Log File Analysis – Attack Detection

Aaron Liske
EMU Game Above College of Engineering & Technology
Ypsilanti, MI - USA

William Smith III
EMU Game Above College of Engineering & Technology
Ypsilanti, MI – USA

Mia Jones
EMU Game Above College of Engineering & Technology
Ypsilanti, MI – USA

Yogesh Chavarkar
EMU Game Above College of Engineering & Technology
Ypsilanti, MI – USA

***Abstract*—The research and sources cover the detection and classification of malware using log file analysis. Our paper covers the overall understanding of system log analysis using various techniques. More so, the sources propose filtering and classification using base classifiers and machine learning (ML) processes. The static and dynamic analysis provides details into malware detection using record files gathered through web browsing, emails, and call logs, among other activities. Some sources explore specific machine learning models such as supply vector machines and extreme gradient boosting. Other articles focus on the feasibility of advanced systems to deal with malware variants. In addition, the research considers basic concept of examining and classifying log files such as window files; the review also provides a comparison of ML learning techniques such as k-NN and Naïve Bayes and their significance in security systems. While mentioning the modern security challenges, the analysis covers the most feared and advanced issues, such as zero-day attacks. Finally, the research provides theoretical and experimental evidence of malware classification using log file analysis. The reader will see that our best value for F1 is 0.92 which is consistent across all algorithms and models.**

Keywords—Machine learning (ML), Log file analysis, classification, malware detection, dynamic analysis, window files, zero-day attacks.

I. Introduction

This project is associated with prediction of cybersecurity attacks on linux based systems that use syslog and have services writing logs to syslog. The initial data normalization covers aspects of analyzing the log structure to extract various fields to classify events and os operations in raw log data. The initial normalization step just creates a data model and dimensions of various operating system events and operations.

This data model is then used for further classifying the os operations and events for building a baseline. To this baseline attack data patterns are added to train the model of what an attack would look like. An iterative learning run would then be able to classify and predict type of attack based on the pattern seen. In subsequent phases additional attack data with different services like SSH, FTP, RPC-NFS etc. are run against the data to analyze the predictability.

A. Dataset

The dataset used is linux syslog data from var/log/messages file in the operating system. The objective behind selecting this data was to identify varying patterns that indicated authentication failures, ftp, http, and nfs-rpc connections indicating possible attacker malicious activity with attempts exploiting buffer flow vulnerability. The sample extract of the syslog file is shown below.

```
Mar 15 00:34:50 combo sshd(pam_unix)[12747]: authentication failure; logname= uid=0
euid=0 tty=NODEVssh ruser= rhost=207.188.80.171  user=root
Mar 15 00:34:50 combo sshd(pam_unix)[12749]: authentication failure; logname= uid=0
euid=0 tty=NODEVssh ruser= rhost=207.188.80.171  user=root
Mar 16 05:39:05 combo rpc.statd[1601]: gethostbyname error for
^X+ÿ¿^X+ÿ¿^Z+ÿ¿^Z+ÿ¿%8x%8x%8x%8x%8x%8x%8x%8x%8x%62716x%hn%51859x%hn\220\220\220\220\22
0\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\
220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\22
0\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\220\
Jun 11 03:28:24 combo ftpd[4321]: connection from 209.184.7.130 () at Sat Jun 11
03:28:24 2005
Jun 11 03:28:24 combo ftpd[4322]: connection from 209.184.7.130 () at Sat Jun 11
03:28:24 2005
Jun 11 03:28:25 combo ftpd[4323]: connection from 209.184.7.130 () at Sat Jun 11
03:28:25 2005
Nov 15 04:03:40 combo su(pam_unix)[29849]: session opened for user cyrus by (uid=0)
Nov 15 04:03:40 combo su(pam_unix)[29849]: session closed for user cyrus
Nov 15 04:03:41 combo logrotate: ALERT exited abnormally with [1]
Nov 15 04:30:29 combo kernel: Out of Memory: Killed process 28473 (httpd).
Nov 15 04:35:29 combo kernel: Out of Memory: Killed process 28474 (httpd).
Nov 15 04:35:42 combo kernel: Out of Memory: Killed process 28475 (httpd)
```

Fig 1. Basic Syslog File Example

B. Code and Scripts

For initial review the data was mounted in a linux vm and ingested into a log analytics workspace as in EMU Azure Tenant. It was quickly realized that this would not have been a good solution for our data and we outgrew that. A rudimentary data classification test was performed with the query outlined below in the Azure space.

```
linuxlogs_CL
| extend result_CF = case(
                message_CF    contains
"Session","Access",
        message_CF  contains  "mount
Wrapper","Process",
         message_CF  contains  "User
Slice","session refresh",
    "Unknown or blank"
)
|    project    time_CF,    host_CF,
service_CF, message_CF, result_CF
```

Fig 2. Initial Code in Azure for Data Classification

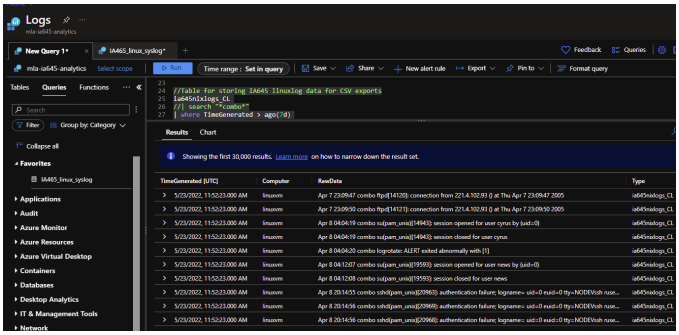

Fig 3. The workspace within Azure

For the CSV extraction phase extract, transform, load ("ETL") scripts were used to standardize the data formats from the log files.

After using Azure for the initial dataset tests, proprietary code was written in Python to process the log files that we had amassed for this project. This allowed us to not only add more log files in the future without making further changes to our environment, but it also afforded the flexibility of being able to quickly churn through the data.

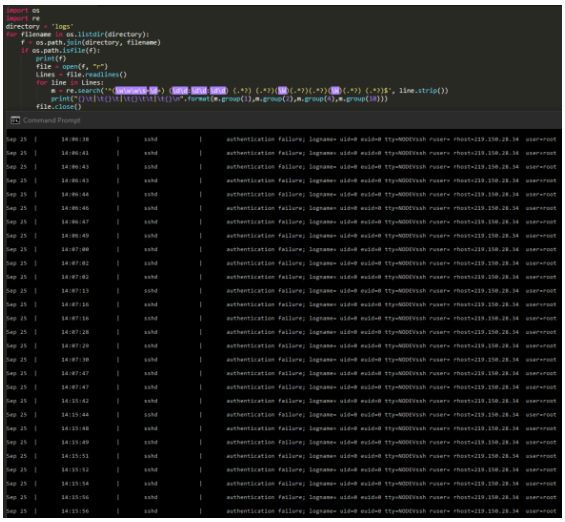

Fig 4. Python and CLI workspace

C. Classifications

The dataset was used to identify the possible patterns in data and define objectives that could be used for machine learning. The aspects displayed included using date field to define it being a weekday or a weekend. Another parameter that was considered was time to classify if it was morning, afternoon, evening or night. Service defined in the syslog file like sshd, ftp, rpc, http was used as another parameter. User type root with uid=0 or another user with uid > 0 was considered as another parameter.

Finally for the classification label, the information from research research articles was used to identify malware techniques, tactics and procedures used during attacks. These behaviors were visible in the dataset in the form of malicious activity in bruteforce, log deletion, vulnerability exploitation etc.

Overall the data from the syslog files would be classified into the attributes and fields below.

- `Type of Day`
- `"Time of Day`
- `User Type`
- `Service`
- `Classification Label(suspicious / not suspicious)`

The method for classifying the data, in the python ETL scripts is laid out in Figure 5 below. First, the data is standardized, with key elements pulled from each message in the log file. Next, the algorithm looks at the data, then determines whether or not it was a potential attack. Finally, the data is written to a CSV file for feeding into Weka for the machine learning training and analysis.

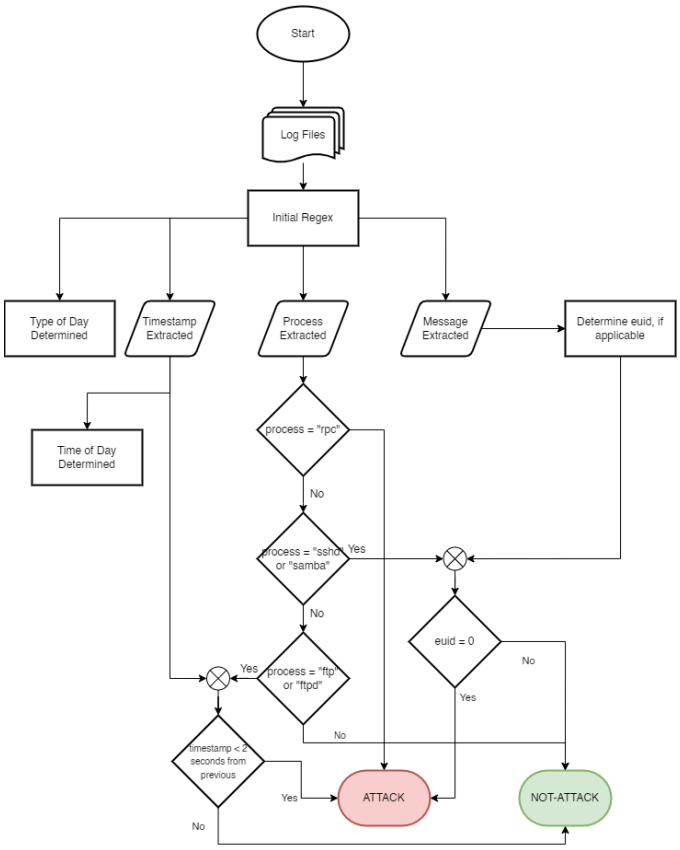

Fig 5. The basic ETL algorithm for classification of the log files used, for machine learning.

D. Tools & Methods

Several data review and implementation tools were used in the project including cloud platforms and locally installed tools on laptops or home computers.

The data files were found from publicly accessible repositories that were meant for cybersecurity research. These logs became the cornerstone of our dataset because they were known to have malicious activity in them and it greatly reduced the time of organically gathering the data through conventional methods.

Following the collection of the datasets, there was manual review to determine the feasibility of their use with the project and to lay the groundwork for the creation of the ETL scripts as well as the final learning CSV file for the dataset. For this initial purpose, an Azure workspace was set up and some scripts were run, using keywords and their inclusion in the log file messages to determine the type of potential message it was. These elementary scripts and queries were the proof of concept that drove the use of Python for creating the final training CSV file.

A SQL database was briefly used for the storage of the data, but it was quickly determined that given the limited dataset and the overhead of running SQL Server as well as the machine learning, it created a resource bottleneck that hampered progress more than it helped. For this reason, a flat CSV file was used in the final iteration. A larger dataset may justify the use of SQL server in the future, or perhaps more complicated processing of the server logs.

SQL database was used for mounting data and classifying based on date, service, user and attack activity. The processed data was exported as a data.csv to validate the processed data. Finally for ingestion into Weka software the raw data field was removed from the data.csv file. The file Weka Dataset.csv was exported with only the classified fields for machine learning.

E. Logfile Analysis

The first step of getting the raw log files into a machine learning algorithm friendly format was to pull out any relevant data and create a CSV file from them. It was determined that the type of day (working or non-working), time of day, system process, and user level (if applicable) were the most important metrics to build our dataset from for machine learning to determine suspicion.

This log processing was done with a python script that used regex to pull each log message apart to find the relevant sections for processing.

```
^(\w\w\w\s+\d+) (\d\d:\d\d:\d\d) (.*?)
(.*?)(\W)(.*?)(.*?)(\W)(.*?) (.*?)$
```

Fig 6. Regex python script I

The main match groups used are the date, time, process, and the message. Depending on the message, it is further matched with another regex to determine the euid, or user level, used for the process. To determine the working versus non-working days, Saturdays and Sundays were considered non-working, and Monday through Friday were considered working. Holidays are not taken into account.
Time of day was determined through the following block of code, using the 24 hour time stamp from the log file:

```
if hour >= 0 and hour <= 6:
      time_of_day = 'early morning'
if hour >= 7 and hour <= 11:
      time_of_day = 'morning'
if hour >= 12 and hour <=17:
      time_of_day = 'afternoon'
if hour >= 18 and hour <= 21:
      time_of_day = 'evening'
if hour > 21:
      time_of_day = 'night'
```

Fig 7. Regex python script II

II. LITERATURE REVIEW

In “Big Data Analysis for Log File and Malware. International Journal of Scientific Development and Research” Nguyen et al. (2020) demonstrate the importance of log file analysis in cyber security, especially in mobile and IoT applications. Given that most digital devices use apps, the world needs a more convenient means of filtering data and detecting possible vulnerabilities and malware. Therefore, the group proposes using system log files from mobile devices and IoT devices for malware predictive analytics. They utilize hand-crafted features such as TF-IDF and machine learning models to perform the analysis, including Extreme Gradient

Boosting and Support Vector Machines. In the end, Nguyen et al. (2020) developed an autonomous system that perceives abnormalities in IoT devices.

Nguyen provides excellent illustrations of how system logs can provide superior performance in malware predictive analytics. Furthermore, their use of machine learning and hand-crafted features creates an autonomous system that reduces costs for organizations since the system runs with minimal human supervision after the initial launch. More so, the article provides a comprehensive exploration of how to secure IoT systems, especially for micro-sensory networks like IoT devices and mobile phones.

The article from "Early-stage malware prediction using recurrent neural networks. Computers & Security" goes over the use of Machine Learning algorithms with Command and Control style of malware and some of the challenges involved with these algorithms. Some of these challenges include cost, loss of performance on the host systems, and the lack of data for modeling because of the ever-changing nature of the C&C Malware. One item that this article covers, that others tend to lack in is that it tells that being initially infiltrated with malware is not only easy to do, but it's difficult to detect the initial attack. This article then goes into depth about what to do after the initial attack, with detection and exfiltration using Machine Learning techniques.

Paper "A heuristics approach to mine behavioral data logs in mobile malware detection system, Data & Knowledge Engineering" reviews using Random Forest and Gradient Boosting algorithms to test for malware. The testing finds the Random Forest slightly more accurate than the Gradient boosting. The article continues to walk through uploading a sample into the program and processing through a simple program that identifies the likelihood of a malicious sample. This was a more simplistic approach than the other two ML papers reviewed.

Prevention is an ideal goal but detection is the most practical and a must. Also if prevention is not fine tuned it may not be as effective of an approach, as it could backfire and lead to denial of service due to false positives. "Securing Control and Data Planes From Reconnaissance Attacks Using Distributed Shadow Controllers, Reactive and Proactive Approaches," asserts that anomaly based intrusion detection is more effective for hunting threats in various logs and data sets. Some basic questions can help define the contextual approach - Who or what type of adversaries are after the data and organization? What is the motivation and their capabilities? How vulnerable are the assets against the adversarial capabilities?

This approach provides a hypothesis for determining the context to define a possible threat hunting exercise. This helps identify where the intrusion can be observed and how it can be detected from available data analysis. The overall behavior and progression needs to be laid out and validated against available data by learning patterns for detecting anomalies from cyber threat intelligence. This helps with proactive planning for emerging threats and APT waves.

Further use of planned red teaming helps to test the effectiveness of the process and generate real life results for machine learning and building efficient models. Future scope is for continuous adaptation to track evolving threats with novel exploits and controlling attack surface of exposed assets.

III. Proposed model

The proposed model covers collecting data with specific event criteria, cleaning it, and feeding It to the machine learning model. It also shows the workflow diagram of the different stages of work in the project.

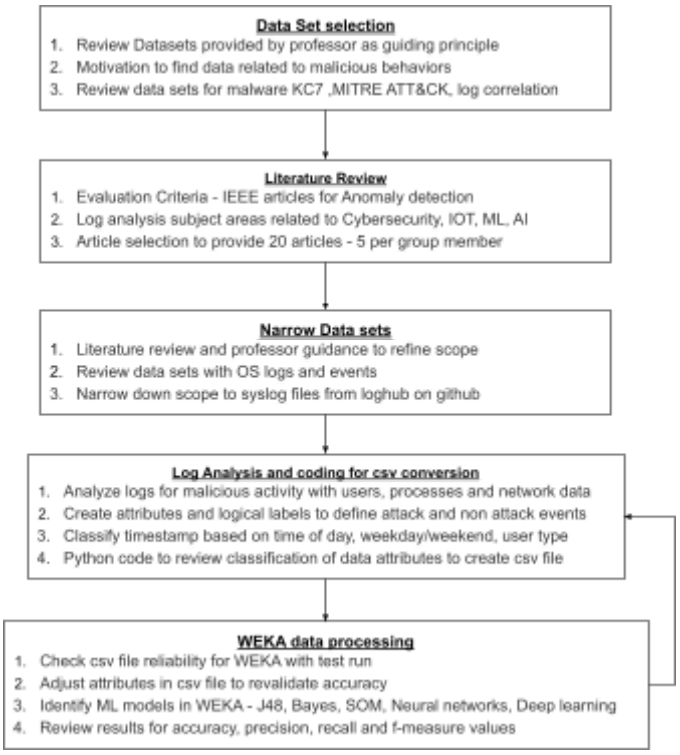

Fig 8. Workflow diagram

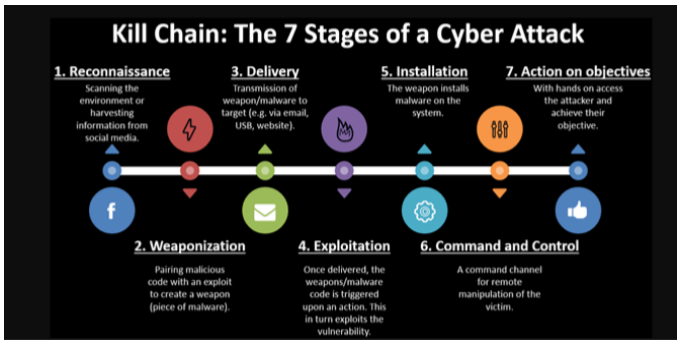

Fig 9. Kill Chain: The 7 Stages of a Cyber Attack

IV. Weka results

A. Dataset Summary

Data Type: Linux syslog files labeled for suspicious activity
File Type:CSV
Name: WEKA Dataset
Instances:71704
Attributes:5

TYPE_OF_DAY
TIME_OF_DAY
PROCESS
SUSPICIOUS

B. Algorithms

TABLE I. Decision Tree

Accuracy	Precision	Recall	F1-Measure
90.03	1.0, 0.77, 0.923	0.850, 1.0, 0.9	0.92, 0.87, 0.90
Inference: Folds 2, 5, and 10 with the same accuracy. Split 66% is slightly higher with 93.9% accuracy.			

TABLE II. Naive Bayes

Accuracy	Precision	Recall	F1-Measure
90.02	1.0, 77, .923	0.850, 1.0, 090	0.919, 0.870, 0.903
Inference: Folds 2, 5, and 10 with the same accuracy. Split 66% is slightly higher with 90.32% accuracy			

TABLE III. Bayes Net

Accuracy	Precision	Recall	F1-Measure
90.02	1.0, 0.77, 0.92	0.85, 1.0, 0.90	0.92, 0.87, 0.90
Inference: Folds 2, 5, and 10 with the same accuracy. Split 66% is slightly higher with 90.32% accuracy			

TABLE IV. Random Tree

Accuracy	Precision	Recall	F1-Measure
90.03	1.0, 0.770, 0.92	0.85, 1.0, 0.900	0.92, 0.87, 0.90
Inference: Folds 2, 5, and 10 with the same accuracy. Split 66% is slightly higher with 90.32% accuracy			

TABLE V. SMO Support Vector Machine

Accuracy	Precision	Recall	F1-Measure
90.03	1.0, 0.77, 0.92	0.85, 1.0, 0.9	0.92, 0.87, 0.90
Inference: Folds 2, 5, and 10 with the same accuracy. Split 66% is slightly higher with 90.32% accuracy			

TABLE VI. Multilayer Perceptron (Deep Learning Neural Network)

Accuracy	Precision	Recall	F1-Measure
90.03	1.0, 0.77, .92	0.95, 1.0, 0.9	0.92, 0.87, 0.90
Inference: Similar results using the "Use training set" test option.			

C. Results and observations

Reruns and revalidations in Weka were performed with members from the team. This was done for checking consistency of the results. Dr Darwish validated the dataset too in his local environment with the same consistent results validating the dataset.

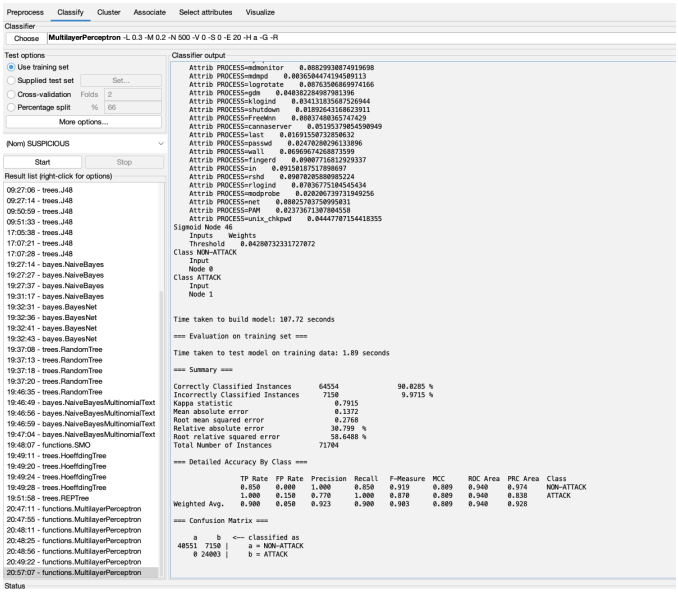

Fig 10. Deep learning multilayer perceptron

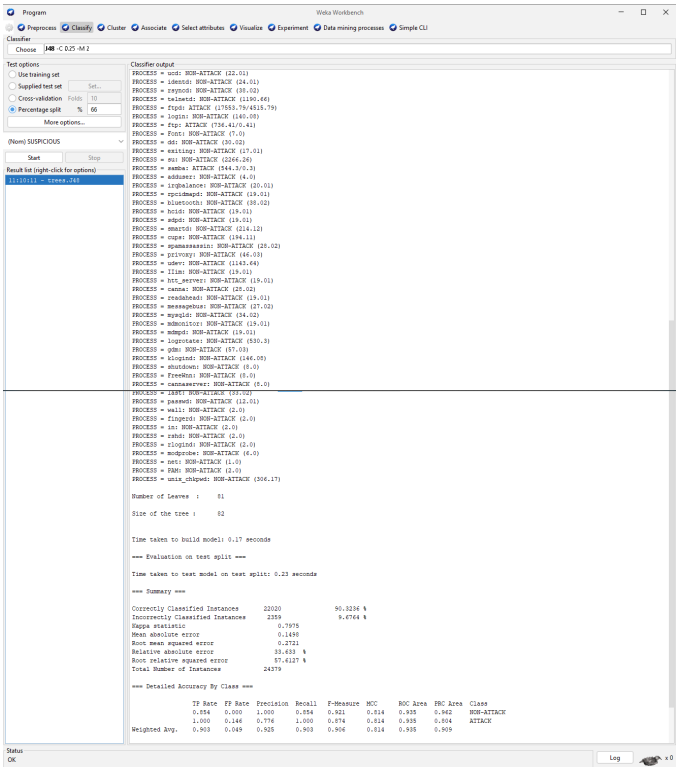

Fig 11. Decision tree - J48 results

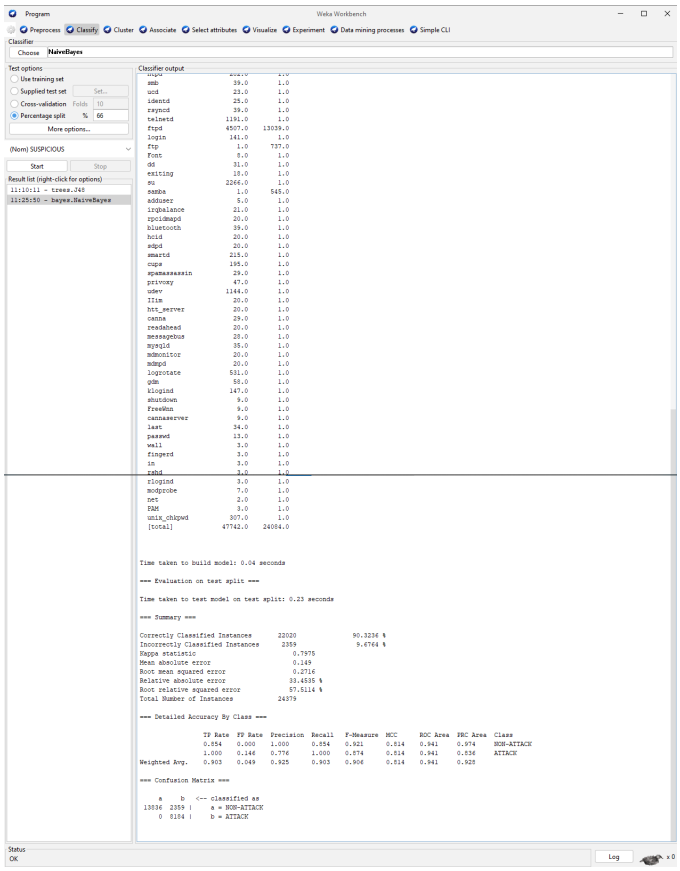

Fig 12. Naive bayes results

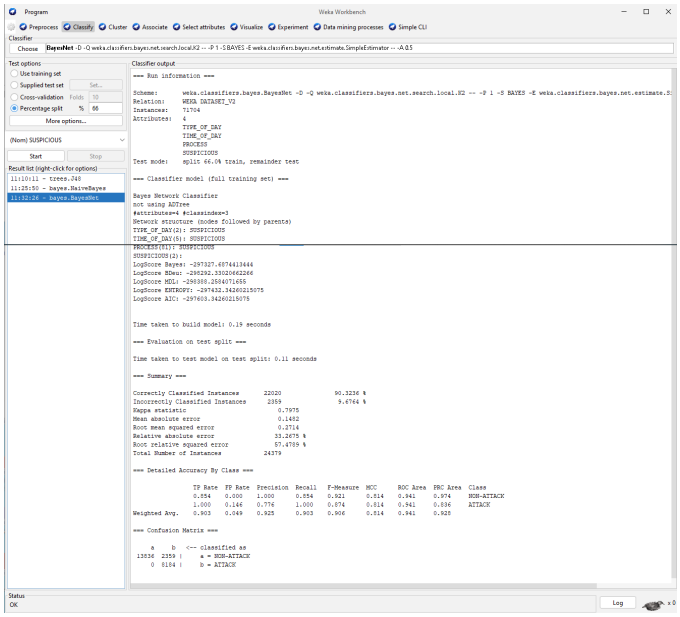

Fig 13. Bayes net results

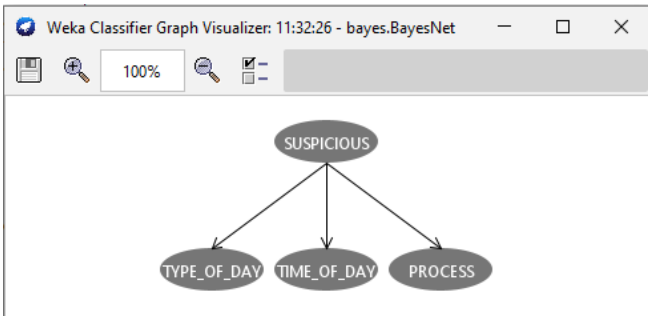

Fig 14. Bayes net graph

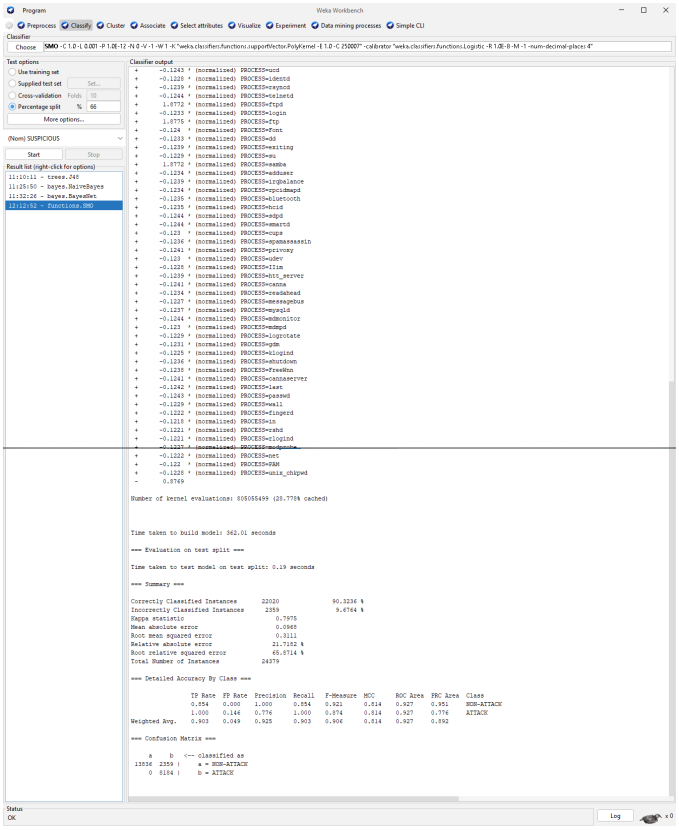

Fig 15. SMO Results

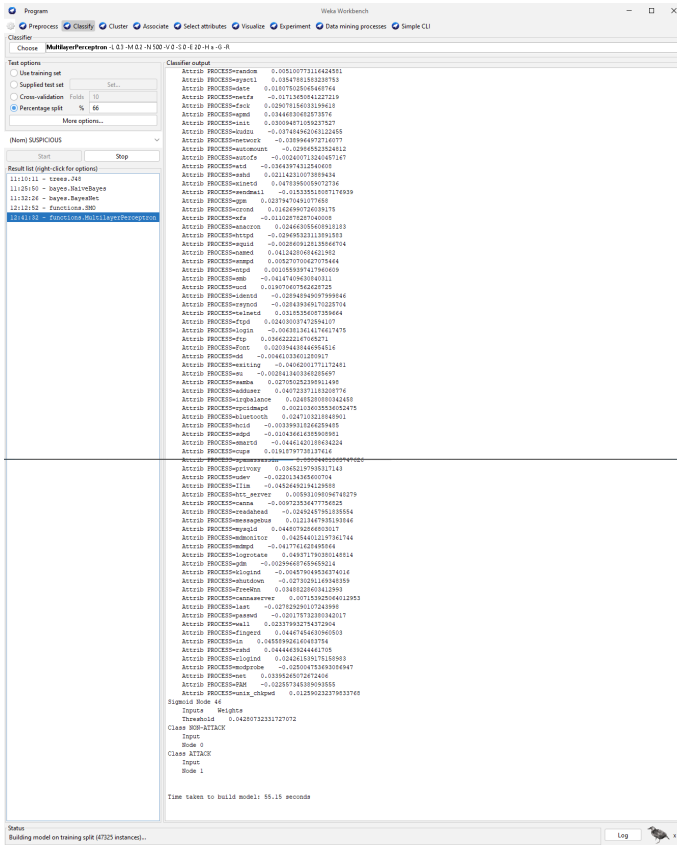

Fig 16. Deep learning - Multilayer Perceptron results

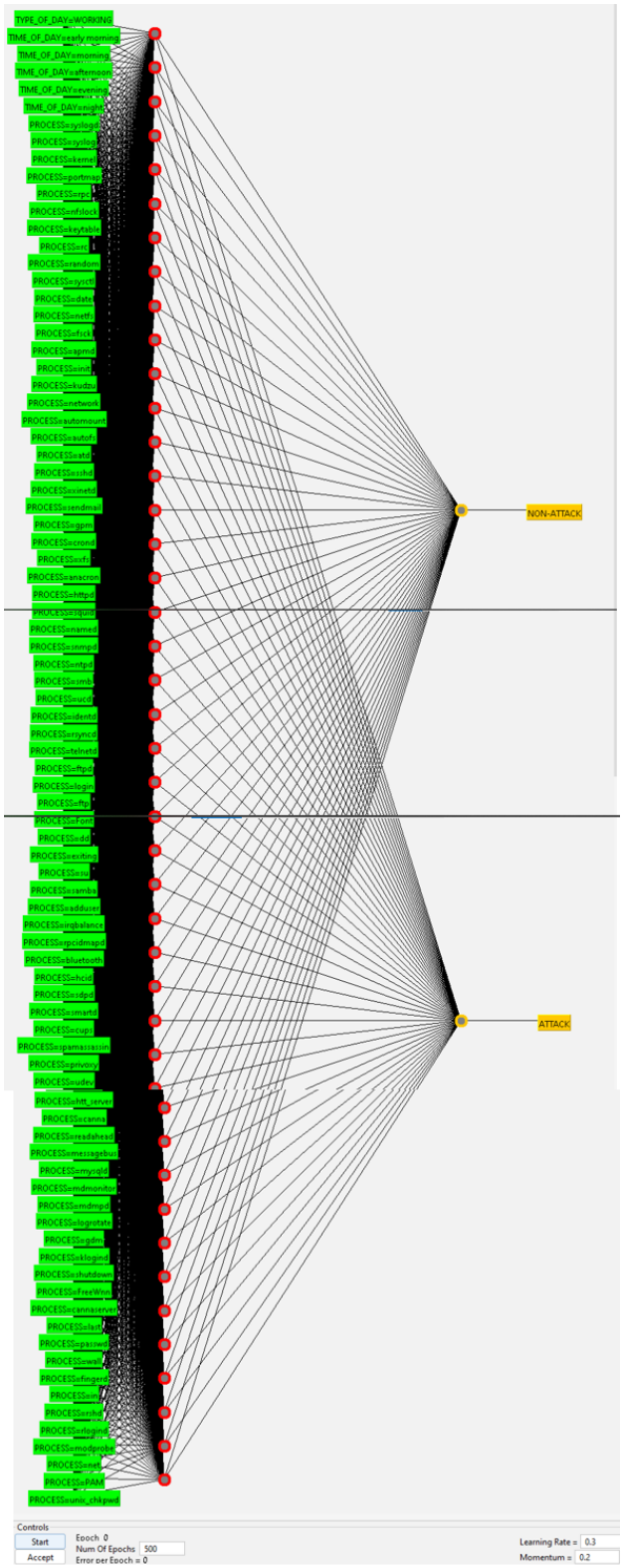

Fig 17. Neural network

V. Conclusion

In closing, the code provided for the machine learning tool can be used as a “stepping stone" to developing a system that uses the prepped data and ML Algs to predict attacks in near-real time. It will help linux based systems learn from

behaviors and patterns, and not depend on hard coded signatures that could become outdated in the constantly changing world of cyberattacks.

The real value of expanding future work would come from adding behaviors for services like squid proxy logs and BinDNS logs. It will help classify outbound connections from compromised systems as a continuation of the attack progression to command & control Kill Chain 6 level. That will provide the system owner the last opportunity to foil the attacker reaching the final level of the Kill Chain. And killing the chance of attackers achieving the objective to gain system access.

VI. Future work

Considering future prospects for this tool, it can effectively work in two different cybersecurity functions -

- Compliance using syslogs showing exposure of misconfigured assets
- Attack prediction for threat hunting and collecting Cyber threat Intelligence from observed attack campaigns

As the machine learning picks up these patterns it can predict a possibility of an attack progression based on services running on a linux system. It can then provide a risk assessment report with possible recommendations. The data used for this experimental project was obtained from logpai loghub repository. In the example project that was carried out as part of IA645 Machine Learning for Cybersecurity at Eastern Michigan University, the team of four students worked on this project with guidance from faculty.

VII. Reference

[1] Nehe, S. M. (2016). Big Data Analysis for Log File and Malware. International Journal of Scientific Development and Research, 1(7), 140-145.

[2] Rhode, M., Burnap, P., & Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. Computers & Security, 77, 578–594. https://doi.org/10.1016/j.cose.2018.05.010

[3] Giang Nguyen, Binh Minh Nguyen, Dang Tran, Ladislav Hluchy,A heuristics approach to mine behavioural data logs in mobile malware detection system, Data & Knowledge Engineering

[4] M. F. Hyder and M. A. Ismail, "Securing Control and Data Planes From Reconnaissance Attacks Using Distributed Shadow Controllers, Reactive and Proactive Approaches," in IEEE Access, vol. 9, pp. 21881-21894, 2021, doi: 10.1109/ACCESS.2021.3055577.
https://ieeexplore.ieee.org/document/9340181

[5] Halvorsen, J., Waite, J., & Hahn, A. (2019). Evaluating the observability of Network Security Monitoring Strategies with tomato. IEEE Access, 7, 108304–108315. https://doi.org/10.1109/access.2019.2933415

[6] Jadidi, Z., & Lu, Y. (2021). A threat hunting framework for industrial control systems. IEEE Access, 9, 164118–164130. https://doi.org/10.1109/access.2021.3133260

[7] Ajmal, A. B., Shah, M. A., Maple, C., Asghar, M. N., & Islam, S. U. (2021). Offensive security: Towards proactive threat hunting via adversary emulation. IEEE Access, 9, 126023–126033. https://doi.org/10.1109/access.2021.3104260

[8] Skopik, F., Wurzenberger, M., & Landauer, M. (2021). The seven golden principles of effective anomaly-based intrusion detection. IEEE Security & Privacy, 19(5), 36–45. https://doi.org/10.1109/msec.2021.3090444

[9] Hyder, M. F., & Ismail, M. A. (2021). Securing control and data planes from reconnaissance attacks using distributed shadow controllers, reactive and proactive approaches. IEEE Access, 9, 21881–21894. https://doi.org/10.1109/access.2021.3055577