

ADSICS: Anomaly Detection System for Industrial Control Systems

DESIGN DOCUMENT

Team Number
38

Client
Manimaran Govindarasu

Advisers
Manimaran Govindarasu

Moataz A. Abdelkhalek

Team Members/Roles
Planning: Pallavi Santhosh

Communication: Jungho Suh

Organizing: Alex Nicolellis

Controlling: Muhamed Stilic

Team Email
sdmay22-38@iastate.edu

Team Website
<https://sdmay22-38.sd.ece.iastate.edu/>

Table of Contents

List of figures/tables/symbols/definitions	2
Introduction	3
Problem Statement	3
Design Evolution	3
Requirements	4
Engineering Standards	5
Security Concerns and Countermeasures	5
Implementation Details	6
Elastic Stack	6
Datasets	8
Local Machine Learning Techniques	11
Testing and Results	12
Testing	12
Elasticsearch Results	12
Local Model Results	21
Comparison	25
Context	25
Related Literature	25
Related Procedures	26
Closing Material	27
5.1 Conclusion	27
Appendix I - Operation Manual	28
6.1 How to operate Local Machine Learning Anomaly Detection System	28
6.2 How to operate Elasticsearch Anomaly Detection System	31
Appendix II - Alternative Versions	32
Appendix III - Other Considerations	33
Works Cited	34

List of figures/tables/symbols/definitions

Footnote Number:	Description:
Figure 1	Design Overview - ElasticStack
Figure 2	Design Overview - Local
Figure 3	Engineering Standards
Figure 4	Elastic Fleet
Figure 5	Elastic Discover Metric Live Data Dashboard
Figure 6	Elastic Machine Learning Anomaly/Analytic Jobs
Figure 7	Wednesday Dataset Summary
Figure 8	Thursday Dataset Summary
Figure 9	Modbus Dataset Summary
Figure 10	SMOTE results
Figure 11	Elasticsearch Anomaly Detection Dashboard
Figure 12a-b	Elasticsearch High Count Anomaly Detection
Figure 13	Elasticsearch High Count/Rare Anomaly Detection
Figure 14a-b	Elasticsearch Machine Learning Results Wednesday
Figure 15a-b	Elasticsearch Machine Learning Results Thursday
Figure 16a-c	Elasticsearch Machine Learning Results Details
Figure 17a-c	Elasticsearch Machine Learning Results Modbus
Figure 18	Local Machine Learning Results Wednesday
Figure 19	Local Machine Learning Results Thursday
Figure 20a-e	Local Machine Learning Results Modbus
Figure 21	Comparison Table
Figure 22a-d	Operational Manual - CICFlowMeter
Figure 23	Operational Manual - List of Extracted Features
Figure 24	Operational Manual - Elasticsearch Result

1 Introduction

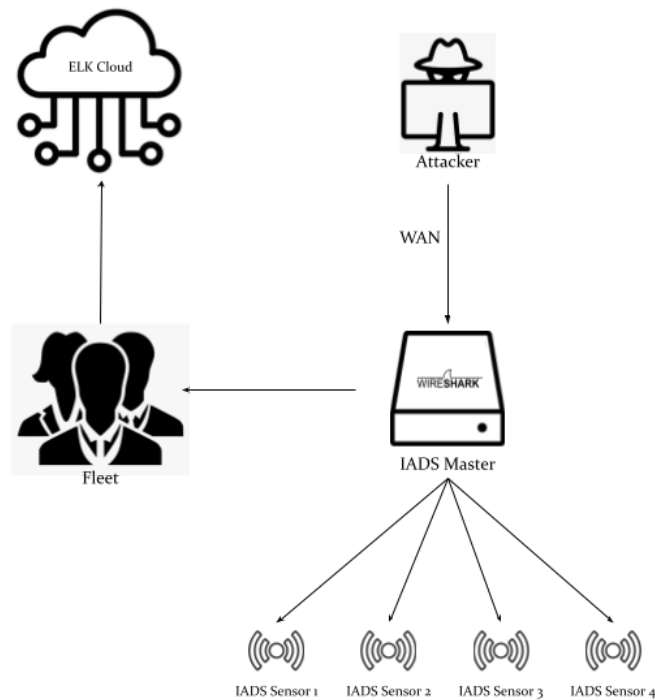
1.1 PROBLEM STATEMENT

With the constant evolution of technology, so much of what we use in our day-to-day lives relies heavily on the power grid. Cyber attacks on distribution companies are debilitating and are now more common due to the increased use of IoT devices and the inadequate security of power grid systems. If we want to feel as though our power is protected, our security measures must continue to advance with the constantly developing technology of cybercriminals. ADSICS is a surveillance program that detects cyber-attacks using machine learning analysis.

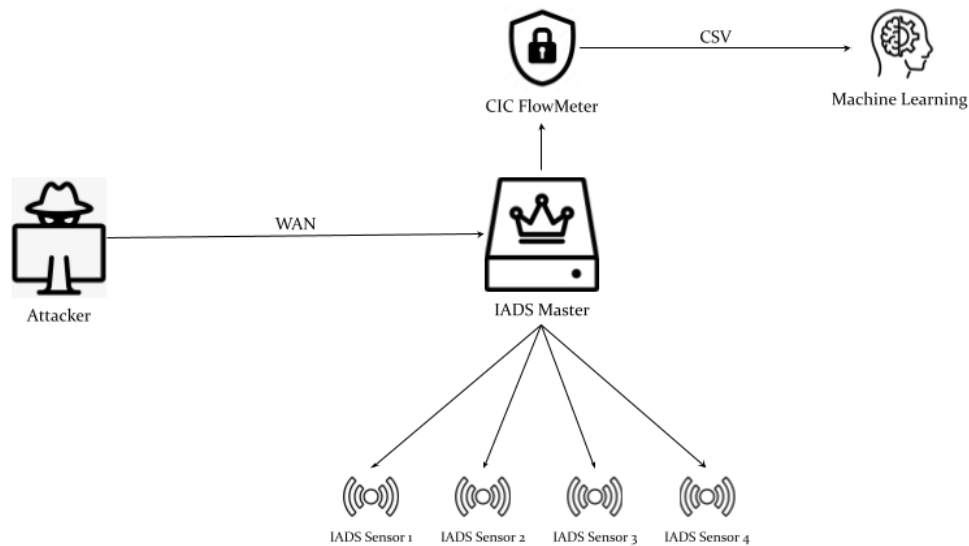
1.2 DESIGN EVOLUTION

In 491, our plan was initially intended to work by controlling and monitoring the sensor network built into the provided testbed. It would then analyze information collected by the sensors and assign alert labels to events completely autonomously. The end goal was to describe a detailed visual output including temporal and spatial correlation, tracking alert information, and performing machine learning analysis quickly and efficiently using the desired platforms (Elastic Stack). We then began to focus on building our machine learning model to compare with the outputs from Elastic to determine whether or not this relatively “new” platform could be relied upon solely as a valid solution to detect and analyze various cyberattacks.

This shift created a split in our project with each branch having a different focus. The first was similar to our initial project design as it was to use the tools within Elastic Stack (Elasticsearch, Kibana, Logstash, etc.) to provide visuals detailing different attacks detected on the testbed. These visualizations show a spectrum of information including the type, length, severity, and of attack. Below is a diagram depicting how the Agents send attack information from the IADS Master to the ELK Cloud for analysis.



The second branch was to create our Machine Learning (ML) model by training it with provided datasets and deploying it on the VM network to detect the anomalies. This portion of the project is used to fully understand the benefits and restraints of relying solely on a platform like Elastic to do anomaly detection. One of the advantages of this approach was that this ML model can be modified to accept the Operational Technology (OT) dataset which aligns with our original goal of the project. Below is a diagram depicting the flow of the ML modeling process. From IADS Master, sensor data is collected and CICFlowMeter will continue to sniff the network traffic. Then, the data is converted into CSV format so that it can be given to the existing ML model. The ML model used is already trained with the multiple attack types and is being deployed on the testbed. From the result of the trained model, the anomalies are detected and classified.



2 Design Overview - Local

1.3 REQUIREMENTS

Functional Requirements:

- Use machine learning to detect network anomalies
- Verify incoming alerts and detect false positives
- Display alerts for easy human understanding
- Present temporal and spatial details for each alert
- Users should be able to add or remove data visualizations on the dashboard
- Alerts should be distinguishable from each other and labeled

Non-Functional Requirements:

- Alerts should be presented intuitively
- Alerts should be color coded by severity
- The system should be able to handle a large volume of alerts
- The system should be reliable and consistent with analysis
- System should have an accuracy of >95% for detecting each alert

Engineering Constraints:

- One branch of the project must use the anomaly detection must use ElasticStack tools (specifically Elasticsearch, Kibana, and Logstash)

1.4 ENGINEERING STANDARDS

Standard	Application	Justification
IEEE 692-2013	IADS SENSOR	Addresses cybersecurity and control related equipment requirements for threat assessment.
	IADS MASTER	
ISO IEC 27039-2015	IADS MASTER	Provides guidelines for selection, deployment, and operations of intrusion detection system detection and prevention systems.
ISO/IEC 27017:2015	CLOUD SERVER	Provides guidance on the information security aspects of cloud computing, recommending the implementation of cloud-specific information security controls that supplement the guidance of the ISO/IEC 27001 and ISO/IEC 27002 standards.
IEEE 802	IADS SENSOR	Describes recommended practices for communication over various types of networks, such as wireless networks.

3 Engineering Standards

1.5 SECURITY CONCERNS AND COUNTERMEASURES

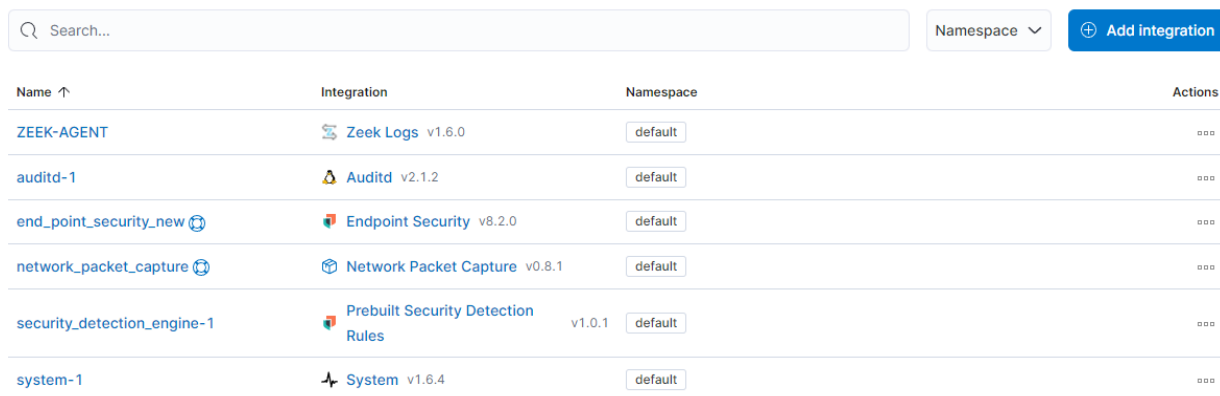
Cybersecurity concerns are the driving force behind the creation of this project and as such, we use a series of simulated attacks by replaying network traffic to model and test our data. As we were mainly developing the functionalities of the system, we did not set up specific security rules on the testbed out of concern that might affect the simulated attacks. Also, a firewall would filter our attacks and be counterintuitive to our project design. However, we can add rules and firewalls to restrict the adverse connection to our anomaly detection system in the future. Additionally, we must use a web application to access the Virtual Machine (VM) testbed and Elastic, so there is a common network security concern such as hijacking or spying on the connection. The communication is protected by HTTPS protocol and both VM testbed and Elastic requires authentication to use the system.

2 Implementation Details

2.1 ELASTIC STACK

The ELK stack, or Elastic Stack, is the amalgamation of three open-source projects into one. Elasticsearch, Logstash(Beats), and Kibana are the layers that make the stack whole. Elasticsearch is the search and analytics engine, Logstash is the ingestion pipeline, and Kibana is the visualizer of all the data coming in. There is also a security aspect built-in where all ingested data is checked off by rules and alerts the system if anyone is trying to intrude. We have used every component offered in Elastic to run our anomaly detection system.

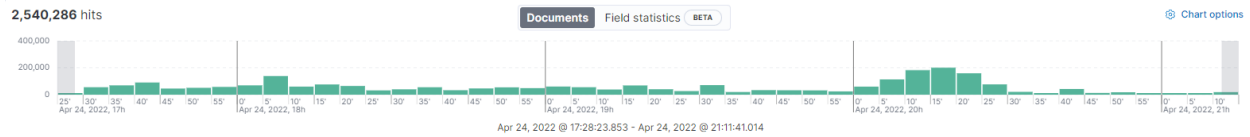
We did not use Elasticsearch for its effective searching capability. However, it is heavily integrated into the Elastic environment's core and it allows us to use the general document storage for all our data and the alert engine that supports our anomaly detection engine. Logstash and Beats are both on the data ingestion layer and do the same thing. They are used as a way to ingest live network traffic and have it be stored on the Elastic cloud system. To ingest data a user can either build their pipeline to the system for ingestion or they can use the variety of integrations prebuilt on Elastic. We went ahead with the integration path and created our fleet of Elastic agents. The integrations we have used in our projects were Network Packet Capture, Zeek, and Auditbeat.



Name ↑	Integration	Namespace	Actions
ZEEK-AGENT	Zeek Logs v1.6.0	default	⋮
auditd-1	Auditd v2.1.2	default	⋮
end_point_security_new	Endpoint Security v8.2.0	default	⋮
network_packet_capture	Network Packet Capture v0.8.1	default	⋮
security_detection_engine-1	Prebuilt Security Detection Rules v1.0.1	default	⋮
system-1	System v1.6.4	default	⋮

4 Elastic Fleet

We call each integration an agent by its data type. Once the Elastic agents can get tangible data coming in, they can also analyze them and enable creation dashboards or other visualizations to use. Then we can monitor the live data by built-in log metric. One downside to the Elastic path instead of the VM testbed ML model is the loss of OT protocols. We cannot show live OT data into Elastic because there is no way to create a pipeline for OT protocols. We can however see we are at least getting the Modbus data by changing one of the ports network packet capture picks up. We cannot visualize anything as the current protocol it is attached to doesn't recognize the Modbus port and Modbus is not supported yet. We have even spoken to the company, Elastic team, and there is no integration to house OT protocols, so we switched to IT network traffic to test our anomaly detection system.



5 Elastic Discover Metric Live Data Dashboard

The Kibana layer has multiple ways to show data. There is a discovery page where users can see the entire packet of live-fed data. The dashboards which come with pre-designed visual showcases what exactly that piece of data is showing. Users can also create their dashboard to their liking and we used it as our base for the front end page to show the user all of the anomalies our system has caught. Kibana also has a machine learning feature. We have used that feature extensively to show live intrusion detection and classification using machine learning jobs. We used the machine learning jobs for high network counts to show the massive spike in the network which can detect DoS and Brute Force attacks. We used the anomaly detection jobs for unusual rare processes and ports to detect XSS attacks. To classify the anomalies in the CSVs we used the data frame analytics option in Kibana. We created classification jobs for all datasets with 4 decision trees inside and 30 important features. We then compared our results from the Elastic and VM ML model to see the difference.

Group ID ↑	Overall score [Ⓢ]	Jobs in group	Latest timestamp	Docs processed	Actions
auditbeat	No anomalies found	7	April 27th 2022, 21:19	319,858	⋮ ⌘
authentication	No results found	1		0	⋮ ⌘
dns	No results found	2		0	⋮ ⌘
endpoint	No anomalies found	6	April 27th 2022, 21:19	319,858	⋮ ⌘
linux	No anomalies found	6	April 27th 2022, 21:19	319,858	⋮ ⌘
logs-ui		2	April 27th 2022, 21:22	258,142	⋮ ⌘
network	No anomalies found	2	April 27th 2022, 21:22	22,741,074	⋮ ⌘
packetbeat	No results found	5	April 25th 2022, 11:32	455,420	⋮ ⌘
process	No anomalies found	5	April 27th 2022, 21:17	308,518	⋮ ⌘
security	No anomalies found	13	April 27th 2022, 21:22	23,505,012	⋮ ⌘

Rows per page: 10 < 1 2 >

Analytics					Total analytics jobs: 23 Running: 0 Stopped: 23	Manage Jobs
ID ↑	Type	Status	Progress	Creation time	Actions	
benign_then_attack	classification	stopped	Phase 8/8	April 24th 2022, 16:53	⋮ ⌘	
mitm_15h_6h_test	classification	stopped	Phase 8/8	April 26th 2022, 21:01	⋮ ⌘	
modbusqueryflood_15m_12h_test	classification	stopped	Phase 8/8	April 26th 2022, 21:19	⋮ ⌘	
no_header_thursday_set	classification	stopped	Phase 8/8	April 24th 2022, 13:06	⋮ ⌘	
pingflooddos_15m_12h_test	classification	stopped	Phase 8/8	April 26th 2022, 21:32	⋮ ⌘	

6 Elastic Machine Learning Anomaly/Analytic Jobs

One of the many tools we used besides the Elastic agents and their fleet was to use TCPReplay. We used TCPReplay as a means to replay live traffic saved in a PCAP file exactly as it was sent when the traffic was captured. TCPReplay is then used by the command line which defines which network interface it will run on, and the PCAP file with the network flow information to emulate the network traffic. We used Wireshark, a tool that visualizes end-to-end packet communication, to see if our simulated network flow is detected on Elasticsearch. At first, while replaying the network traffic according to the PCAP file, TCPReplay frequently failed to send the traffic and even stopped during execution. Later, it was found that the speed of replayed

network traffic logs was too fast for Elasticsearch to accept, resulting in skipping most of the packets which made it undetectable. Also, the messages being sent by TCPReplay were too long for the network interface's Maximum Transmission Unit (MTU). Although we changed the MTU of the network interface to its maximum, 9000, some of the packets still failed to be sent since they were longer.

To resolve the problem we used TCPReplay's sub-option 'TCPReplay-edit' and 'MTU trunc' to truncate the longer packets to the MTU size. Despite the truncation, the number of affected packets was less than 1% so there was no adverse effect on the testing accuracy. Alongside TCPReplay, we also utilized Editcap. It is used to extract the attack by setting start and end times from the entire PCAP file. Instead of waiting eight hours to get the testing result, we could start the attack that lasts only twenty minutes to run which increased our efficiency of testing. The ability of editing PCAP files led to making different scenario PCAP files for different testing purposes.

2.2 DATASETS

The main datasets we used for different types of attacks such as DoS, Brute Force, and XSS from <https://www.unb.ca/cic/datasets/ids-2017.html>. We got our datasets from the 2017 intrusion detection evaluation data sets from the University of New Brunswick. We used the 2017 dataset because of the variety of attacks and the length of an entire working hour with network traffic. Out of five days we used two different working days, Wednesday for DoS attacks and Thursday for Website attacks. Each dataset comes with the CSV and PCAP files of each day. We used the PCAP files to replay the network flow along with attacks inside it and used CSV files to classify attacks with our local VM ML model.

Prebuilt CSV datasets were used for training the local ML model and for testing before live traffic tests. It has 78 features and "Label" is already provided for classification. Also it was used as a reference for making CSV files from PCAP files using CICFlowMeter. From the PCAP file, FIN packet used as an indicator of the end of the flow, CICFlowMeter created the CSV datasets. The CSV datasets are each compiled of a flow. These data flows consist of 78 features ranging from Destination Port to Flow ID. By analyzing the PCAP files, a "Label" column was created and imputed with the various "attack" types associated with each data flow.

Wednesday_Workingday dataset is the largest dataset with thirteen gigabytes and has four classified DoS attacks: DoS slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye as well as a heartbleed port 444 attack that occurs later in the afternoon.

Wednesday_Workingday			
DoS / DDoS			
Network Traffic from 9:00 A.M. to 5:00 P.M.			
	OS	IP	Local IP
Attacker	Kali	205.174.165.73	
Victim DDoS	WebServer Ubuntu	205.174.165.68	192.168.10.50
Victim Heartbleed	Ubuntu12	205.174.165.66	192.168.10.51
Type of Attack		Start	End
DoS Slowloris		9:47	10:10
DoS Slowhttptest		10:14	10:35
DoS Hulk		10:43	11:00
DoS GoldenEye		11:10	11:23
Heartbleed Port 444		15:12	15:32

7 Wednesday Dataset Summary

The other dataset we used is the Thursday_Workday_Morning dataset. We used the morning subset that includes Web Attacks: Brute Force, XSS, SQL Injection.

Thursday_Workingday_Morning			
Web Attack			
Network Traffic from 9:00 A.M. to 12:00 P.M.			
	OS	IP	Local IP
Attacker	Kali	205.174.165.73	
Victim	WebServer Ubuntu	205.174.165.68	192.168.10.50
Type of Attack		Start	End
Brute Force		9:20	10:00
XSS		10:15	10:35
SQL Injection		10:40	10:42

8 Thursday Dataset Summary

For Modbus, we used Modbus dataset that has Modbus query flood, Ping flood, and tcpSYN flood attacks. https://github.com/tjcruz-dei/ICS_PCAPS

Modbus_Combined_Train			
Three different DDoS. Modbus query flood, Ping flood, tcpSYN flood			
Network Traffic lasts 6,12,12 hours			
	IP		
Attacker for MQF	172.27.224.50	172.27.224.70	172.27.224.80
Victim	172.27.224.250		
Type of Attack		Description	
Modbus Query Flood		30 minute duration	
Ping Flood		30 minute duration	
TCP SYN Flood		30 minute duration	

9 Modbus Dataset Summary

2.3 LOCAL MACHINE LEARNING TECHNIQUES

The datasets mentioned above were quite imbalanced in terms of the amount of each “attack” alert to the majority category, “Benign.” For example, in the Wednesday_Workinghours dataset, there was initially 44042 BENIGN alerts (0.0 in Figure 10), 5796 DoS Slowloris (1.0 in Figure 10), 5499 DoS Slowhttptest (2.0 in Figure 10), 231073 DoS Hulk (3.0 in Figure 10), and 10293 DoS GoldenEye (4.0 in Figure 10). This created an issue as the Benign alerts had a high accuracy but the more scarce ones like DoS Slowloris were very hard to detect accurately.

To balance out the number of each of these attacks, we used Synthetic Minority Oversampling Technique (SMOTE) on the minority alerts (alerts 1.0, 2.0, and 4.0) and random undersampling on the majority alert (alert 0.0 and 3.0). We decided to set the undersampling amount to $\frac{1}{2}$ the largest alert count (ie. the BENIGN count). Then, we set the oversampling amount to $\frac{1}{2}$ of the undersampling amount. Afterwards, the count of each attack became as follows; 231073 BENIGN alerts (0.0 in Figure 10), 115536 DoS Slowloris (1.0 in Figure 10), 115536 DoS Slowhttptest (2.0 in Figure 10), 231073 DoS Hulk (3.0 in Figure 10), and 115536 DoS GoldenEye (4.0 in Figure 10). The results of using SMOTE can be seen below in Figure 10.

```
Counter({0.0: 440042, 3.0: 231073, 4.0: 10293, 1.0: 5796, 2.0: 5499})  
Counter({0.0: 231073, 3.0: 231073, 1.0: 115536, 2.0: 115536, 4.0: 115536})
```

10 SMOTE results

As mentioned earlier, there were 78 features in the dataset and we felt that the model would become more precise if we used feature selection to narrow down the important features for our data. We decided that the best way to do this would be to use forward model selection with cross validation to select the features and narrowed them down from 78 to 30. Unfortunately, this did not work as the features outputted fit only the dataset we trained them on (Thursday Morning). To expand, because Thursday’s attacks were XSS, SQL, and Brute Force, all of the features extracted did not fit perfectly with the Wednesday’s attacks which were mostly DoS types. This issue was fixed, however, in the final model as the code used included feature selection so regardless of the dataset, the model would use the 30 most relevant features for that dataset.

All of this was implemented using scikit-learn and the machine learning algorithms we used were Decision Tree and Random Forest. For each dataset, models were trained and saved to a .joblib file. Then, they were employed by loading the file and obtaining the predictions. The output is in the format of a confusion matrix, and accuracy table which shows results individually for each possible label value including the accuracy.

3 Testing and Results

3.1 TESTING

The code for this project is simple in structure, but involves complex machine learning models. Therefore, traditional code testing is not needed for this project, as there are no branches or individual functions to consider. However, it is important to verify that our machine learning models provide consistent and accurate results. Even more important is the verification of Elasticsearch's results, as that software does not expose its internal process to its clients. This project considers Elasticsearch's machine learning model as a black box, so it is important and useful to test its output.

The tests were conducted by creating several small test datasets that were used as input to our trained models. The tests were: anomalous data followed by benign data, anomalous data preceded and followed by benign data, and benign data followed by anomalous data. The idea behind these tests was that the results should be easily predicted and verified by humans. Therefore, the machine learning model's output could be evaluated without having to rely on more machine learning. These tests were also conducted multiple times to check for consistency. Each type of anomaly was tested in this manner. More details regarding these tests can be found in section 3.2.

3.2 ELASTICSEARCH RESULTS

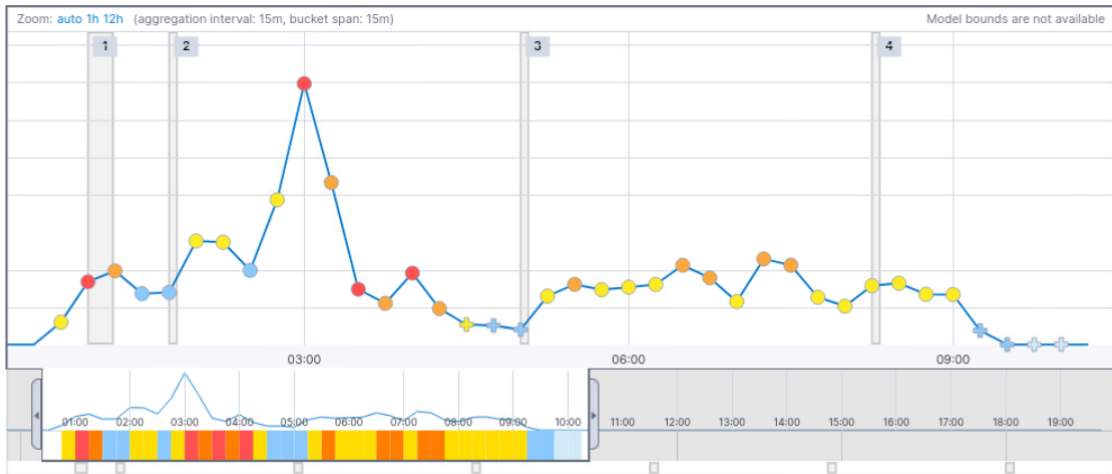
The below graphs illustrate the severity of anomalies over time in the dataset, as well as the overall accuracy for each anomaly type.



11 Elasticsearch Anomaly Detection Dashboard

The figure 11 here shows our front end page of our anomaly detection system. Here the user can choose what time range they want to analyze, in our case we chose a week of data. Then show all of the anomalies scores at the top left by each job. Going down it showcases the exact contents inside each job. To the left those jobs showcase the rare processes going on with hosts and that is what we used in finding XSS anomalies. The

graph to the right showcases the high count of events happening in the network for us to pinpoint DDoS attacks and Brute Force attacks.

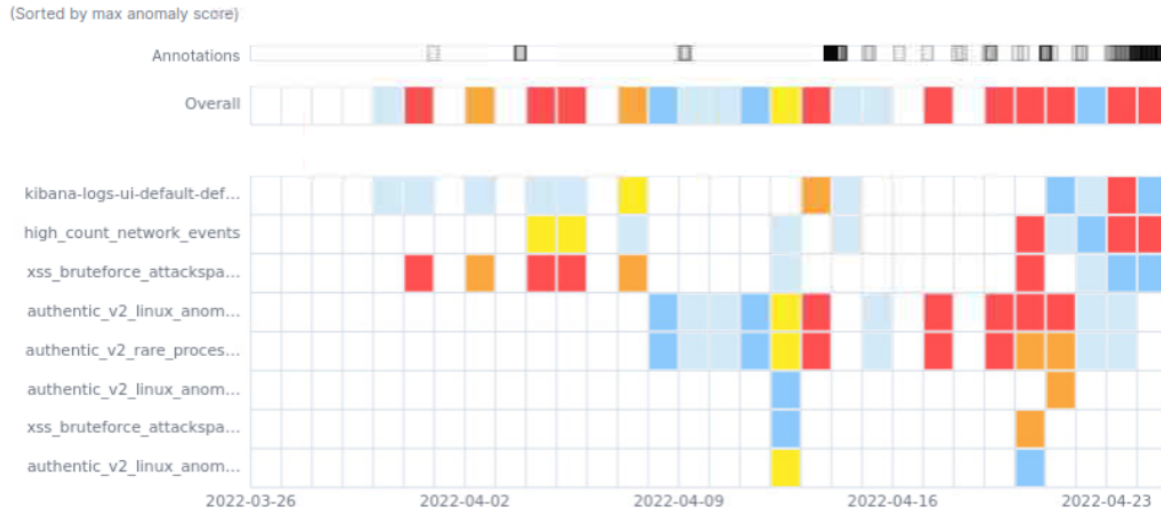


12a Elasticsearch High Count Anomaly Detection



12b Elasticsearch High Count Anomaly Detection

Figure 12a and b showcase the high counts jobs. We focused primarily on DDoS attack verification and we used the highcount_network jobs in order to visualize when the network has activity that rises way more than the normal. Here we have the full 8.5 hr pcap file from the Wednesday data simulated like it really did back when they ran the tests. The attacks can be classified into the slowLoris at the first red dot, slow http to the second spike, hulk the massive spike, then finally finish off the red dot with the goldeneye attack.



13 Elasticsearch High Count/Rare Anomaly Detection

Figure 13 showcases all of the anomaly detection jobs we have used for our analysis on the elastic jobs and their anomaly scores to determine when we run our attacks it can pick them up. We used the highcounts attack to determine the bruteforce and ddos attacks. Then we used the rare_user activity and processes to determine the scoring for XSS attacks.

Actual class	BENIGN	DoS Hulk	DoS Slowhttptest	DoS slowloris
BENIGN	99%	1%	0%	0%
DoS Hulk	0%	100%	0%	0%
DoS Slowhttptest	1%	0%	99%	1%
DoS slowloris	0%	0%	0%	100%

Evaluation quality metrics

0.998 Overall accuracy **0.994** Mean recall

Per class recall and accuracy

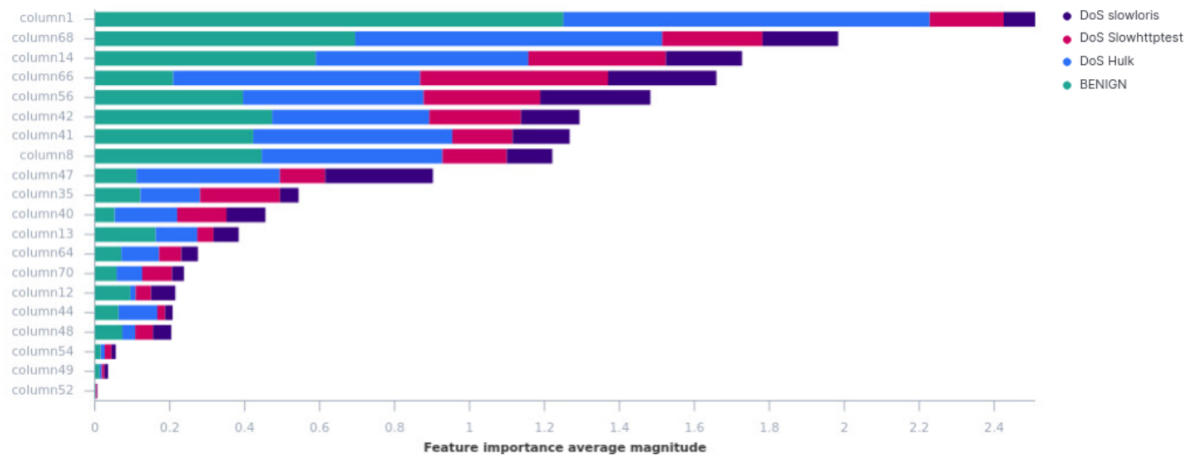
Class	Accuracy	Recall
BENIGN	0.998	0.993
DoS Hulk	0.998	1
DoS Slowhttptest	1	0.985
DoS slowloris	1	0.996

14a Elasticsearch Machine Learning Results Wednesday

Total feature importance ^

[Feature importance docs](#)

Total feature importance values indicate how significantly a field affects the predictions across all the training data.



14b Elasticsearch Machine Learning Results Wednesday

We used the machine learning algorithm option, but going the data frame analytic route instead of intrusion detection machine learning like the figures 11-13. With the CSV data set we can now classify the attacks using decision trees, just like the local implementation and then compare the two in the future to see how they run side by side. Figures 14a and 14b show how the the dataset for the wednesday attacks come up to be fairly accurate after running. We have nearly a perfect 99% accuracy and recall for all attacks. One thing that the elastic search version has that the local implementation doesnt is the feature selection importance graphs which is Figure 14b, there the user can see what exact feature the algorithm is using to base its test off from and is useful in messing with the training of the model to make it more accurate.

Model evaluation [^](#)

Job status docs evaluated
stopped **170366**

Normalized confusion matrix for entire dataset [🔗](#)

Predicted class

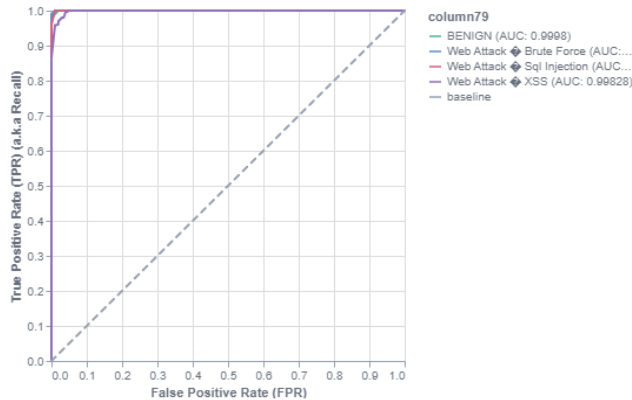
Actual class	BENIGN	Web Attack ⚡ Brute F...	Web Attack ⚡ Sql Inje...	Web Attack ⚡ XSS
BENIGN	100%	0%	0%	0%
Web Attack ⚡ Brute Force	1%	19%	0%	80%
Web Attack ⚡ Sql Injection	0%	43%	0%	57%
Web Attack ⚡ XSS	5%	0%	0%	95%

Evaluation quality metrics

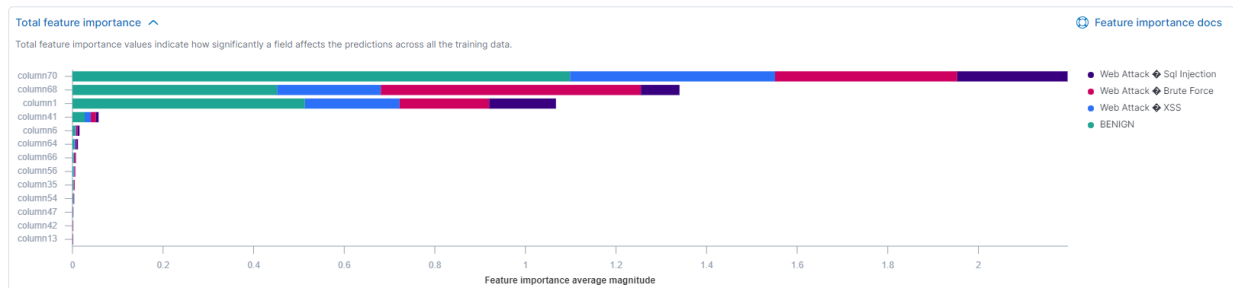
0.99 **0.534**
Overall accuracy [?](#) Mean recall [?](#)

> Per class recall and accuracy

Receiver operating characteristic (ROC) curve [🔗](#)



15a Elasticsearch Machine Learning Results Thursday



15b Elasticsearch Machine Learning Results Thursday

Figures 15a and 15b showcase the Thursday_MorningWorkday data set in which we have attacks classified for benign, XSS, Brute Force, and SQL injection attacks. This dataset is the most skewed than the Wednesday dataset. The reason why is because the data set only comprises just a few nodes of data for the attack. The only attack with more than a couple hundred was the brute force attack while the Wednesday one has multiple thousand for each attack. It was able to pick up the benign classification of normal behavior then

pick up the attacks for XSS but the amount for both brute force and sql was too small so there was nothing to show. The feature selection was also way smaller than the Wednesday datta set with not many important values to show for the classification.

Evaluation quality metrics

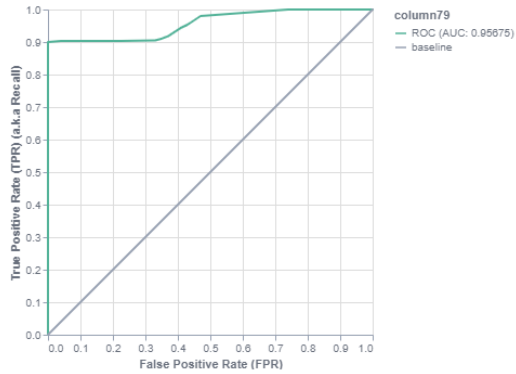
0.94 Overall accuracy ⓘ **0.95** Mean recall ⓘ

∨ Per class recall and accuracy

Class	Accuracy	Recall
BENIGN	0.94	0.9
DoS GoldenEye	0.94	1

Rows per page: 10 ∨

Receiver operating characteristic (ROC) curve ⓘ



16a Elasticsearch Machine Learning Results Details

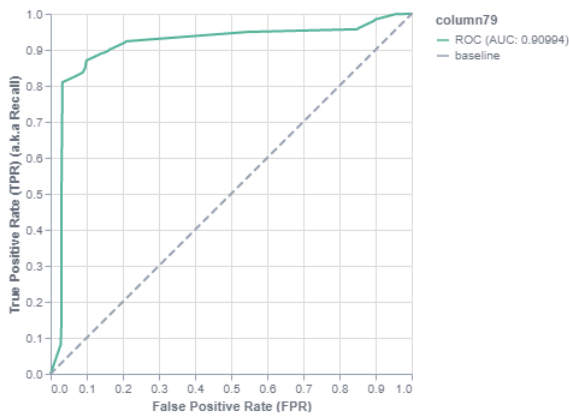
Actual class	BENIGN	Web Attack T ₂ ½ Brute ...
BENIGN	91%	9%
Web Attack T ₂ ½ Brute For...	19%	81%

Evaluation quality metrics

0.88 Overall accuracy ⓘ **0.862** Mean recall ⓘ

> Per class recall and accuracy

Receiver operating characteristic (ROC) curve ⓘ



16b Elasticsearch Machine Learning Results Details

Actual class		BENIGN	Web Attack 1/2 Brute ...
BENIGN		91%	9%
Web Attack 1/2 Brute For...		0%	100%

Evaluation quality metrics

0.94

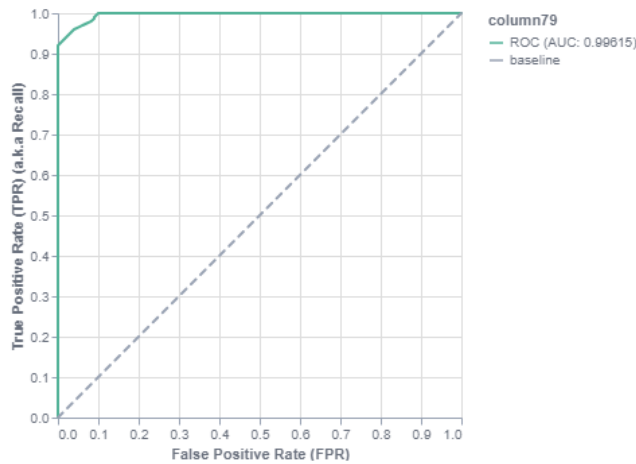
0.956

Overall accuracy ②

Mean recall ②

> Per class recall and accuracy

Receiver operating characteristic (ROC) curve ②



16c Elasticsearch Machine Learning Results Details

These are of the many tests we have done to do more training on the Elastic System to see if we are correctly getting the right classification of information. We have done attacks at a time range starting at the very top, very bottom, and some tests where they are located in the middle of the network flow. All of the attacks came through with high accuracy and recall to 99% for the Wednesday dataset. The only ones with trouble are the figures 16b and 16c. Those are the ones done for the Thursday_Morning dataset. They have a higher skew due to the fact that the datasets have smaller datasets and that we haven't run that many attacks for better classification.

Columns			
Actual class	Benign	mitm	
Benign	100%	0%	
mitm	0%	100%	

Evaluation quality metrics

Overall accuracy $\text{\textcircled{1}}$ Mean recall $\text{\textcircled{2}}$

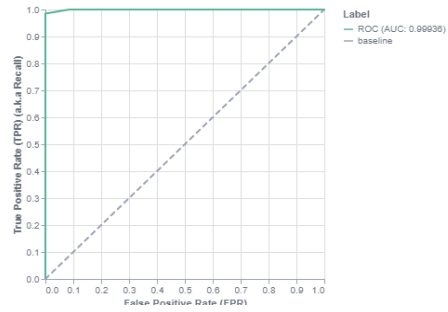
Per class recall and accuracy

Class	Accuracy	Recall
Benign	1	1
mitm	1	1

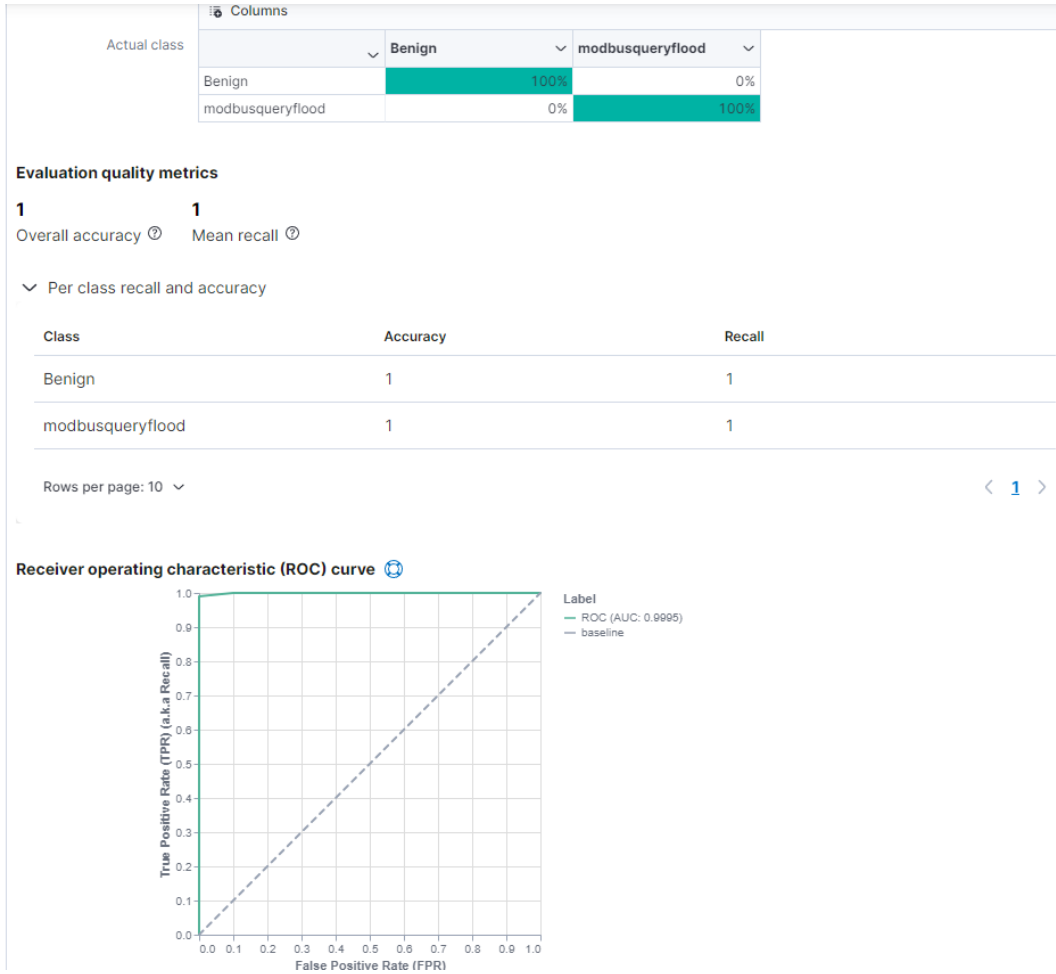
Rows per page: 10

< 1 >

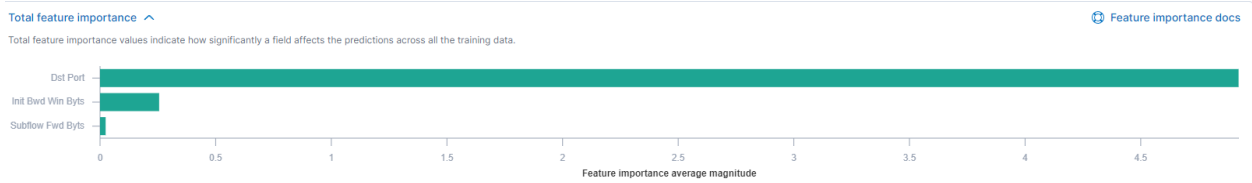
Receiver operating characteristic (ROC) curve



17a Elasticsearch Machine Learning Results Modbus



17b Elasticsearch Machine Learning Results Modbus



17c Elasticsearch Machine Learning Results Modbus

Figures 17a-17c showcase the Elastic machine learning outcomes from running the Modbus datasets attack. Here we show mitm and Modbus query flood attacks being classified and reaching a high 100% accuracy level nailing all of the tests. Figure 17c shows the exact important features that made the correct classification and we used those to better fit our models accuracy.

3.3 LOCAL MODEL RESULTS

This confusion matrix represents the local model's success rate on each type of anomaly.

DECISION TREE WEDNESDAY TEST				
[[17380	8	428	26]	
[10	8902	0	16]	
[10	1	17760	14]	
[3	0	2120	6787]]	
	precision	recall	f1-score	support
0.0	1.00	0.97	0.99	17842
1.0	1.00	1.00	1.00	8928
2.0	0.87	1.00	0.93	17785
3.0	0.99	0.76	0.86	8910
accuracy			0.95	53465
macro avg	0.97	0.93	0.94	53465
weighted avg	0.96	0.95	0.95	53465

RANDOM FOREST WEDNESDAY TEST				
[[17400	2	421	19]	
[9	8903	0	16]	
[9	1	17764	11]	
[3	0	2120	6787]]	
	precision	recall	f1-score	support
0.0	1.00	0.98	0.99	17842
1.0	1.00	1.00	1.00	8928
2.0	0.87	1.00	0.93	17785
3.0	0.99	0.76	0.86	8910
accuracy			0.95	53465
macro avg	0.97	0.93	0.95	53465
weighted avg	0.96	0.95	0.95	53465

18 Local Machine Learning Results Wednesday

The Wednesday dataset contains benign data (0) and three types of DoS attacks (1, 2, 3). The local model had great success on classifying this data, with an overall accuracy of 95% on the testing data.

```

DECISION TREE THURSDAY
[[167541 48 533 64]
 [ 2 220 1213 72]
 [ 22 0 629 1]
 [ 0 0 0 21]]
precision recall f1-score support
0 1.00 1.00 1.00 168186
1 0.82 0.15 0.25 1507
2 0.26 0.96 0.42 652
3 0.13 1.00 0.23 21

accuracy 0.99 170366
macro avg 0.55 0.78 0.47 170366
weighted avg 1.00 0.99 0.99 170366

```

```

RANDOM FOREST THURSDAY
[[167627 6 496 57]
 [ 3 219 1213 72]
 [ 21 0 630 1]
 [ 1 0 0 20]]
precision recall f1-score support
0 1.00 1.00 1.00 168186
1 0.97 0.15 0.25 1507
2 0.27 0.97 0.42 652
3 0.13 0.95 0.23 21

accuracy 0.99 170366
macro avg 0.59 0.77 0.48 170366
weighted avg 1.00 0.99 0.99 170366

```

19 Local Machine Learning Results Thursday

The Thursday dataset contains benign data as well as three types of web attacks (Brute Force, SQL injection, XSS). The local machine learning model had major problems classifying two of the attacks. However, we believe that this performance is primarily an issue with the dataset itself, as it is highly skewed to the Benign. We partially mitigated this problem by using oversampling for attacks and undersampling for Benign, and this worked for Brute Force. However, accuracy for other types of attacks are low and we believe that this phenomenon occurred because of lacking data points beyond the mitigation capability of oversampling, especially for XSS (Attack type 3) which has less than 150 data points. Elasticsearch did not perform very well on it either, which indirectly proves that the dataset itself is too imbalanced.

The Modbus dataset contains benign data as well as three types of DDoS Attacks (Modbus Query Flood, Ping Flood, TCP SYN Flood). The training is first done separately by each attack type, trained with 30 minute attack duration for 6h, 12h, 12h dataset and then tested with 30 minute attack duration for 1 hour, 15 minute attack duration for 12h, and 12h.

[[16653 0] [0 26354]]		precision	recall	f1-score	support
0		1.00	1.00	1.00	16653
1		1.00	1.00	1.00	26354
accuracy				1.00	43007
macro avg		1.00	1.00	1.00	43007
weighted avg		1.00	1.00	1.00	43007

20a Local Machine Learning Results Modbus - Modbus Query Flood

[[28748 2] [5633 35637]]		precision	recall	f1-score	support
0		0.84	1.00	0.91	28750
1		1.00	0.86	0.93	41270
accuracy				0.92	70020
macro avg		0.92	0.93	0.92	70020
weighted avg		0.93	0.92	0.92	70020

20b Local Machine Learning Results Modbus - Ping Flood

[[16237 0] [0 30530]]		precision	recall	f1-score	support
0		1.00	1.00	1.00	16237
1		1.00	1.00	1.00	30530
accuracy				1.00	46767
macro avg		1.00	1.00	1.00	46767
weighted avg		1.00	1.00	1.00	46767

20c Local Machine Learning Results Modbus - TCP SYN Flood


```

[[60811  787]
 [ 5635 92568]]

```

	precision	recall	f1-score	support
0	0.92	0.99	0.95	61598
1	0.99	0.94	0.97	98203
accuracy			0.96	159801
macro avg	0.95	0.96	0.96	159801
weighted avg	0.96	0.96	0.96	159801

20d Local Machine Learning Results Modbus -Benign as 0, DDoS Attack as 1

```

[[60809  0  789  0]
 [  0 26333  21  0]
 [ 5635 14882 5765 14988]
 [  2 14896  649 15032]]

```

	precision	recall	f1-score	support
0	0.92	0.99	0.95	61598
1	0.47	1.00	0.64	26354
2	0.80	0.14	0.24	41270
3	0.50	0.49	0.50	30579
accuracy			0.68	159801
macro avg	0.67	0.65	0.58	159801
weighted avg	0.73	0.68	0.63	159801

20e Local Machine Learning Results Modbus -Benign as 0, other attacks as 1,2,3

Figure 20a, 20b, 20c shows the high accuracy for each attack type. The local machine learning model had no problems classifying each of the attacks with high accuracy. However, when we trained those altogether, the lowering accuracy phenomenon was shown. When we considered all attacks as one DDoS attack, its performance was fine, but when we considered each attack as separate, it resulted in low accuracy for some types of DDoS attacks. We believe that this phenomenon is the result from the three attacks' characteristics being reasonably similar. This is supported by the fact that when tested with normal (0) and DDoS labels (1), it had 92% accuracy, and even when the attacks were labeled separately, normal state's accuracy was still 92%, meaning that it has a low false positive rate. We concluded that the local ML model can distinguish normal and DDoS attack, but not by which type of DDoS attack.

3.4 COMPARISON

	Elasticsearch		Local ML			
	Wednesday	Thursday	Wednesday	Thursday	Modbus	
					Combined	Separate
Max Precision	100.0%	100.0%	100.0%	100.0%	99.0%	95.0%
Min Precision	99.0%	0.0%	87.0%	23.0%	92.0%	24.0%
Accuracy	99.8%	99.0%	95.0%	99.0%	96.0%	68.0%
Macro Average	99.4%	53.4%	95.0%	48.0%	96.0%	67.0%

21 Comparison Table

The Elasticsearch machine learning outperformed the local machine learning models in both datasets. The Wednesday dataset results only differed by a few percentage points, but the performance increase was more noticeable for the Thursday dataset anomalies. Although both models were able to correctly identify benign data, which makes up the vast majority of samples in that dataset, Elasticsearch did outperform the local model on the anomalies, except for SQL injection, which elasticsearch could not identify at all. In addition to the improved metrics, Elasticsearch provides many more results that are not available with our machine learning implementation, such as a graph that shows anomalies over time, and the ability to filter results. This means that an actual client would greatly prefer Elasticsearch over our own methods, because the dashboard in Elasticsearch can give them the information they need to understand and possibly prevent these anomalies from occurring in the future. However, Elasticsearch does not currently have native support for parsing Modbus data. In our client's system, Elasticsearch's utility is diminished because its features are determined solely by its developers, who may have different priorities. A local machine learning setup like ours may still have its place in situations where some features such as Modbus traffic integration, which is underdeveloped on the Elasticsearch platform, and the ability to fully customize the models without relying on third-party developers outweighs the performance decrease.

4 Context

4.1 RELATED LITERATURE

In the context of literature and industry products, our project is considered an anomaly-based intrusion detection system (AIDS). Definitions for terms like AIDS, as well as other information for this section, are drawn from the research of Khraisat et al. An intrusion detection system seeks to identify cyber-attacks, and some, like our own, do so through the concept of an anomaly, or an outlier piece of data. On the other hand, a signature-based intrusion detection system (SIDS) detects cyber-attacks by maintaining a database of known attacks, and checking the signatures of new data against the known attacks. The primary advantage of using an AIDS is that it is capable of detecting new types of intrusions that have not been seen before. However, this often means that unexpected non-anomalous data may be detected as an attack. This event is called a false positive, and mitigating them is a primary focus of AIDS.

There are three main methods of detecting attacks in AIDS. First, there are statistics-based techniques, which build a distribution model for normal behavior. Then, new packets are assigned a probability based on the model. Low probability packets are considered anomalous. Next, knowledge-based techniques use a model of normal behavior that is manually built by humans. Then, data that differs from the model is marked as anomalous. If built correctly, the knowledge base will prevent false positives because all normal

behavior can be recognized. However, it is very difficult in practice to build and maintain such a knowledge base. Finally, machine learning techniques apply various rules that attempt to recognize patterns in data, resulting in new data being assigned a score that predicts whether it is anomalous.

Our system uses machine learning techniques. Specifically, it uses decision tree and random forest algorithms. Decision trees develop a series of branches, which classify samples based on certain feature values. Random forests consist of a series of decision trees. They are both supervised machine learning algorithms, which means that they are trained with datasets that contain labels that define whether each sample is anomalous. Unsupervised machine learning techniques exist that do not have such labels, and instead detect more general patterns and data and assign anomalies based on which data do not fit into the patterns. In addition, other supervised learning techniques are used in AIDS, such as nearest neighbor and support vector machines. Nearest neighbor techniques assign an anomaly score to data based on the nature of data points with similar values. Support vector machines draw a hyperplane between data points, and classify anomalies based on which side of the plane they are on. Out of the many options, the decision tree was chosen as the ideal algorithm because it fits well with our dataset, which contains a large number of samples with many features. Decision tree is efficient and relatively simple, and can handle our dataset well. With this information, our system's place in the wide array of intrusion detection systems should be clear.

4.2 RELATED PROCEDURES

The field of cybersecurity contains many products that serve as an intrusion detection system. One that is similar to our design is Sqrrl, which develops software that allows its clients to detect and respond to data breaches, among many other types of cyber-attacks. It uses many advanced technologies, including machine learning. However, there are many differences between what we have developed compared to products like Sqrrl. Sqrrl is designed to be applied generally to any company with cybersecurity concerns, and therefore must be designed to reply to any kind of conceivable attack. On the other hand, our project is very specifically targeted towards examining network packets received by our virtual machines, and we only focus on a few types of attack. Although our project is similar in concept to the designs of many real-world companies, they often must design their software to be much more broadly applicable, while our project is specifically tailored to one system.

5 Closing Material

5.1 CONCLUSION

In conclusion, we would say that we succeeded in satisfying functional requirements. We have been able to showcase how we use our trained machine learning model to classify Modbus attacks as well as various DDoS and Web attacks. We also have been able to create a full system on the cloud using Elasticsearch and locally using our own machine learning model. We have Elastic set up to be able to ingest, monitor, classify, and graph all the anomalies with their data algorithm visualizer and anomaly detection visualizer on Kibana. One of the biggest turning points for us was that Elastic did not have means for ingesting Modbus data. So, we have been able to replay and edit constant live network traffic using various tools such as tcpreplay, tcpdump, and tshark. While having successfully ingested network information into Elastic, we have been able to detect high counts of network events with DDoS attacks in Wednesday dataset. We also have been able to detect the high counts of the brute force attacks from the Thursday dataset as well as the hidden processes of XSS web attacks by detecting rare processes. Although we were not able to ingest live Modbus data with Elastic, we have been able to classify Modbus attacks with Elastic's machine learning data analytics module. On top of this, we created our own local machine learning model and enabled live anomaly detection for Modbus. We have tested the model multiple times by splitting the original data into a train and test set or using the new dataset. Based on the results compared to Elasticsearch, we could say that our local machine learning worked. Our trained machine learning models all have an accuracy of 95 percentage or higher from our various testing except one. Overall, our implementation of the ADSICS environment locally and on the Elastic cloud is expected to have the ability to monitor power grids from Modbus attacks and different kinds of IT attacks.

6 Appendix I - Operation Manual

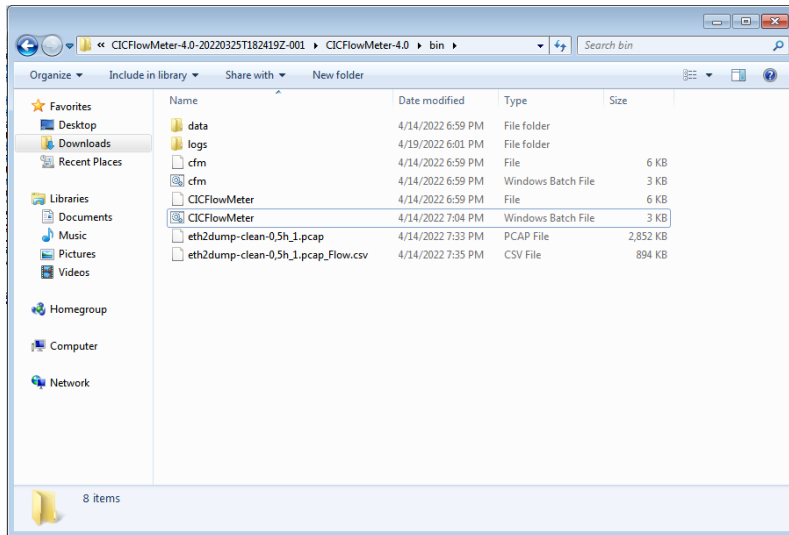
6.1 HOW TO OPERATE LOCAL MACHINE LEARNING ANOMALY DETECTION SYSTEM

1. Operate CICFlowMeter
 - a. Start up the CICFlowMeter.

*CICFlowMeter is used to sniff the current network traffic and extract features from the traffic.

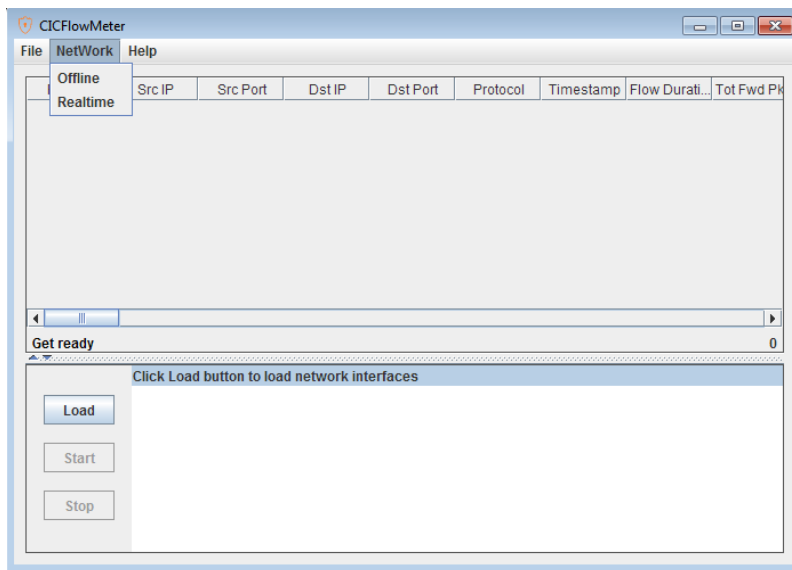
*You need to have **WinPcap** and **JDK** installed on your system to execute this program.

- b. Inside the /CICFlowMeter-4.0/bin execute **CICFlowMeter.bat** to execute related jar files.



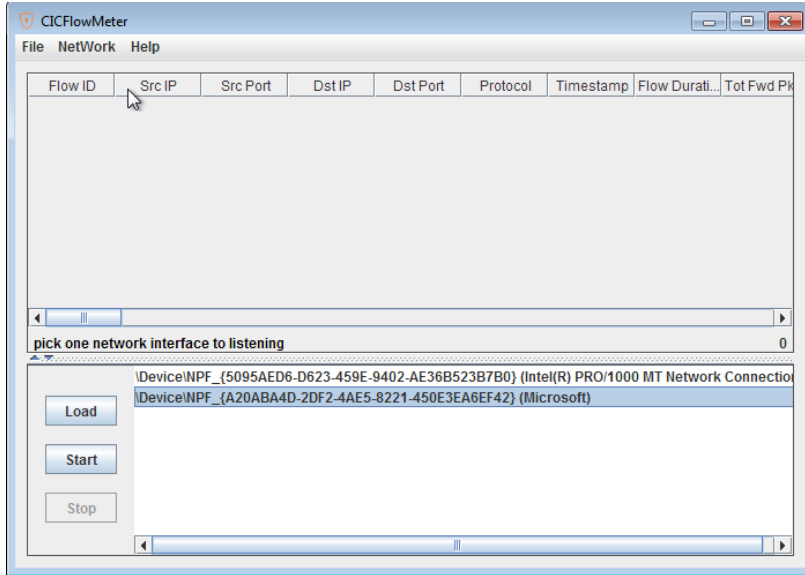
22a Operational Manual - CICFlowMeter

- c. Choose "Realtime" to go into the real time sniffing mode.



22b Operational Manual - CICFlowMeter

- d. List of your device's network interface will show up, choose your intended interface and then click start to start sniffing.



22c Operational Manual - CICFlowMeter

- e. The extracted features will show up in the form of a csv file in the /bin/data/daily folder with corresponding date.

2022-04-23 15:20:09 - path:c:\users\wjddj\Downloads\ADS_ICS-main\ADS_ICS-main\cicflowmeter\CICFlowMeter-4.0-20220325T182419Z-001\CICFlowMeter-4.0\bin\data\daily\2022-04-23_Flow.csv

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W		
1	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Dura	Tot Fwd Pk	Tot Bwd Pk	TotLen Fw	TotLen Bw	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Fwd Pkt Le	Bwd Pkt Le	Bwd Pkt Le	Bwd Pkt Le	Bwd Pkt Le	Flow Byts/	Flow Pkts/	Flow IAT N.Fk	
2	172.27.22-172.27.22	62377	172.27.22	172.27.22	502	6	#####	8793	0	3	0	0	0	0	0	0	0	0	0	0	0	341.1805	4396.5	5
3	172.27.22-172.27.22	62629	172.27.22	172.27.22	502	6	#####	9170	0	3	0	0	0	0	0	0	0	0	0	0	0	327.1538	4585	6
4	172.27.22-172.27.22	62791	172.27.22	172.27.22	502	6	#####	9217	0	3	0	0	0	0	0	0	0	0	0	0	0	325.4855	4608.5	6
5	172.27.22-172.27.22	62934	172.27.22	172.27.22	502	6	#####	9087	0	3	0	0	0	0	0	0	0	0	0	0	0	330.142	4543.5	6
6	172.27.22-172.27.22	63177	172.27.22	172.27.22	502	6	#####	9176	0	3	0	0	0	0	0	0	0	0	0	0	0	326.9398	4588	6

22d Operational Manual - CICFlowMeter

*Following is the list of features and their description.

Features	Description
Flow ID	ID of Flow
Src IP	Source IP
Src Port	Source Port
Dst IP	Destination IP
Dst Port	Destination Port
Protocol	Protocol Used
Timestamp	Timestamp
Flow Duration	Duration of the flow in Microsecond
Tot Fwd Pkts	Total packets in the forward direction
Tot Bwd Pkts	Total packets in the backward direction
TotLen Fwd Pkts	Total size of packet in forward direction
TotLen Bwd Pkts	Total size of packet in backward direction
Fwd Pkt Len Max	Maximum size of packet in forward direction
Fwd Pkt Len Min	Minimum size of packet in forward direction
Fwd Pkt Len Mean	Mean size of packet in forward direction
Fwd Pkt Len Std	Standard deviation size of packet in forward direction
Bwd Pkt Len Max	Maximum size of packet in backward direction
Bwd Pkt Len Min	Minimum size of packet in backward direction
Bwd Pkt Len Mean	Mean size of packet in backward direction
Bwd Pkt Len Std	Standard deviation size of packet in backward direction
Flow Byts/s	Number of flow bytes per second
Flow Pkts/s	Number of flow packets per second
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Fwd IAT Tot	Total time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Max	Maximum time between two packets sent in the forward direction

Features	Description
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Bwd IAT Tot	Total time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Fwd PSH Flags	Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP)
Bwd PSH Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)
Fwd URG Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)
Bwd URG Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)
Fwd Header Len	Total bytes used for headers in the forward direction
Bwd Header Len	Total bytes used for headers in the backward direction
Fwd Pkts/s	Number of forward packets per second
Bwd Pkts/s	Number of backward packets per second
Pkt Len Min	Minimum length of a packet
Pkt Len Max	Maximum length of a packet
Pkt Len Mean	Mean length of a packet
Pkt Len Std	Standard deviation length of a packet
Pkt Len Var	Variance length of a packet
FIN Flag Cnt	Number of packets with FIN
SYN Flag Cnt	Number of packets with SYN
RST Flag Cnt	Number of packets with RST
PSH Flag Cnt	Number of packets with PUSH
ACK Flag Cnt	Number of packets with ACK
URG Flag Cnt	Number of packets with URG
CWE Flag Count	Number of packets with CWR
ECE Flag Cnt	Number of packets with ECE
Down/Up Ratio	Download and upload ratio
Pkt Size Avg	Average size of packet
Fwd Seg Size Avg	Average size observed in the forward direction

Features	Description
Bwd Seg Size Avg	Average size observed in the backward direction
Fwd Byts/b Avg	Average number of bytes bulk rate in the forward direction
Fwd Pkts/b Avg	Average number of packets bulk rate in the forward direction
Fwd Blk Rate Avg	Average number of bulk rate in the forward direction
Bwd Byts/b Avg	Average number of bytes bulk rate in the backward direction
Bwd Pkts/b Avg	Average number of packets bulk rate in the backward direction
Bwd Blk Rate Avg	Average number of bulk rate in the backward direction
Subflow Fwd Pkts	The average number of packets in a sub flow in the forward direction
Subflow Fwd Byts	The average number of bytes in a sub flow in the forward direction
Subflow Bwd Pkts	The average number of packets in a sub flow in the backward direction
Subflow Bwd Byts	The average number of bytes in a sub flow in the backward direction
Init Fwd Win Byts	The total number of bytes sent in initial window in the forward direction
Init Bwd Win Byts	The total number of bytes sent in initial window in the backward direction
Fwd Act Data Pkts	Count of packets with at least 1 byte of TCP data payload in the forward direction
Fwd Seg Size Min	Minimum segment size observed in the forward direction
Active Mean	Minimum time a flow was active before becoming idle
Active Std	Mean time a flow was active before becoming idle
Active Max	Maximum time a flow was active before becoming idle
Active Min	Standard deviation time a flow was active before becoming idle
Idle Mean	Minimum time a flow was idle before becoming active
Idle Std	Mean time a flow was idle before becoming active
Idle Max	Maximum time a flow was idle before becoming active
Idle Min	Standard deviation time a flow was idle before becoming active


23 Operational Manual - List of Extracted Features

- After you get CICFlowMeter running, now you can run the python script import.py to run the code and get the result from the trained machine learning model whether this traffic is anomalous or not.

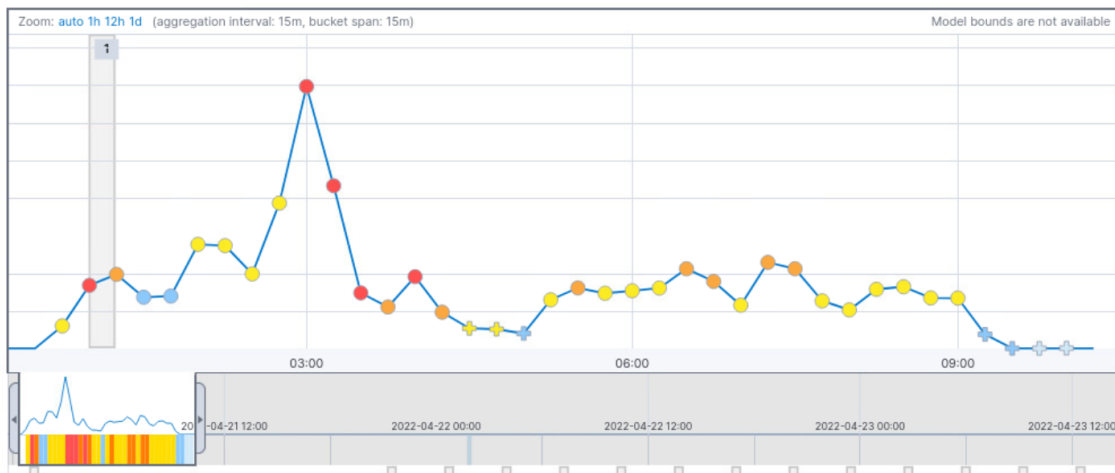
6.2 HOW TO OPERATE ELASTICSEARCH ANOMALY DETECTION SYSTEM

Elasticsearch is already monitoring the virtual machine testbed network.

- When you run the attack script on the testbed, Elasticsearch will detect it as an anomaly and will flag it by severity, red as the most severe.

Single time series analysis of count 

annotations



24 Operational Manual - Elasticsearch Anomaly Detection Result

*In this example, sample network flow of average 2 mb/s started at 1 a.m, Elasticsearch flagged it as most anomalous(red). The main DDoS attack was executed on 2:00 to 3:30, which is flagged red multiple times by Elasticsearch.

7 Appendix II - Alternative Versions

Our project's requirements have changed multiple times since the beginning. Originally, the project was supposed to be about anomaly correlation rather than anomaly detection. We had believed that the project would be provided with an anomaly detection system, and that our role would be to analyze the characteristics of the generated anomalies and design a UI to display that analysis. However, in February after discussion with our client, we determined that it would be more appropriate based on our knowledge and research that we would be better off building an anomaly detection system for this project instead. This shifted our goal greatly.

During the early stages of our work with Elasticsearch, it appeared as though we would need to manually retrieve results from Elasticsearch and display them ourselves. However, after meeting with the Elastic team, we realized that the platform had much more robust tools than we had realized. Eventually, we were able to automate the retrieval and analysis of data within Elasticsearch rather than access those features remotely. Additionally, we believed that we would need to design our own custom interface to display the Elasticsearch results. However, it later became apparent that Elasticsearch's interface would be more than enough for our needs, and we shifted our focus to learning more about its dashboard so that we could customize it to benefit our project.

Our project was originally going to perform all of its analysis on Modbus data. However, Elasticsearch turned out to not have any native support for Modbus data, and such functionality could only be achieved through using third party tools to parse the data in a custom plugin. At that point in the project, we realized that we did not have the time to design a custom plugin for Elasticsearch. After discussion with our client, we began to use only data that Elasticsearch could recognize by default. We were still determined to perform some analysis of Modbus data, so we implemented it locally using a decision tree model. We have been able to test the three kinds of DDoS attacks and it succeeded in classifying which one is normal and which one is an attack. However, it lacked the ability to discern between DDoS attacks, scored low by accuracy.

8 Appendix III - Other Considerations

Most of the team entered this project with little to no knowledge about Elasticsearch, anomaly detection, or machine learning. Nearly all of the technical knowledge needed for this project involved advanced computer science topics or specific tools, meaning that our group spent a great deal of time researching and learning during the year.

Initially, the Thursday dataset appeared useful because it was relatively small and had diverse and interesting anomalies. However, after testing the data we realized that it had very poor performance compared to other datasets like Wednesday because of its skewed state. Since we had spent so much time working with the Thursday dataset, we still decided to include it in this paper despite the issues.

CICFlowMeter, our essential part of sniffing live network traffic and converting it to CSV file so that we can analyze the flow and determine whether it is anomaly or not, was very hard to work with. Because it was developed more than four years ago, and has not been compatibility tested with different operating systems, enabling it to work took a long time. Fortunately, it was pointed out by our advisor that it should run on the Windows 7 system because the access to some of the network functions are blocked in later Windows versions.

Modbus dataset itself came with a PCAP file, and there was no resulting “Label” to test with. The only information we had on the attack was the duration of attack and hours for the whole network traffic. As a result of diligent searching, we figured out the IP address of the attacker and IP address of the victim, eventually enabling us to label the status correctly to benign or attack.

When we first started to test the network packet capture integration with tcpreplay. We kept getting an error packet sent because the DDoS attack message was way too big to send over our network interfaces MTU. We changed ours from the normal rate 1200 to 9000 and we still get issues. When we added in mtu-trunc to our extension and changed tcpreplay to tcpreplay-edit, we were able to truncate all packets down to a normal size and still have everything run at how the network would have operated in real time. In the process of doing so I messed with the ‘-t’ command and it accelerated the time way too much and it finished the network run in one hour instead of eight hours.

Elastic has a lot of incredible features to mess with for system analysis, machine learning, and data manipulation. There is one thing that isn’t included in the ELK stack, and that is the OT protocols for their integrations. We have spoken to the Elastic team and they have stated that there is no support for OT protocols just yet. Since we were not able to do live data traffic with modbus we went with DDoS attack, brute force, and XSS attacks instead to show we can detect anomalies in live network traffic similar to how it would be with OT traffic. We were also able to classify modbus attacks just like the local machine learning model to get results and they were classified with an accuracy of 99%.

A tool that is integrated with Wireshark, editcap is an amazing way to change pcap files to your liking. At first it was hard for us to figure out how to change the start and end times of the massive 13 gig pcap file to the 20 min attack window so we don’t need to replay the full 2 hours of the attack range and create our own network attacks for testing. One issue we had was that it didn’t understand the “DD-MM-YYYYTHH:MM:SS”, the T is supposed to split the date and the time so instead we just removed it and we were able to get the correct pcap file we wanted.

9 Works Cited

Khraisat, A., Gondal, I., Vamplew, P. *et al.* Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecur* 2, 20 (2019). <https://doi.org/10.1186/s42400-019-0038-7>
<https://www.linkedin.com/company/sqrrl>

Intrusion Detection Evaluation Dataset (CIC-IDS2017). *University of New Brunswick*.
<https://www.unb.ca/cic/datasets/ids-2017.html>

Applications - CICFlowMeter (formerly ISCXFlowMeter). *University of New Brunswick*.
<https://www.unb.ca/cic/research/applications.html#CICFlowMeter>

- ahlashkari/CICFlowMeter. <https://github.com/ahlashkari/CICFlowMeter>

Frazão, I. and Pedro Henriques Abreu and Tiago Cruz and Araújo, H. and Simões, P. , "Denial of Service Attacks: Detecting the frailties of machine learning algorithms in the Classification Process", in 13th International Conference on Critical Information Infrastructures Security (CRITIS 2018), ed. Springer, Kaunas, Lithuania, September 24-26, 2018, Springer series on Security and Cryptology , 2018. DOI: 10.1007/978-3-030-05849-4_19

- modbus TCP SCADA #1. https://github.com/tjacruz-dei/ICS_PCAPS/releases

Network datasets, Overview of Network Forensic Tools and Datasets. *Github*.
<https://martinazembjakova.github.io/Network-forensic-tools-taxonomy/network-datasets.html>

Adobe Stock. (n.d.). Safety Lock Icon.

https://as2.ftcdn.net/v2/jpg/01/07/62/09/1000_F_107620948_3lKHqH5pVg3LTBaKSOkAWiQmkgMBfhcY.jpg

Icon Archive. [Harddrive Icon].

<https://icons.iconarchive.com/icons/icon8/ios7/128/Network-Master-icon.png>

The Noun Project. [Sensor Icon]. <https://static.thenounproject.com/png/3001155-200.png>

Shutter Stock. [Network Cloud Icon].

<https://image.shutterstock.com/image-vector/cloud-computing-vector-thin-line-260nw-1684678789.jpg>

Shutterstock. [Modern Machine Learning Icon].

<https://image.shutterstock.com/image-vector/modern-machine-learning-line-icon-260nw-737356600.jpg>

Vector Seek. [Wireshark Logo].

<https://vectorseek.com/wp-content/uploads/2022/02/Wireshark-Logo-Vector.jpg>