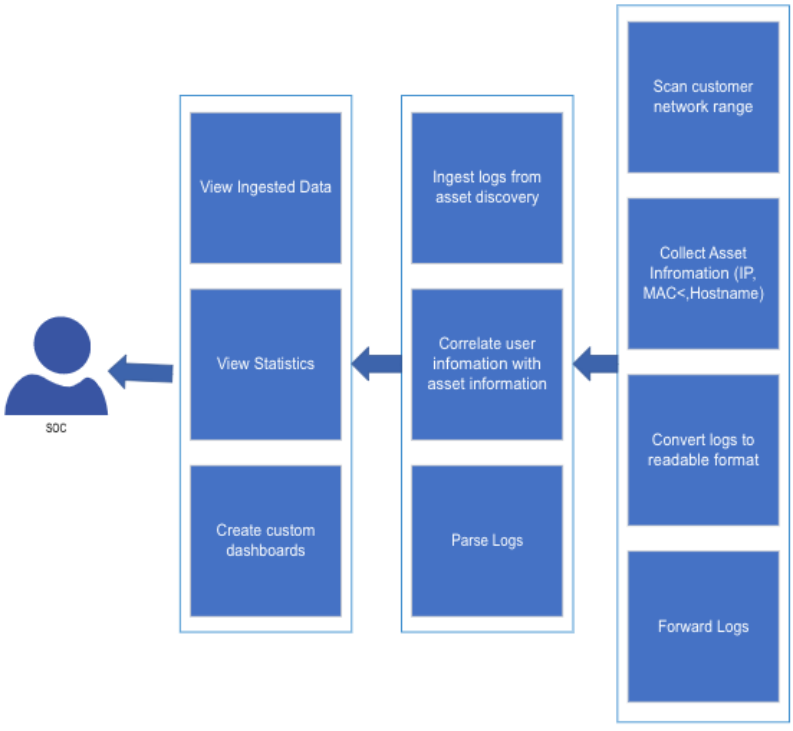
Table 1.3.1: User Profile Table

1.11 Project Name:

Scout

1.12 Figure 1: Use Case Diagram illustrates a use case diagram for Scout

There is 1 end-user that is interfacing with the system, in our case is a security analyst. This user will have the role of viewing and analyzing the data the SIEM and Asset Discovery tool provides from the backend. The interface will provide charts and graphs as well as numerical data for a user to base their analysis on.



1.12.1 Figure 1: Use Case Diagram

2.0 Project Management

2.1 Budget

A project budget is a key component of any project as it provides the expected expenses a company would be expected to spend during a project. Scout's expenses are sorted into two materials which will consist of materials and labor. To keep the cost of hardware down, the backend will be hosted through Elastic's Cloud Service. All framework and development tools to create this project are free and open source. Once we simulated the real-world example, the wage will cost $24,000 stimulated under the assumption that each team member earns $18 an hour and the values that are based around the cost of the cloud. The actual cost of this project will end up being $210 after deploying months of costs for cloud services to rent hardware power for the procurement of the Bachelor of Science degree in Information Technology.

Category	Item	Description	Expected Cost	Actual Cost
Materials	Hardware	Computer servers and devices used to deploy, test, and power the application.	$0	TBD
	Software	Tools and application frameworks that will be used to develop Scout.	$0	TBD
Labor	Conferences and Meetings	Estimated funding for employee learning events, meeting, and collaborations.	$0	TBD
	Actual Wage Costs	Estimated wages that will be distributed for development of the project.	$0	TBD
	Simulated Wage Costs*	Assuming two developers will be making ~18 an hour. The predicted wages that would be distributed if the project were done in the current market.	$24,000	TBD
TOTALS			$24,000	TBD
**Simulated Wage Costs aren't in the costs total*				

2.1.1 Table 2: Project Budget illustrates expected budget for solution in production

2.2 Schedule

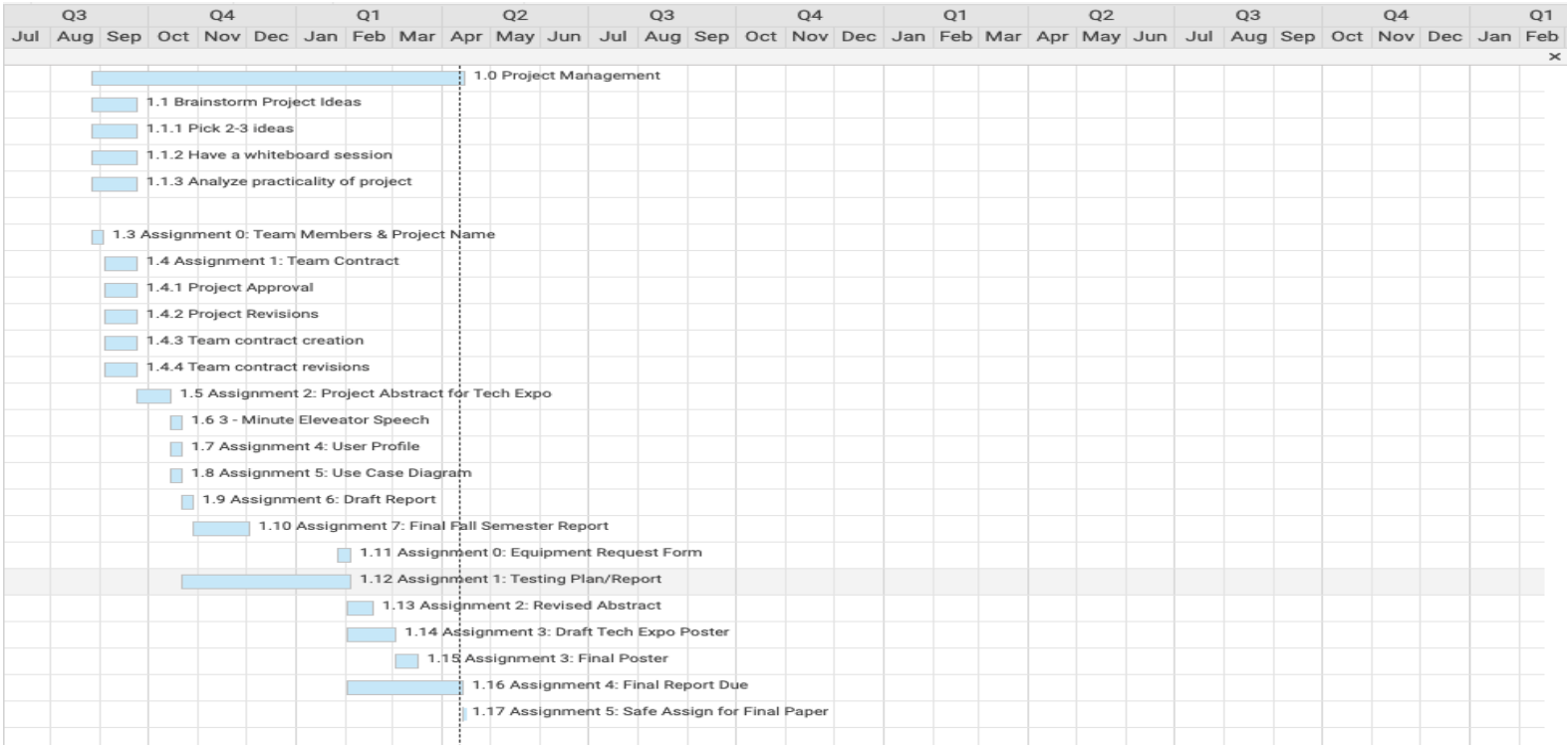
Table 3: Represents full timeline of the Scout project
Figure 2 &3: Represents full timeline of the Scout project presented in a Gantt Chart

Task	**Duration**	**Start Date**	**End Date**
1.0 Project Management	232	08/26/19	04/14/20
1.1 Brainstorm Project Ideas	28	08/26/19	09/23/19
1.1.1 Pick 2-3 ideas	28	08/26/19	09/23/19
1.1.2 Have a whiteboard session	28	08/26/19	09/23/19
1.1.3 Analyze practicality of project	28	08/26/19	09/23/19
1.2 Assign roles of project			
1.3 Assignment 0: Team Members & Project Name	7	08/26/19	09/02/19
1.4 Assignment 1: Team Contract	20	09/03/19	09/23/19
1.4.1 Project Approval	20	09/03/19	09/23/19
1.4.2 Project Revisions	20	09/03/19	09/23/19
1.4.3 Team contract creation	20	09/03/19	09/23/19
1.4.4 Team contract revisions	20	09/03/19	09/23/19
1.5 Assignment 2: Project Abstract for Tech Expo	21	09/23/19	10/14/19
1.6 3 - Minute Elevator Speech	7	10/14/19	10/21/19
1.7 Assignment 4: User Profile	7	10/14/19	10/21/19
1.8 Assignment 5: Use Case Diagram	7	10/14/19	10/21/19
1.9 Assignment 6: Draft Report	7	10/21/19	10/28/19
1.10 Assignment 7: Final Fall Semester Report	35	10/28/19	12/02/19
1.11 Assignment 0: Equipment Request Form	7	01/26/20	02/03/20
1.12 Assignment 1: Testing Plan/Report	105	10/21/19	02/03/20
1.13 Assignment 2: Revised Abstract	18	02/01/20	02/17/20
1.14 Assignment 3: Draft Tech Expo Poster	33	02/01/20	03/02/20
1.15 Assignment 3: Final Poster	14	03/02/20	03/16/20
1.16 Assignment 4: Final Report Due	72	02/01/20	04/13/20

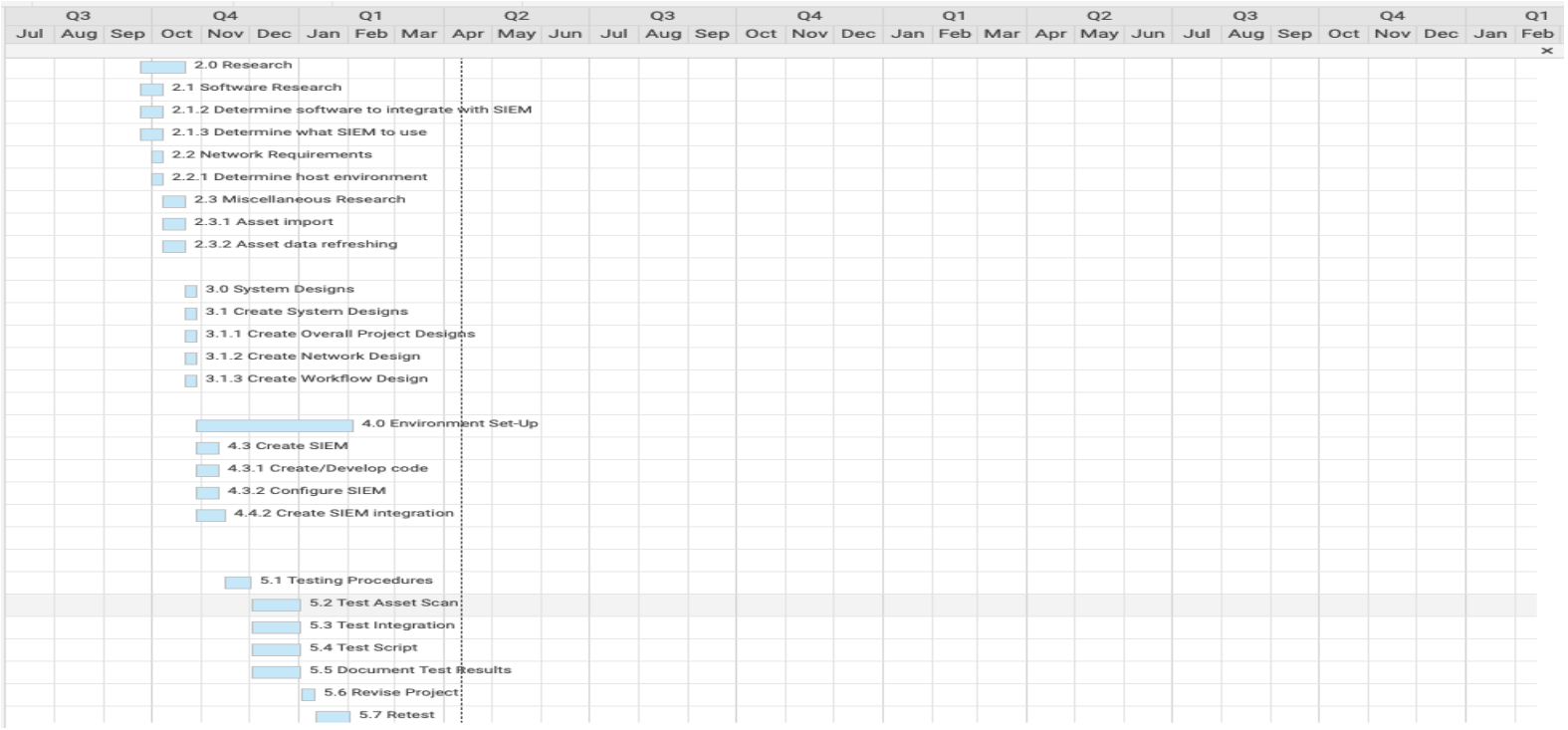
1.17 Assignment 5: Safe Assign for Final Paper	1	04/13/20	04/13/20
2.0 Research	28	09/23/19	10/21/19
2.1 Software Research	14	09/23/19	10/07/19
2.1.2 Determine software to integrate with SIEM	14	09/23/19	10/07/19
2.1.3 Determine what SIEM to use	14	09/23/19	10/07/19
2.2 Network Requirements	7	09/30/19	10/07/19
2.2.1 Determine host environment	7	09/30/19	10/07/19
2.3 Miscellaneous Research	14	10/07/19	10/21/19
2.3.1 Asset import	14	10/07/19	10/21/19
2.3.2 Asset data refreshing	14	10/07/19	10/21/19
3.0 System Designs	7	10/21/19	10/28/19
3.1 Create System Designs	7	10/21/19	10/28/19
3.1.1 Create Overall Project Designs	7	10/21/19	10/28/19
3.1.2 Create Network Design	7	10/21/19	10/28/19
3.1.3 Create Workflow Design	7	10/21/19	10/28/19
4.0 Environment Set-Up	98	10/28/19	02/03/20
4.3 Create SIEM	14	10/28/19	11/11/19
4.3.1 Create/Develop code	14	10/28/19	11/11/19
4.3.2 Configure SIEM	14	10/28/19	11/11/19
4.4.2 Create SIEM integration	18	10/28/19	11/15/19
5.0 Testing			
5.1 Testing Procedures	16	11/15/19	12/01/19
5.2 Test Asset Scan	30	12/02/19	01/01/20
5.3 Test Integration	30	12/02/19	01/01/20
5.4 Test Script	30	12/02/19	01/01/20
5.5 Document Test Results	30	12/02/19	01/01/20
5.6 Revise Project	8	01/02/20	01/10/20
5.7 Retest	21	01/11/20	02/01/20

2.2.1 Table 3: Schedule represents the full timelines for Scout development

2.2.2 Figure 2: Schedule Part 1 Represents first half of the Scout project presented in the Gantt Chart



2.2.3 Figure 3: Schedule Part 2 Represents first half of the Scout project presented in the Gantt Chart



3.0 Technical Elements

The success of any well coded application depends on the software used to build the application. Having a sound and structured design will effectively produce the results what will have a functional application while designing an unstructured application will fail and not meet the emotional criteria of the end user that will be analyzing logs. Figure 4 shows the software design of the Scout Application. A detailed explanation of Scout's network and application layer will follow.

3.0.1 Figure 4: Application Architecture

3.1 Network:

Scout will be hosted through Elastics Cloud Service. This is a cost-effective way to make the asset discovery scalable if the enterprise network is large and needs to be updated hourly. An account was created on the cloud service to give each team member admin access to review and modify our server to serve the needs of our project. The plan is to increase the capacity of the power to parse through data so information can be renewed more quickly without taking a long amount of time.

3.2 Application:

This application was created using Python projects and Postman. The postman project created GET and POST commands that had to be modified with Rumble in order to delete old data if it was already indexed within Logstash.

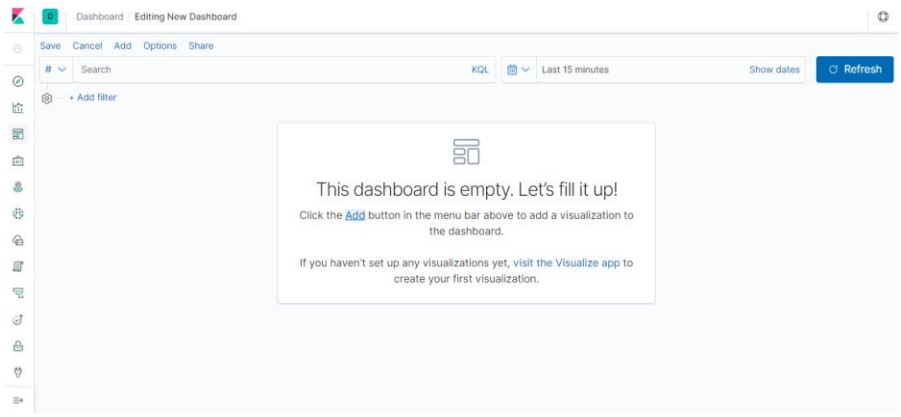
3.3 Database:

The Rumble will scan assets within a network and store it within their interface. From there, any correlated information can be sent using the python script created to import all information into Logstash where network details are to be kept.

3.4 Security:

Elastic Cloud's SaaS offerings are hosted through certified cloud platforms by industry leading infrastructure as a service such as AWS (Amazing Web Services), Google Cloud Platform (GCP), and Microsoft azure. Elastic reviews the security certifications and practices of its sub processors to ensure that there are appropriate physical security measures in force at all premises at which Elastic Cloud data will be processed and stored so Scout's data will not be affected once imported.

3.4.1 Figure 5: Application Log UI represents data beings fed into SIEM



3.4.2 Figure 6: Application Dashboard UI represents usable dashboards displayed in SIEM

4.0 Project Testing: Scout

4.1. Overview & Methodology

To test our project Scout, we used the testing methodology and approach of Iterative testing. This approach suited our project because Scout consists of multiple different applications and processes working seamlessly together that only allows the team to move forward if the step, we are working on, is functioning properly.

- Asset Discovery Scans from Rumble
- Python script pulling asset information
- Integration with Elastic Stack Instance
- Dashboard setup

4.2 Objective

Our objective of testing is to confirm all aspects of the project are working properly before moving into the next phase. With a functioning model in regard to Scout, it needs to work 100% of the time so the program will parse the correct information and send information over in a timely manner. Going through the process of testing phases will allow us to ensure Scout is integrated together through all aspects without issue which will give users a solid experience in which data can be integrated in a seamless manner.

4.3 Testing Scout

When testing Scout, we will be running the Rumble asset scans, the backend scripts, and the integration with Elastic all from our test laptop. This testing process will test the entire stack of processes necessary for Scout to run and function as intended. Testing it this way will help us locate any issues that a process or step may be experiencing that could be causing other steps to malfunction. The primary goal of this whole process is to ensure scans within the network from Rumble is parsed correctly and in an orderly fashion that is aligned with what the current status of the environments are in terms of network recognition and detailing. The positive to testing all steps of the process is to allow the user to experience what Scout would behave like at an enterprise level.

4.4 Testing Procedures

To begin testing we will be setting a baseline report on the stability in each step of Scout. We will record all outcomes in the data table.

- Scout process test:
    1. **Asset scan**: Confirm asset scan is gathering correct information from assets along with various zones
    2. **Scripting**: Ensure the scripts are pulling & parsing the asset information. Asset information is converted into json format.
    3. **Integration**: The proper format is being displayed with the correct information within the Elastic Instance.
    4. **Dashboards**: This test will involve the asset information being automatically uploaded into readable dashboards.

- Asset Scan Testing:

The asset scan test will be conducted using the application Rumble. The main purpose of this test is to ensure the asset scan is able to run and collect data from the specified network. Our group ran 15 individual asset scans on a single network and compared the assets gathered.

The questions asked from this test are:

- **Did the asset scans successfully finish without any errors?**
- **Were all assets discovered across all 15 scans identical?**

4.4.1 Table 4: Asset Scan Test Results displays the results of the asset scan test.

Asset Scan Test via Rumble	Number of Test	Number of Successful Test
Asset Scans Successfully Finished	15	15
All Asset Data Identical	15	15

4.4.2 Figure 7: Assets On Network shows the assets currently on the network that have not yet been assimilated into a SIEM.

IS	UP	NAME	OS	TYPE	MAC	VENDOR	AGE	HW	COMMENTS	TAGS	SVCS	TCP	UDP	ICMP	ARP	RTT/MS	HOP	DET	FIRST SEEN	LAST SEEN	AGENT	SI
00.0	●		Linux	Server						2	2					1.98	0	5060/TCP	2 months ago	14 days ago	DESKTOPBUNCE	Pr
00.0 +2	●		Cisco IOS	Device	00:70:88:E7:2F:BF +1	Cisco Systems,...	2015-12-31	Cisco Device		6	1	3	✓	✓	✓	2.83	0	ARP	3 months ago	5 days ago	DESKTOPBUNCE	Pr
00.1	●		Linux	Server						2	2					3.91	0	5060/TCP	2 months ago	14 days ago	DESKTOPBUNCE	Pr
00.2	●		Linux	Server						2	2					0.88	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.3	●		Linux	Server						2	2					0.96	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.4	●		Linux	Server						2	2					1.82	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.5	●		Linux	Server						2	2					0.93	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.6	●		Linux	Server						2	2					0.99	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.7	●		Linux	Server						2	2					0.97	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.8	●		Linux	Server						2	2					2.79	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.9	●		Linux	Server						2	2					1.93	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.10	●		Linux	Server						2	2					1.99	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.11	●	CP-8831-SEPAC44F219B0F3 +1		IP Phone	AC:44:F2:19:B0:F3	YAMAHA CORPOR...	2009-12-24	Cisco CP-8831		5	3			✓		0.86	1	ICMP	3 months ago	5 days ago	DESKTOPBUNCE	Pr
00.12	●		Linux	Server						2	2					1.04	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.13	●	DVGC7S2	Windows	Desktop						3	2						1	5040/TCP	8 days ago	8 days ago	DESKTOPBUNCE	Pr
00.13	●		Linux	Server						2	2					3.95	0	5060/TCP	2 months ago	14 days ago	DESKTOPBUNCE	Pr
00.14	●	6MGC7S2	Windows	Desktop						3	2						1	5040/TCP	11 days ago	5 days ago	DESKTOPBUNCE	Pr
00.14	●		Linux	Server						2	2					1.79	0	5060/TCP	3 months ago	14 days ago	DESKTOPBUNCE	Pr
00.15	●		Linux	Server						2	2					1.81	0	2000/TCP	2 months ago	2 months ago	DESKTOPBUNCE	Pr
00.15 +1	●	JGLEASON-US4N1 +1	Windows 10 (1903)	Desktop	28:F1:0E:45:82:90	Dell Inc.	2016-03-18			9	6	1	✓			2.00	1	ICMP	8 days ago	7 days ago	DESKTOPBUNCE	Pr

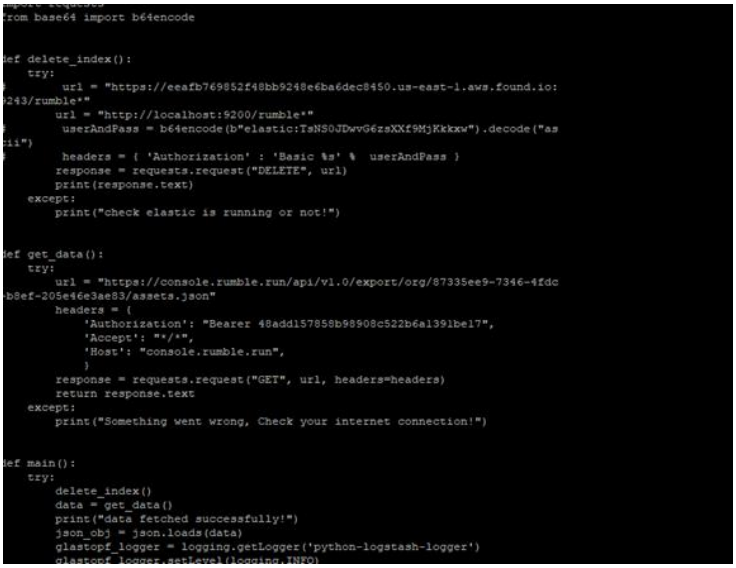
All tests were successful for the Asset scan step in Scout, when using Scout asset information and stability with the scans will not be an issue.

- Script Testing

The integration of Rumble to Elastic was the majority of the testing occurred to confirm or deny whether data would be sent over in the proper format, in a timely manner, and without errors. The goal of this stage was to ultimately build a script where the components of Rumble would be integrated into Elastic.

Through this phase of testing, we had to build logic into the code where it would catch errors and report them back if a step was not able to be processed. All previous activities of asset assimilation would seize until the bug was fixed and data was able to be parsed.

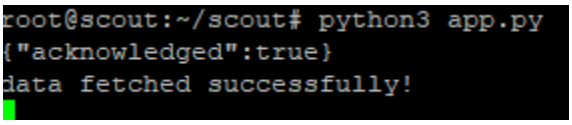
4.4.3 Figure 8: Code Structure provides the structure of code after configuring logic/error handling



4.4.4 Figure 9: Shows Real Time Data being parsed from Rumble into Elastic

Our team had to play and explore around the systems to try and break the logic of the code until the processes were able to be assimilated into the SIEM. After data was able to be sent over without any errors, a message would show at the end of the scans stating the code's process was complete.

4.4.5 Figure 10: Shows the results after data has been assimilated



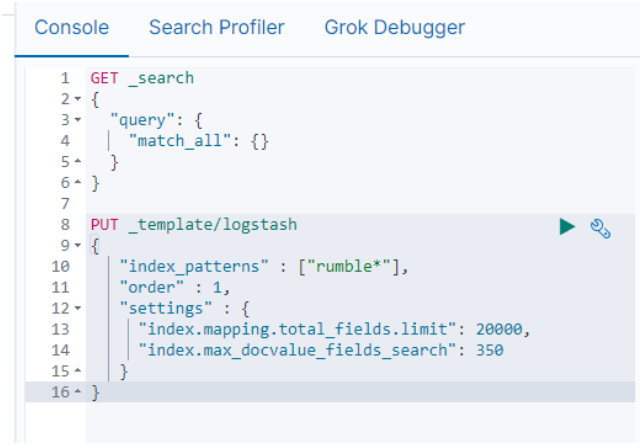
4.4.6 Table 5: Pass/Fail Test Results In this table it displays the results of the script tests which include pass/fails

Script Testing	Pass / Fail	Bug
Parse Test 1	FAIL	API Client error
Parse Test 2	FAIL	Acknowledgement Error - API
Parse Test 3	FAIL	Logic Error
Parse Test 4	FAIL	Functional Error
Parse Test 5	PASS	N/A
Parse Test 6	PASS	N/A
Parse Test 7	FAIL	Run Time Error - Character Limit
Parse Test 8	PASS	N/A
Parse Test 9	PASS	N/A
Parse Test 10	PASS	N/A

- Integration Testing

We were able to verify this as well by manually going into the SIEM environment and comparing the total “hits” that were able to be retrieved. Through the first few weeks of testing we were unable to transfer the data from Rumble to the SIEM instance due to a character limit. After researching and fiddling with the host, we were able to modify a command to increase the character limit from 1000 to the desired field we needed in order for more information to be able to be displayed.

4.4.7 Figure 11: Shows the changes made to increase the max field to allow data to be indexed.

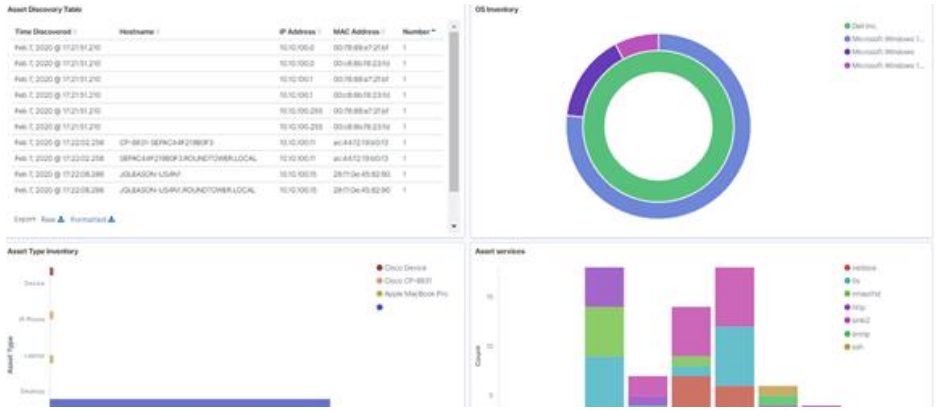


4.4.8 Table 6: Functional Testing Pass/Fail shows the fields that needed to be met in order to satisfy assimilation.

Functional Testing Description	**Pass/Fail**
Assets Parsed with Identical Fields (Host Name, IP, Protocol, etc…)	Pass
Assets Parsed with Matching Scan Time	Pass
Asset Parsed with Correct Order in Which Assets were Scanned	Pass
Assets are Separated in Proper Format	Pass

- Dashboard Testing

4.4.9 Figure 12: Dashboard View shows brief summary of what was collected and converted to dashboards for analyst to use.



During this test, we wanted to ensure that the information parsed through the code was able to recreate dashboards within the SIEM.

4.4.10 Table 7: Dashboard Testing Pass/Fail shows top level needs that needed to match up to satisfy accuracy of imported data to be made into dashboards.

Dashboard Testing Criteria	**Pass/Fail**
Time Matches	Pass
Assets Matching Description from Rumble	Pass
Able to Add Specific Criteria to be Shown	Pass
Correct Proportioning of Numbers when Presented	Pass

Pass / Fail Conditions

The four tests that are conducted must be considered “passed” before we deem Scout to be functional. Testing is conducted until we have no “failed” results.

- Testing Timeline

See below the timeline in which we tested each phase to check the functionality of Scout. We have different roles in our team to maximize efficiency and minimize errors.

Our product is tested at least once a week by our Programmer, and at least once a month by our Project Manager and Designer. This schedule is based upon our Programmer making changes to Scout at a consistent rate and a double-check methodology from the other two. If only the Programmer tests, Jack and Chase can’t confirm functionality the way it is expected to work.

4.4.11 Table 8: Testing Timeline shows our timeline of testing

Team Member	Title	Timeline Testing	Frequency
Jack Brown	Project Manager	10/21/2020 – 4/13/2020	Monthly
Joseph Hu	Programmer	10/21/2020 – 4/13/2020	Weekly
Chase Hood	Designer	10/21/2020 – 4/13/2020	Monthly

- External User Testing Timeline

See below the timeline in which we had outside users test the functionality of Scout. We have a specific end-user based upon the purpose and target of this tool.

The purpose of having SOC analysts test our product is because they ultimately will be the ones utilizing it in the long run because of the nature of the tool and their role in security. We have them testing monthly

to make sure that the functionality that would expect stays and suggestions they want removed are not there.

4.4.12 Table 9: Shows our external user timeline of testing

Group	Timeline Testing	Frequency
SOC Analyst	01/13/2020 – 4/13/2020	Monthly

5.0 Conclusion

Fall Semester 2019

During the fall semester the hardest part of our project was determining what problem in the world we could solve that hasn't been satisfied. Along the way, we met up with Bo after talking about deploying a SIEM using purely ELK. This was too simple at first was not approved due to the vagueness of what we wanted to do with the SIEM. So, a couple days later our group members met up with our directors and managers at our company, RoundTower and discussed the arising problems we had with stale IP's being correlated with assets in our network. From there we wanted to create a network asset discovery pipeline that would easily send data from the Rumble API to ELK. Once we figured out the needs and skills required for our project we quickly got on our feet and started working.

Previously none of us hard any experience with scripting or coding but Joseph took the challenge of this. Chase had some familiarity with Elastic from work and decided he will partake in the creation of the Dashboards and finding the right data to be displayed to potential customers. Jack was always around RoundTower and assisting others as he is the System Admin, so we knew right away we wanted him as our team manager to keep us on our feet.

For our project, we discovered Rumble, a network discovery tool that was in its beta stages but now it is deployed to the public for commercial use. This was something we could use to get asset information within our company and have the correct internal IP associated with the proper devices in the network. To delete and post API data to Logstash in Elastic to be indexed, we needed to create a Python script that can do this. This took a lot of time and researching for Joseph to develop because Rumble is new and there are not existing guides that have configured the API to delete stale data that was indexed and ingest new data that was not concurrent. Once the time that was spent coding and pipelining Rumble's API to Logstash using Python, we were able to index parameters within the information that was discovered. The demo environment is almost finished, we need to create dashboards and specify the parameters that we want to display to our customers and viewers when it is time to present.

This Semester, our team had to learn how to use tools that were outside our normal class and working environments. The biggest challenge was communicating what we all wanted to develop for our company and display for our senior design project. Some other challenges were configuring API's, creating scripts using Python, and developing Elastic stack to be configured on out host environment.

Spring Semester 2020

During the course of the spring semester our group learned and developed multiple skills that helped us complete this project. When it started, our scripting skills were at a beginners level but by the end to have all the features and the entire process run from a single script we had gained a lot of knowledge on how to write an efficient script. We also developed our skills when it comes to setting up and managing a SIEM. This is a sought after trait to have backend engineering skills for any security engineer. Since the start of the fall semesters we were able to complete the script making sure virtually all the bugs and errors were removed. We also were able to make the entire process from asset discovery to SIEM integration automated. Along with finishing and perfecting the script and processes we added additional features such as custom dashboards on the SIEM to easily display the information gathered.

6.0 Appendix

6.1 Problems Encountered

During course of these two semesters we ran into a few critical problems we had to overcome. The first major problem we faced was ingesting asset detail logs into the SIEM. The SIEM had a character limit in the raw format we were trying to ingest with so only certain asset details would be collected. To fix this issue we decided to add to our script that automatically converted to asset information collected into JSON format. This allowed us to send as much information to the SIEM and it be ingested without any issues.

Another problem we faced that our team had to overcome was an issue with keeping up to date asset information. One of our goals and problems we looked to fix is old asset information being stored in SIEMs so when we were designing Scout we had to find a way to make the asset information up to date. Finding an efficient way to do this was challenging but in the end we decided to have all old asset information removed from the SIEM every time a new batch of asset details was ran. To do this we had to add to our script a section that was able to remove old assets as well as uploading the new ones.

6.2 Future Recommendations

If we could go back and do our entire project over I don't believe our group would make many changes. Our project had a solid foundation and we accomplish what we were aiming for. If we had additional time, we would've liked to add more features and details on asset information mainly the usernames. This was a goal of ours but with limited time and the unexpected pandemic we were unable to work on this in the office to gather this information from network devices. Our plans for Scout are to continue using it at RTT to help our security analyst with their incident response. With it being implemented it should prove to be very valuable to our SOC team for the foreseeable future.

Poster 6.3

6.3.1 Figure 13: Expo Poster Illustration