PROJECT INFORMATION

Project Title	**A Practical Introduction to Applying Machine Learning to Malware Detection**		
Technology Area	**Machine Learning**		
Project Team	**Name of Team Members**		**Main Responsibilities**
		Gaston Carvallo	Script Development
		Loyd Rafols	Networking/Infrastructure
Keywords (max. 4)	**1. Machine Learning**		**2. Malware Detection**
	3. Educational		

Abstract

Malicious attacks are one of the most damaging and expensive threats organizations face. Machine learning methods have shown great potential toward the problem of an increasing number of variants through there are still a number of challenges to implement these methods in production environments that require further research. Yet, in our experience there seem to be a tendency for students and beginners to treat it as an inscrutable topic reserved for only the selected few. We propose creating a practical introduction to the subject that can illustrate how machine learning works, helping to demystify it and serve as a base for further research and learning.

1. INTRODUCTION

Malware is one of the fastest growing cybersecurity threats that individuals and organization face on a regular basis. The constant evolution of malware proves to be an ongoing issue for

solutions that attempt to stop its attacks, as relying on human-based analysis proves to be more infeasible day by day.

The current state of detecting malware through network-based and host-based methods through machine learning is well-developed, but tend to look at each aspect in isolation. With the majority of malware payloads taking place through the network and/or through the compromised host, it is important to consider both types when implementing a malware detection scheme. Information is missed when only considering one aspect, which may be crucial in identifying that a malware attack is underway or has occurred, as well as estimating the damage done by the attack.

The number of recent disruptive ransomware attacks is growing significantly (Cook). Symantec claims to have found over 186,000 new ransomware variants in 2018 alone (Symantec, 38). One notable victim of these malware attacks was the UK's National Health Services, which cost them over $100 million in damages (Brunau) because it failed to detect the malware traversing their network and the encryption it did on the infected systems.

In 2020, the average cost for a destructive malware breach was $4.52 million, and the average cost for a ransomware breach was $4.44 million (IBM, 44). As the number of malware variants keeps increasing, detecting malware using machine learning is thought to be one the most promising ways to help with this issues and present significant amount of research in the field. And yet in our experience most students in information security have no practical experience on how theses method work.

We propose the creation of a curriculum that includes practical labs describing the steps we needed to take in order to create a machine learning model to detect malware. That can serve as a practical introduction the use of machine learning in security.

2. PROJECT OBJECTIVES

The main objective is to teach beginners about implementing machine learning in malware detection. We will create a machine learning model to detect Metasploit malicious payloads to illustrate the process. We will create our own dataset by generating the malicious samples using the Metasploit Framework, extract the features using CAPA[1], training 4 classifiers, J48, Support Vector Machine (SVM), and Naïve Bayes. The best classifier will be implemented to classify live traffic and recording its result to a log. Since our goal is to illustrate the practical aspects in applying Machine Learning to security so we will create accompanying material illustrating in detail how the data was gathered, and the model was created and applied.

3. LITERATURE REVIEW

Malware detection has traditionally been classified in static and dynamic analysis. Static analysis looks at the source code of the malware in isolation. Signature based detection is one of the main approaches to detect malware in this manner, while it can be fast and efficient for known malware it is ineffective for novel attacks and is susceptible to obfuscation attempts, like making changes to the source code or encrypting the file (Aslan and Samet, 6253). Dynamic analysis, on the other hand looks how the malware behaves, e.g., what system call it makes or how it changes the filesystem. While network analysis can be considered a subset of dynamic analysis, Manzano, Meneses and Leger (1) instead propose to classify detection methods as host-based and network-based contexts.

In the literature, we noticed one certain common limitation between a majority of our papers. That limitation was that each paper or model was good at detecting one certain characteristic of a

[1] CAPA is an open-source malware analysis tool.

malware, however, they were poor at detecting other characteristics, or omit them entirely. There is little overlap in terms of a detecting both network-based and host-based features of malware in a hybrid malware detection model, and thus in a realistic scenario, would excel at detecting one type or family of malware, but would fail at detecting another.

3.1 Network-based Detection

Some malware families require a connection to a command and control server[2] in order to grab data needed for delivering its payload. After the victim is infected, it establishes a connection to a server under the attacker's control. Through this connection, the attacker can issue direct commands to the malware and extract data. Researchers have proposed different methods that seek to determine the presence of a malware by trying to detect and classify these connections.

Modern malware tends to use DGAs[3] to establish a channel to its C2 server using subdomains instead of hard coded IPs to prevent defenders from blocking the specific IP or domain used by a family of malware.

Salehi and others (6) studied and showed success in detecting ransomwares based on their use of DGAs for subdomains. They identified 3 classes of features: gibberish domains, the frequency of requests to different domains and re-generation of domains by the algorithm. Their detection engine is supplemented by a black/white list module to reduce false positives. Zhang utilized deep learning algorithms to use one-hot encoding[4] for their DGA detection features (Zhang, 464).

Most of the research for detecting DGAs is under the assumption that the traffic is in plain text (Patsakis et al. 101616) however there are several protocols being evaluated to offer encrypted

2 Also known as C2 or C&C - a method of controlling multiple infected hosts through a centralized server.
[3] Domain Generation Algorithm - instead of using a static IP to create a C2 channel, pseudorandom generated subdomain names are used.
[4] A method of encoding non-rankable items into a numeric order (such as colors)

DNS services. These approaches are good at detecting network-based feature of malware, but their limitation is around their easiness of tampering by attackers.

Patsakis and others (101620) developed indicators of compromise that could distinguish legitimate DNS from those generated by a malware DGA. They identified that DGAs tend to generate domains of similar length and therefore the response packets tend to be similar in length, they also noticed that DGAs queries have a cyclical component that is possible to detect through a statistical analysis (Patsakis et al. 101618). While these methods might work currently, the behaviors seem to be easily modified by attackers.

Research has also been conducted in detecting malware directly through its network traffic. Zhu and others (1008) proposed a model to detect Remote Access Trojans[5] that looks at the TCP[6] headers, they selected 4 features based on RAT's different traffic pattern. For example, benign applications tend to send as much data as possible as soon as the connection is established, RATs might show what they called early-stage, a period of time where noticeable idle time is present between packets (Zhu et al. 1008). This model has a good baseline for detecting RATs, and can be modified to detect C2 traffic for malware.

Alhawi, and others (98) proposed a model to detect ransomware on Windows machines called NetConverse. They manually selected 13 features from traffic conversations[7] but their model cannot detect ransomware using real-time data.

In contrast, Almashhadani and others (47063) created a working prototype with two network detectors, one packet-based and a second flow-based for the Locky ransomware family. The features were selected both manually and through the WEKA[8] feature selection tool. The features revolved

[5] Also known as RATs - allows an attacker to remotely control a machine over a network or the Internet
[6] Transport Control Protocol - used for transporting data over a network, the internet.
[7] Defined as the bidirectional traffic for a 5-tuple flow (from an source ip:port to a destination ip:port on the same protocol)
[8] Waikato Environment for Knowledge Analysis, an open-source data mining and machine learning tool

around 3 aspects of the network traffic: a distinguishable use of RST, ACK-flagged[9] packets to terminate connections, its use of POST[10] requests and DGA-generated subdomains. This paper provides a good basis for network feature-based detection of malware, but does not test other malware families aside from the Locky ransomware family.

In order to obfuscate their presence, some malware variants encrypt their traffic. Premrn explores creating a device capable of detecting encrypted C2 channels using a machine learning model (Premrn, 5). They manually selected 6 features from the connection logs (instead of traffic capture) (Premrn, 54). Their model presented a high False Positive Rate which would make it unsuitable for day-to-day operations, so, they proposed integrating it with some kind of IP whitelisting to reduce false positives (Premrn, 90).

Modi (6) also explored detecting malware through encrypted traffic, instead of just using connection statistics they also selected features related to the TLS[11] hand-shake and the certificate used (Modi, 35). They propose to increase the model efficiency by adding an additional detector of DGAs (Modi, 68). Overall, their model is limited in capability because it can only classify if the sample is ransomware or not, and it cannot perform multiclass classification to attribute the ransomware to a specific family or as a general malware.

In summary, the approaches we reviewed have the limitation in that they do not account for host-based features, so if malware was to exist on a host that does not communicate with an external host, this approach would be ineffective.

[9] Flags used to terminate a TCP connection - stands for Reset, Acknowledge
[10] POST is one the methods used in for HTTP traffic
[11] Transport Layer Security - a protocol that encrypts internet traffic

3.2 Host-based Detection

While most ransomware families need to contact their C2 server, about a third do not require C2 traffic, in such cases detecting it through network traffic is not viable (Berrueta et al. 144929). Host-based methods are also harder to evade, while attackers can and do change malware behavior to obfuscate their presence, ultimately there are action the malware need to perform to accomplish its objective which cannot be hidden (Almashhadani et al. 47057).

Arabo and others (291) proposed a system to detect ransomware that used two detection modules: One that uses machine learning and the other based on manually configured thresholds. The machine learning features were selected around the malware resource usage (CPU, RAM and disk access). Their machine learning model was only partially successful in detecting the ransomware (Arabo et al. 294), as it does not consider if the malware was not particularly resource intensive, or used other resources such as networking.

Bae, Lee and Im (3) explored using machine learning to detect ransomware through Windows Native API[12] invocation sequences when a file is executed (Bae et al. 4). They proposed a classification model called Class Frequency - Non-Class Frequency (CF-NCF). This classification model focuses around how many times something shows up in a certain class (benign, malware and ransomware), instead of the traditional Term Frequency - Inverse Document Frequency that looks how many times the term shows up in a document (Bae et al. 4). This approach is limited as it does not utilize other API function calls for malware detection, as well as not utilizing network-based features on the host for C2 detection.

In comparison to the previous paper, Hirano and Kobayashi (1) proposed a framework to detect ransomware that collects I/O requests through a hypervisor[13] instead of the OS to make the

[12] Application Programming Interface - a means for software to allow interaction with itself through predefined functions or tasks.
[13] A means of running one or several virtual computers on one or more physical computers

framework portable. They selected 5 features related to the read/write characteristic of the encryption process ransomware use (Hirano and Kobayashi, 4). This enables usage with any operating system instead of just Windows exclusively in the previous paper.

Some researchers select their features manually, according to their knowledge of the dataset and the malware behavior, others however use automated tools to extract numerous raw features from the system and then use an automated algorithm (heuristics), such the Chi-squared test method[14] and fine tune the final feature set used by their machine learning model.

For example, Sethi and others (1) put forward a framework where raw features are extracted from a sandbox's report when a file is executed and then the chi-squared test is used to select features for the detection model, they create two models, the first one classifies the executable in benign/malware, when a malware is detected a second model classify the malware family (Sethi et al. 3). This approach to host-based malware detection provides a good framework for the host-based component of our hybrid malware detection model, and future work can be done using other malware families and network-based features.

Shhadat and others (918) looked at the impact the heuristics can have in the model accuracy. They expanded on the work of Chumachenko that used a similar framework than Sethi of extracting features from a sandbox. Shhadat et al. (919) used a different heuristic to select the features (cross-validation). While the models that used decision-tree and Random forest saw no significant change, models using Naïve Bayes saw significant improvement (Shhadat et al. 922).

Jethva (6) suggested a hybrid host-based malware detection model. Their solution has two detectors, one based on a ML model using heuristics (chi-squared test) to narrow the features (Jethva, 43); and the other based on the combination of file entropy (encrypted files show higher

[14] A test used to determine the differences between a theoretical model and actual data, in this case used to refine the accuracy of the model

entropy) and the presence of file signatures (magic numbers) to help distinguish benign compressed files that also show high entropy (Jethva, 34).

The lack of labelling datasets may prove to be a limitation on research datasets. Noorbehbahani and Saberi (24) looked into the use of semi-supervised methods. They used 5 supervised heuristics to extract the feature set and then used semi-supervised classifiers to identify ransomware (Noorbehbahani and Saberi, 25). The main limitation of this approach is that the utilized unsupervised feature selection accuracy was very poor, which makes it unfeasible to use by itself and would improve by implementing it in a hybrid malware detection scheme.

In summary, the literature around host-based detection mainly suggests that there is a limitation with most approaches in that only host-based features are considered, whereas the detection rate would improve if network-based features were also implemented.

4. DESCRIPTION OF THE PROPOSED WORK

We have break down the project in 5 different objectives, creating the dataset, selecting a ML model, implementing the selected model, creating the educational elements, and writing the final report (Table 1).

The dataset will be created by generating malicious payloads and selecting random benign files and hosting them on a web server. From there, a target machine will download the files while a monitoring server will capture the traffic and extract the binaries from it. Finally, the binaries will be processed to extract the features, and the data will be cleaned and labelled (Figure 1).

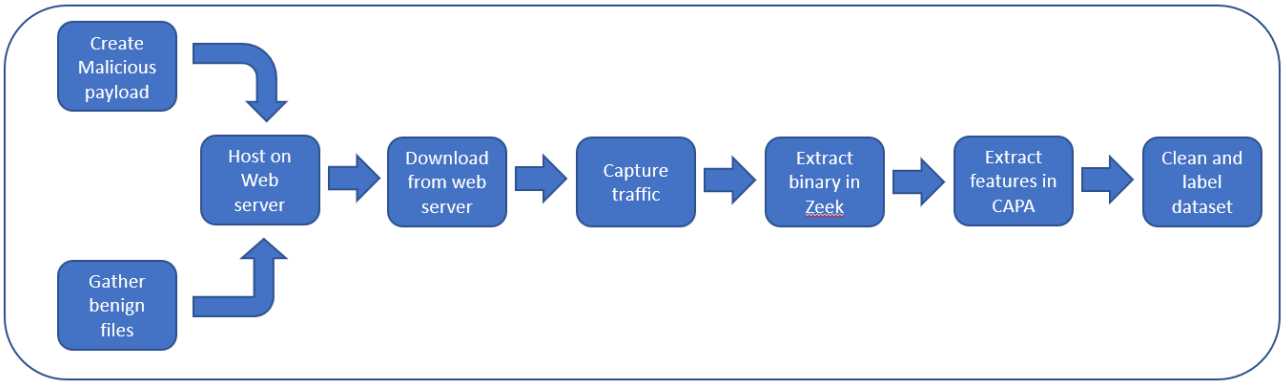


Figure 1 Dataset creation

The model selection will consist of training and validating 4 classifiers (C4.5, SMV, Naïve Bayes, and Random Forest), a model will be selected according to the resulting confusion matrix and ROC.

The selected model will be implemented on an environment like the one used in creating the dataset. Files will be hosted on a web server and downloaded by a windows machine. The traffic will be captured by a monitor server and the binary will be extracted from the traffic, processed, and used by the classifier who will write its results to a log (Figure 2).

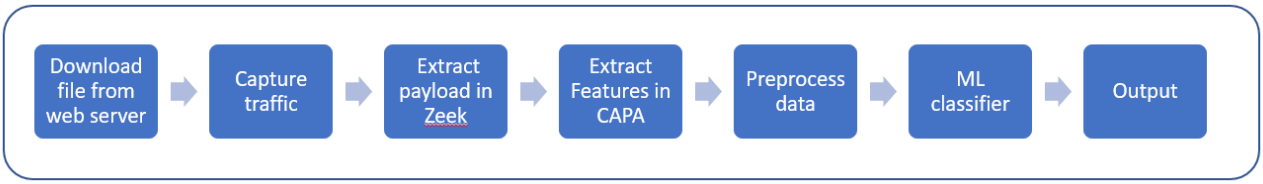


Figure 2: ML implementation

The educational component will consist of 4 labs, each will have a practical and theorical component and will cover the process end to end.

The final report will describe all the work done and include all the artifacts created during the project, i.e., all configuration files and scripts used.

4.1 Approach, tasks and phases:

The project work will be divided in two, the technical implementation that will be performed in 4 phases (Table 2); The work that will be done in the first semester includes deploying the testbed, creating the dataset, selecting the model, and implementing the model. The theoretical component, consisting of the educational material and labs, and the final report will be done in the second semester.

4.1.1 Technical Implementation

The virtual testbed (Figure 3) will be created to perform the experiments, all machines will be run as virtual machines using VMware. It will consist of one web server who will host the files, a client machine that will download the fi and a monitor server that will capture the traffic and run the classifier.

The virtual testbed (Figure 3) will be created to perform the experiments, and all machines will be run as virtual machines on a central ESXi hypervisor host (S1). It will consist of one server (Web Server) that will host our benign and malicious samples, a client machine that downloads these samples (Client) through a script, and a monitoring server that runs Zeek as the NSM tool (network security monitoring) in order to capture the traffic and extract the samples from it (Monitor). To satisfy the technical requirements of the Monitor, S1 will be additionally configured with a vSwitch (virtual switch) that has port mirroring/SPAN enabled to capture the traffic being sent from the Web Server to the Client.

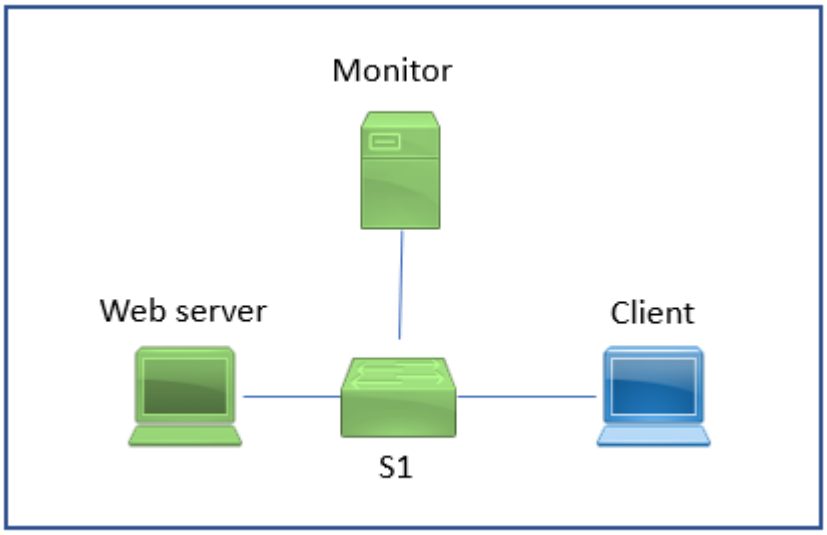


Figure 3 Testbed

To create the dataset used for training the ML model (Figure 1), we will generate the malicious samples using the Metasploit framework. We will use a script to automate the process. Script #1 (Table 3) will generate the Metasploit payload, document their MD5 hash, their name, and their label (malicious/benign) on a document.

The benign samples will be obtained by randomly selecting PE files from a fresh Windows installation, if additional benign files are needed, they will be obtained by downloading them from an open repository such download.com and checking the files in virustotal.com to verify they are not malware. This process will be done by script #2, it will also document the file name, md5 hash, and label.

All payloads will be hosted on the web server, script #3, script #4, will download all the payloads from the web server.

On the monitoring server, Zeek will be configured to extract the binaries. Script #5 will run the binaries through CAPA, process the output so it is in a format usable by the ML scripts, and label the data. The resulting output will be the dataset.

A binary classification model (benign/malicious) will be created using 4 algorithms, C4.5 decision tree, Random Forest, Support Vector Machine and Naïve Bayes. The classifiers will be evaluated using confusion matrix, the receiver operating characteristics curve (ROC) and the classification reports, these are standards and are used universally when evaluating machine learning models.

The scikit-learn library will be used to implement all the machine learning functionality. The features to be used in the models will be selected using the scikit-learn feature selection classes. At least two different methods will be used to illustrate how they work and show the different models created. While in practice these activities might be split into smaller python scripts, they will be referred as the script #6 (pre-process) and script #7 (model training and validation).

One classifier will be selected and implemented through a script #8 on the monitor server. The objective of this model is to be used as an example and to help illustrate educational objectives. It is not expected that the model should be able to run in a production environment, or to offer improvements in detection over existing models.

The implemented classifier script will run on the monitoring server. The script will check for new extracted files at regular intervals of time. When a file is found, it will be run through CAPA and the output will be processed and run through the classifier, the results will be recorded on a log. Experiments will then be run to evaluate the model efficacy

4.1.2 Curriculum implementation

The curriculum will be created in two phases (Table 2), the theorical component and a practical lab to reinforce the material. They will be designed to introduce detection through machine learning. This curriculum will cover 4 topics, gathering data, selecting features and training a ML model, and evaluation and implementing a ML. They are not meant to satisfy requirements for any accreditation or certification, and only serve as a practical introduction to the topic.

A final report will also be created, describing the project and analysis of our results and will include all the data and scripts used through the project.

TABLE 1: APPROACH UTILIZED FOR ACHIEVING OBJECTIVES

Objective	Approach of achieving the objective
1: Create dataset	msfvenom automation scripts, randomized benign data samples, download automation scripts
2: Model selection	ML training, validation, and evaluation
3: Model Implementation	Implemented by using python and experiments
4: Create educational material	Labs, PowerPoint presentations
5: Create final report	Report writing

TABLE 2: MAPPING OF PHASES AND TASKS TO ACHIEVE OBJECTIVES

Objectives	Phases	Tasks
Create Dataset	1.1: Deploy Testbed	• Deploy Kali workstation • Deploy Windows • Deploy Ubuntu monitoring server, install Zeek • Configure Zeek to extract files • Configure port mirroring through managed vSwitch on the ESXi hypervisor • Validate connectivity and configuration
Create dataset	1.2: Create Dataset	• Develop automation for creating dataset (scripts 1, 2, 3, 4) • Test and debug dataset automation (scripts 1, 2, 3, 4) • Run scripts 1, 2, 3 and 4 • Create script 5. • Test and debug script 5 • Generate dataset
Model selection	1.3: Model Selection	• Data pre-processing (script 6.a) • Data pre-processing (script 6.b) • Train and validate ML using Random Forest (script 7a) • Fine tune Random Forest model • Train and validate ML using C4.5 (script 7b) • Fine tune C4.5 model • Train and validate ML using SVM (script 7c) • Fine tune Support Vector Machine model • Train and validate ML using Naive Bayes (script 7d) • Fine tune Naive Bayes model
Model Implementation	1.4 Model Implementation	• Select classifier to implement • Implement classifier on monitoring server (script 8) • Test and debug classifier implementation (script 8) • Design experiments • Perform experiments
Create educational material	2.1: PowerPoint Presentation	• Create presentation for data gathering • Create presentation for feature selection

		• Create presentation for training, validation, and model evaluation • Create presentation for model implementation
Create educational material	2.2: Labs	• Create tutorial for data gathering • Create tutorial for feature selection • Create tutorial for training, validation, and model evaluation • Create tutorial for model implementation
Create educational material	2.3: Revise	• Revise tutorial for gathering data • Revise tutorial for feature selection • Revise tutorial for training, validation, and model evaluation • Revise tutorial for model implementation • Review and provide feedback of labs and presentations
Create final report	3.1: Report Draft	• Write abstract and introduction • Write Literature Review and Background • Write Methodology and Experiments • Write Discussion and Conclusion
Create final report	3.2: Final Report	• Revise abstract and introduction • Revise Literature Review and Background • Revise Methodology and Experiments • Revise Discussion and Conclusion

TABLE 3: LIST OF SCRIPTS

Identifier	Description
1	Creates malicious payloads, document name, MD5 hash, and label on a file.
2	Selects benign payloads, document name, MD5 hash, and label on a file.
3	Hosts file on web server, creates pages or file structure as necessary.
4	Download files from web server.
5	Extract features from binaries using CAPA.
6	Pre-process dataset using scikit-learn, there will be multiple variants of the script using different methods.
7	Model training and validation. There will be multiple variants, one for each classifier.

8	ML implementation, checks for binaries, extract features using CAPA, run it through the classifier and records results to a log.

4.2 Research methodology

Our research methodology will be applied to two separate experiments, in which we will use a different data source in order to evaluate our ML model. However, pending discussion with faculty administration, we may perform additional experiments as required with different parameters.

The first experiment will revolve around using automatically-generated samples using a script developed by us that leverages Metasploit reverse shell payloads encoded into an executable by way of msfvenom. After generating the samples - we propose a minimum of 50 samples-, it will be uploaded to the Web Server; the generation of the malicious samples and the Web Server are ideally the same machine. The Web Server machine will also house a minimum of 50 benign samples, gathered from random sources. Next, the download automation script will iteratively retrieve each sample to the Client, the traffic of which will be captured by Zeek on the Monitor host. The captured samples will be pre-processed and handled by other scripts for Phase 1.3 and 1.4 to implement ML model. The intention of this experiment is to see if the ML model is sufficient in binary classification for simple malicious executables. We hypothesize that the classifiers will be able to successfully class the malicious executables from the benign executables. Our limitations concern the breadth of the benign samples we are able to retrieve. It could be possible that the benign samples we retrieve may invertedly have some capabilities or actions that cause it to look more like the malicious samples, which may cause skewing of the data when it is presented to the ML model, leading to potential misclassification. Unfortunately, we do not have many means of mitigating these limitations due to the wide variety of capabilities used by benign executables. The bounds of this experiment are of course, limiting the malicious samples to those reverse shell executables generated by msfvenom using our automation script.

Our second experiment methodology is largely the same as the first, with the main difference being the acquisition of the malicious samples. In this experiment, we will be using in-the-wild malware samples provided by DASMALWERK (https://dasmalwerk.eu/). Our intention with this experiment is to benchmark our ML implementation in a real-world environment to ensure it can function competently with actual samples, as this will help students experiment with our environment beyond our practical introduction curriculum. The limitation of this experiment is the limited number of samples provided by DASMALWERK. The samples provided by DASMALWERK may not be indicative of the current threat landscape, and may also not provide an accurate representation of which malware is most present on the Internet at that current moment in time. We also do not have a means of mitigating this limitation at this time, since we do not have an accurate assessment on the ratio of different malware types that exist out in-the-wild. The bounds of this experiment will be constrained to the samples provided by DASMALWERK. We will not be including malware from other sources, including those generated by us.

4.3 Management Plan

The project will be implemented in two semesters, the technical implementation will be on the first semester, and the final report and educational material will be created on the second semester. Planning activities assume 12 weeks of work per semester, and 10 hours of work per week (Tables 5 and 6, and Figures 4 through 8).

All project material will be hosted online, details will depend on the REA705 and REA820 course requirements. On the first week of the semester, a weekly meeting will be scheduled, where project progress, risk and issues will be discussed. A short minute will be produced and hosted on the project repository. A periodic meeting with the faculty advisor will be scheduled according to the course requirements.

All decisions that do not require faculty approval, will be considered final if agreed by both participants, if necessary, these will be documented and hosted on the project repository.

Currently the project does not require the purchase of material and therefore no budget is expected or managed.

TABLE 4: ROLE AND INVOLVEMENT DURATION OF RESEARCH TEAM

Team Members	**Role**
Gaston Carvallo	Development Lead
Loyd Rafols	Infrastructure Lead

4.4 Project Deliverables:

We are expecting the following deliverables to be submitted at the completion of this project:

- A Linux-based (most likely Ubuntu) OVA file containing our learning environment for malicious executable detection using machine learning. This will include the tools (such as CAPA) and scripts (used for scikit-learn and other workflow automation processes) needed for a student to run through the process that we did when working on the projects.
- A dataset that will have a mix of benign and malicious applications along with their capabilities and other features used to classify each sample in the ML model.
- The scripts created to implement the machine learning infrastructure (such as cleaning and feature selection), sample data generation, and handling of downloads from the attacker to the victim.
- Presentations and lectures that will serve as the theoretical introduction for how machine learning can be used to detect malicious items in infosec.

- Labs that will serve as the practical introduction and build on the theory disseminated from the presentations and lectures.
- The final companion report will serve as the summation of all work done for the project, including administrative work done.

5. REFERENCES

Alhawi, Omar MK, et al. "Leveraging machine learning techniques for windows ransomware network traffic detection." *Cyber Threat Intelligence*, pp. 93-106. Springer, Cham, doi.org/10.1007/978-3-319-73951-9_5

Almashhadani, Ahmad, et al. "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware", *IEEE Access, vol. 7*, pp. 47053-47067, 2019, doi: 10.1109/ACCESS.2019.2907485

Arabo, Abdullahi, et al. "Detecting Ransomware Using Process Behavior Analysis." *Procedia Computer Science 168* (2020): pp. 289-296.

Aslan, Omar and Refik Samet. "A Comprehensive Review on Malware Detection Approaches," *IEEE Access*, vol. 8, 2020, pp. 6249-6271, doi: 10.1109/ACCESS.2019.2963724.

Bae, Seong Il, et al. "Ransomware detection using machine learning algorithms." *Concurrency and Computation: Practice and Experience* 32, no. 18 (2020): pp.1-11, doi: 10.1002/cpe.5422

Berrueta, Eduardo, et al. "A survey on detection techniques for cryptographic ransomware*." IEEE Access 7* (2019): pp. 144925-144944, doi: 10.1109/ACCESS.2019.2945839.

Brunau, Chris. “Ransomware News: WannaCry Attack Costs NHS Over $100 Million”. Datto, last modified October 18, 2018. www.datto.com/uk/blog/ransomware-news-wannacry-attack-costs-nhs-over-100-million

Chumachenko, Kateryna. *Machine learning methods for malware detection and classification*. Bachelor’s Thesis, 2017, South-Eastern Finland University of Applied Sciences (Xamk).

Cook, Sam. "Malware Statistics In 2021: Frequency, Impact, Cost & More". Comparitech, last modified 2021. www.comparitech.com/antivirus/malware-statistics-facts.

Hirano, Manabu and R. Kobayashi, "Machine Learning Based Ransomware Detection Using Storage Access Patterns Obtained From Live-forensic Hypervisor," *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Granada, Spain, 2019, pp. 1-6, doi: 10.1109/IOTSMS48152.2019.8939214.

IBM. "Cost Of A Data Breach Report 2020." IBM Security, IBM Corporation, last modified July 2020, www.ibm.com/security/digital-assets/cost-data-breach-report/

Jethva, Brijesh. *A new ransomware detection scheme based on tracking file signature and file entropy*. Master's Thesis, 2019, University of Victoria, Department of Electrical and Computer Engineering.

Manzano, Carlos, et al. "An Empirical Comparison of Supervised Algorithms for Ransomware Identification on Network Traffic." *39th International Conference of the Chilean Computer Science Society (SCCC)*, pp. 1-7. IEEE, 2020.

Modi, Jaimin. *Detecting Ransomware in Encrypted Network Traffic Using Machine Learning*, Master's thesis, 2019, University of Victoria.

Noorbehbahani, Fakhroddin, et al. "Ransomware Detection with Semi-Supervised Learning." In *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 024-029. IEEE, 2020.

Patsakis, Constantinos, et al. "Encrypted and covert DNS queries for botnets: Challenges and countermeasures." *Computers & Security 88.* 2020 doi.org/10.1016/j.cose.2019. pp.101614-101626.

Premrn, Jakob, 2020. "*Analysis of command and control connections using machine learning algorithms*." Master's thesis, University of Ljubljana, Faculty of Electrical Engineering.

Salehi, Saeid, et al. "A Novel Approach for Detecting DGA-based Ransomwares," *2018 15th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC)*, Tehran, Iran, 2018, pp. 1-7, doi: 10.1109/ISCISC.2018.8546941.

Sethi, Kamalakanta, et al. "A novel machine learning based malware detection and classification framework." *In 2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2019, pp. 1-4. IEEE, doi:10.1109/CyberSecPODS.2019.8885196.

Shhadat, Ihab, et al. "The Use of Machine Learning Techniques to Advance the Detection and Classification of Unknown Malware.*" Procedia Computer Science 170*, 2020, pp 917-922.

Symantec. “Internet Security Threat Report Volume 24, February 2019”. ISTR, Symantec Corporation, last modified 2019. docs.broadcom.com/doc/istr-24-2019-en

Zhang, Yihang. "Automatic Algorithmically Generated Domain Detection with Deep Learning Methods," *2020 IEEE 3rd International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, Shenyang, China, 2020, pp. 463-469, doi: 10.1109/AUTEEE50969.2020.9315559.

Zhu, H., et al. “A Network Behavior Analysis Method to Detect Reverse Remote Access Trojan.” *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, 2018, pp. 1007-1010, doi: 10.1109/ICSESS.2018.8663903.

TABLE 5: PROJECT WORK PLAN REA705

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Phase 1.1: Deploy Testbed		1											
Deploy Kali workstation	Loyd Rafols	1											
Deploy Windows	Loyd Rafols	1											
Deploy Ubuntu monitoring server	Loyd Rafols	1											
Configure Zeek to extract files	Loyd Rafols	1											
Configure port mirroring	Loyd Rafols	1											
Validate connectivity and configuration	Loyd Rafols	1											
Phase 1.2: Create Dataset		4											
develop automation for creating dataset (scripts 1, 2, 3, 4)	Loyd Rafols		1										
Test and debug dataset automation (scripts 1, 2, 3, 4)	Loyd Rafols			1									

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Run scripts 1, 2, 3 and 4	Loyd Rafols			1									
Create script 5.	Gaston Carvallo	1											
Test and debug script 5	Gaston Carvallo		1										
Generate dataset	Gaston Carvallo				1								
Phase 1.3: Model Selection						5							
Data pre-processing (script 6.a)	Loyd Rafols					1							
Data pre-processing (script 6.b)	Gaston Carvallo					1							
Train and validate ML using Random Forest (script 7a)	Loyd Rafols						1						
Fine tune Random Forest model	Loyd Rafols							1					
Train and validate ML using C4.5 (script 7b)	Gaston Carvallo						1						
Fine tune C4.5 model	Gaston Carvallo							1					

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Train and validate ML using SVM (script 7c)	Loyd Rafols								1				
Fine tune Support Vector Machine model	Loyd Rafols									1			
Train and validate ML using Naive Bayes (script 7d)	Gaston Carvallo								1				
Fine tune Naive Bayes model	Gaston Carvallo									1			
Phase 1.4: Model Implementation											3		
Prepare and deploy environment	Loyd Rafols										1		
Implement classifier on monitoring server (Script 8)	Gaston Carvallo										1		
Test and debug classifier implementation (Script 8)	Gaston Carvallo											1	
Design experiments	Loyd Rafols											1	
Perform experiments	Loyd Rafols												1

TABLE 6: PROJECT WORK PLAN REA820

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Phase 2.1: PowerPoint Presentations				2									
Create data gathering presentation	Loyd Rafols			1									
Create feature selection presentation	Loyd Rafols				1								
Create training, validation, and model evaluation presentation	Gaston Carvallo			1									
Create model implementation presentation	Gaston Carvallo				1								
Phase 2.2: Labs		2											
Create tutorial for data gathering	Loyd Rafols	1											
Create tutorial for feature selection	Loyd Rafols		1										
Create tutorial for training, validation, and model evaluation	Gaston Carvallo	1											

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Create tutorial for model implementation	Gaston Carvallo		1										
Phase 2.3: Presentation and tutorial revision						3							
Revise tutorial for data gathering	Loyd Rafols					1							
Revise tutorial for feature selection	Loyd Rafols						1						
Revise tutorial for training, validation, and model evaluation	Gaston Carvallo					1							
Revise tutorial for model implementation	Gaston Carvallo						1						
Review and provide feedback on other group member's labs and presentations	Loyd Rafols							1					
Review and provide feedback on other group member's labs and presentations	Gaston Carvallo							1					
Phase 3.1: Report Draft									2				

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Write abstract and introduction	Gaston Carvallo								1				
Write literature review and background	Loyd Rafols								1				
Write research methodology and experiments	Gaston Carvallo									1			
Write discussion and conclusion	Loyd Rafols									1			
Phase 3.2: Final Report											2		
Revise abstract and introduction	Gaston Carvallo										1		
Revise literature review and background	Loyd Rafols										1		
Revise methodology and experiments	Gaston Carvallo											1	
Revise discussion and conclusion	Loyd Rafols											1	

	Task Mode	Task Name	Duration	Start	Finish	F
1		**REA Project**	**315 days**	**Mon 21-05-24**	**Fri 22-08-05**	
2		**Objective 1: Implement machine learning model**	**60 days**	**Mon 21-05-24**	**Fri 21-08-13**	
3		Project Kickoff	0 days	Mon 21-05-24	Mon 21-05-24	
4		**Phase 1.1: Deploy Testbed**	**4.5 days**	**Mon 21-05-24**	**Fri 21-05-28**	
12		**Phase 1.2: Create dataset**	**20 days**	**Mon 21-05-24**	**Fri 21-06-18**	
20		**Phase 1.3: Model Selection**	**25 days**	**Mon 21-06-21**	**Fri 21-07-23**	1
32		**Phase 1.4: Machine Implementation**	**15 days**	**Mon 21-07-26**	**Fri 21-08-13**	3
39		Objective 1 completion	0 days	Fri 21-08-13	Fri 21-08-13	3
40		**Objective 2: Create educational material**	**35 days**	**Mon 22-05-23**	**Fri 22-07-08**	3
41		**Phase 2.1: PowerPoint Presentations**	**10 days**	**Mon 22-06-06**	**Fri 22-06-17**	4
47		**Phase 2.2: Labs**	**10 days**	**Mon 22-05-23**	**Fri 22-06-03**	
53		**Phase 2.3: Lab revision**	**15 days**	**Mon 22-06-20**	**Fri 22-07-08**	4
61		Objective 2 completion	0 days	Fri 22-07-08	Fri 22-07-08	6
62		**Objective 3: Create final report**	**20 days**	**Mon 22-07-11**	**Fri 22-08-05**	3
63		**Phase 3.1: Report Drafting**	**10 days**	**Mon 22-07-11**	**Fri 22-07-22**	
69		**Phase 3.2: Final Report**	**10 days**	**Mon 22-07-25**	**Fri 22-08-05**	6
75		Objective 3 completion	0 days	Fri 22-08-05	Fri 22-08-05	6

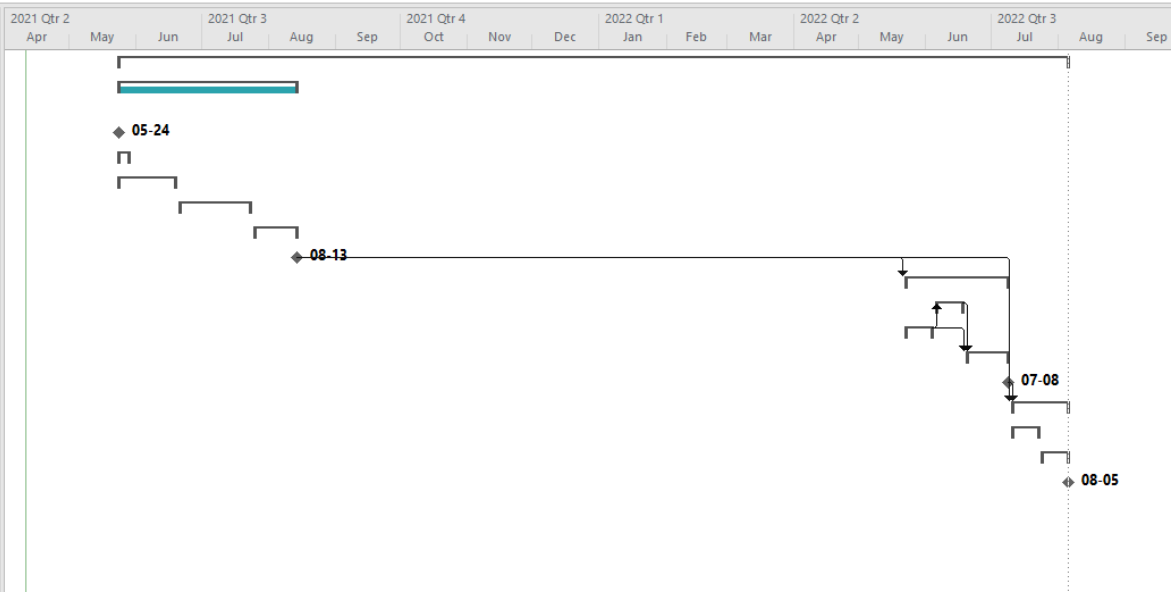
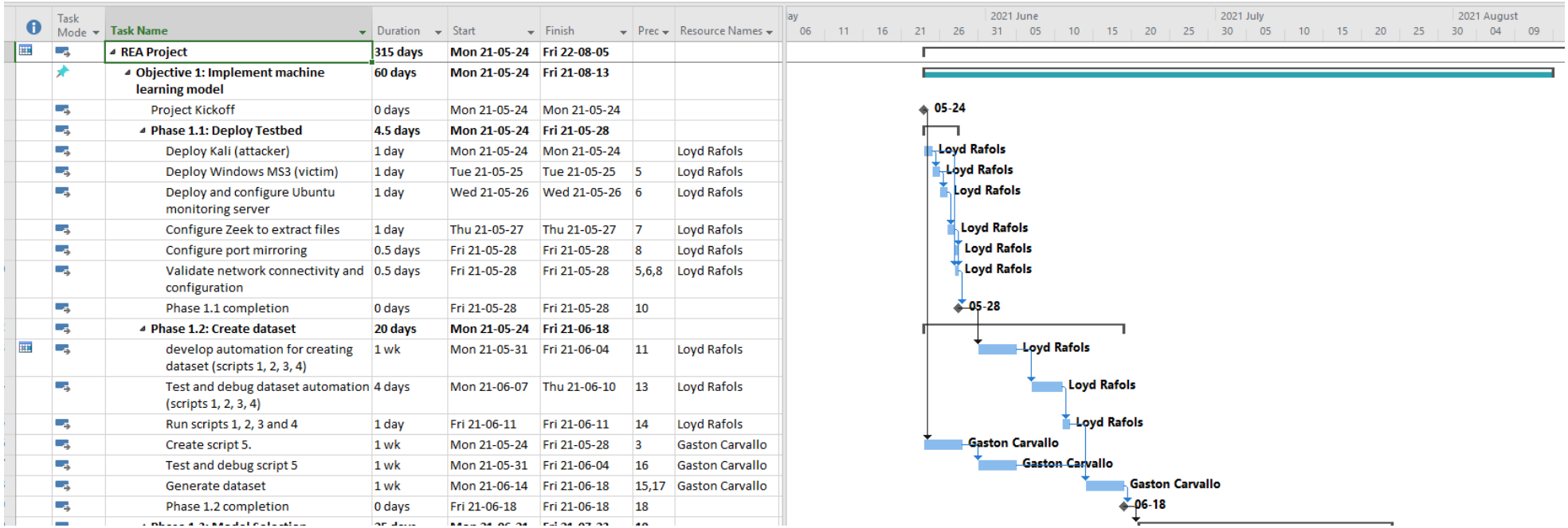
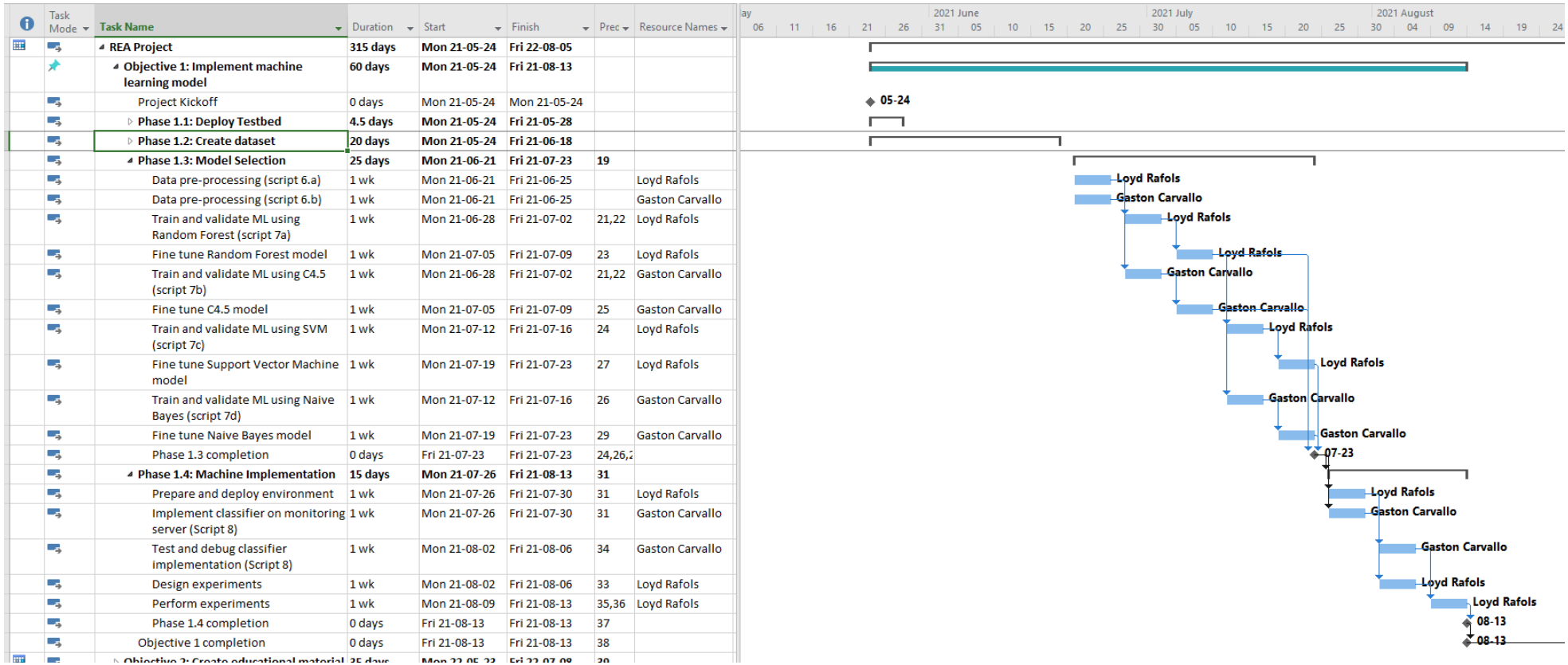


Figure 4 High Level Gantt



Task Mode	Task Name	Duration	Start	Finish	Prec	Resource Names
	REA Project	**315 days**	**Mon 21-05-24**	**Fri 22-08-05**		
	Objective 1: Implement machine learning model	**60 days**	**Mon 21-05-24**	**Fri 21-08-13**		
	Project Kickoff	0 days	Mon 21-05-24	Mon 21-05-24		
	Phase 1.1: Deploy Testbed	**4.5 days**	**Mon 21-05-24**	**Fri 21-05-28**		
	Deploy Kali (attacker)	1 day	Mon 21-05-24	Mon 21-05-24		Loyd Rafols
	Deploy Windows MS3 (victim)	1 day	Tue 21-05-25	Tue 21-05-25	5	Loyd Rafols
	Deploy and configure Ubuntu monitoring server	1 day	Wed 21-05-26	Wed 21-05-26	6	Loyd Rafols
	Configure Zeek to extract files	1 day	Thu 21-05-27	Thu 21-05-27	7	Loyd Rafols
	Configure port mirroring	0.5 days	Fri 21-05-28	Fri 21-05-28	8	Loyd Rafols
	Validate network connectivity and configuration	0.5 days	Fri 21-05-28	Fri 21-05-28	5,6,8	Loyd Rafols
	Phase 1.1 completion	0 days	Fri 21-05-28	Fri 21-05-28	10	
	Phase 1.2: Create dataset	**20 days**	**Mon 21-05-24**	**Fri 21-06-18**		
	develop automation for creating dataset (scripts 1, 2, 3, 4)	1 wk	Mon 21-05-31	Fri 21-06-04	11	Loyd Rafols
	Test and debug dataset automation (scripts 1, 2, 3, 4)	4 days	Mon 21-06-07	Thu 21-06-10	13	Loyd Rafols
	Run scripts 1, 2, 3 and 4	1 day	Fri 21-06-11	Fri 21-06-11	14	Loyd Rafols
	Create script 5.	1 wk	Mon 21-05-24	Fri 21-05-28	3	Gaston Carvallo
	Test and debug script 5	1 wk	Mon 21-05-31	Fri 21-06-04	16	Gaston Carvallo
	Generate dataset	1 wk	Mon 21-06-14	Fri 21-06-18	15,17	Gaston Carvallo
	Phase 1.2 completion	0 days	Fri 21-06-18	Fri 21-06-18	18	

Figure 5 Detailed Gantt Semester 1 (part 1)



Task Mode	Task Name	Duration	Start	Finish	Prec	Resource Names
	REA Project	**315 days**	**Mon 21-05-24**	**Fri 22-08-05**		
	Objective 1: Implement machine learning model	**60 days**	**Mon 21-05-24**	**Fri 21-08-13**		
	Project Kickoff	0 days	Mon 21-05-24	Mon 21-05-24		
	Phase 1.1: Deploy Testbed	**4.5 days**	**Mon 21-05-24**	**Fri 21-05-28**		
	Phase 1.2: Create dataset	**20 days**	**Mon 21-05-24**	**Fri 21-06-18**		
	Phase 1.3: Model Selection	**25 days**	**Mon 21-06-21**	**Fri 21-07-23**	**19**	
	Data pre-processing (script 6.a)	1 wk	Mon 21-06-21	Fri 21-06-25		Loyd Rafols
	Data pre-processing (script 6.b)	1 wk	Mon 21-06-21	Fri 21-06-25		Gaston Carvallo
	Train and validate ML using Random Forest (script 7a)	1 wk	Mon 21-06-28	Fri 21-07-02	21,22	Loyd Rafols
	Fine tune Random Forest model	1 wk	Mon 21-07-05	Fri 21-07-09	23	Loyd Rafols
	Train and validate ML using C4.5 (script 7b)	1 wk	Mon 21-06-28	Fri 21-07-02	21,22	Gaston Carvallo
	Fine tune C4.5 model	1 wk	Mon 21-07-05	Fri 21-07-09	25	Gaston Carvallo
	Train and validate ML using SVM (script 7c)	1 wk	Mon 21-07-12	Fri 21-07-16	24	Loyd Rafols
	Fine tune Support Vector Machine model	1 wk	Mon 21-07-19	Fri 21-07-23	27	Loyd Rafols
	Train and validate ML using Naive Bayes (script 7d)	1 wk	Mon 21-07-12	Fri 21-07-16	26	Gaston Carvallo
	Fine tune Naive Bayes model	1 wk	Mon 21-07-19	Fri 21-07-23	29	Gaston Carvallo
	Phase 1.3 completion	0 days	Fri 21-07-23	Fri 21-07-23	24,26,2	
	Phase 1.4: Machine Implementation	**15 days**	**Mon 21-07-26**	**Fri 21-08-13**	**31**	
	Prepare and deploy environment	1 wk	Mon 21-07-26	Fri 21-07-30	31	Loyd Rafols
	Implement classifier on monitoring server (Script 8)	1 wk	Mon 21-07-26	Fri 21-07-30	31	Gaston Carvallo
	Test and debug classifier implementation (Script 8)	1 wk	Mon 21-08-02	Fri 21-08-06	34	Gaston Carvallo
	Design experiments	1 wk	Mon 21-08-02	Fri 21-08-06	33	Loyd Rafols
	Perform experiments	1 wk	Mon 21-08-09	Fri 21-08-13	35,36	Loyd Rafols
	Phase 1.4 completion	0 days	Fri 21-08-13	Fri 21-08-13	37	
	Objective 1 completion	0 days	Fri 21-08-13	Fri 21-08-13	38	

Figure 6 Detailed Gantt Semester 1 (part 2)

Task Mode	Task Name	Duration	Start	Finish	Prec	Resource Names
	REA Project	**315 days**	**Mon 21-05-24**	**Fri 22-08-05**		
	Objective 1: Implement machine learning model	**60 days**	**Mon 21-05-24**	**Fri 21-08-13**		
	Objective 2: Create educational material	**35 days**	**Mon 22-05-23**	**Fri 22-07-08**	**39**	
	Phase 2.1: PowerPoint Presentations	**10 days**	**Mon 22-06-06**	**Fri 22-06-17**	**47**	
	Create data gathering presentation	5 days	Mon 22-06-06	Fri 22-06-10		Loyd Rafols
	Create feature selection presentation	5 days	Mon 22-06-13	Fri 22-06-17	42	Loyd Rafols
	Create training, validation, and model evaluation presentation	5 days	Mon 22-06-06	Fri 22-06-10		Gaston Carvallo
	Create model implementation pres	5 days	Mon 22-06-13	Fri 22-06-17	44	Gaston Carvallo
	Phase 2.1 completion	0 days	Fri 22-06-17	Fri 22-06-17	42,43,4	
	Phase 2.2: Labs	**10 days**	**Mon 22-05-23**	**Fri 22-06-03**		
	Create tutorial for data gathering	5 days	Mon 22-05-23	Fri 22-05-27		Loyd Rafols
	Create tutorial for feature selection	5 days	Mon 22-05-30	Fri 22-06-03	48	Loyd Rafols
	Create tutorial for training, validation and model evaluation	5 days	Mon 22-05-23	Fri 22-05-27		Gaston Carvallo
	Create tutorial for model implemer	5 days	Mon 22-05-30	Fri 22-06-03	50	Gaston Carvallo
	Phase 2.2 completion	0 days	Fri 22-06-03	Fri 22-06-03	48,49,5	
	Phase 2.3: Lab revision	**15 days**	**Mon 22-06-20**	**Fri 22-07-08**	**41,47**	
	Revise tutorial for data gathering	1 wk	Mon 22-06-20	Fri 22-06-24		Loyd Rafols
	Revise tutorial for feature selection	1 wk	Mon 22-06-27	Fri 22-07-01	54	Loyd Rafols
	Revise tutorial for training, validation, and model evaluation	1 wk	Mon 22-06-20	Fri 22-06-24		Gaston Carvallo
	Revise tutorial for model implemer	1 wk	Mon 22-06-27	Fri 22-07-01	56	Gaston Carvallo
	Review and provide feedback on othjer group member's labs and presentations	1 wk	Mon 22-07-04	Fri 22-07-08	54,55	Gaston Carvallo
	Review and provide feedback on othjer group member's labs and presentations	1 wk	Mon 22-07-04	Fri 22-07-08	56,57	Loyd Rafols
	Phase 2.3 completion	0 days	Fri 22-07-08	Fri 22-07-08	54,55,5	
	Objective 2 completion	0 days	Fri 22-07-08	Fri 22-07-08	60,52,4	

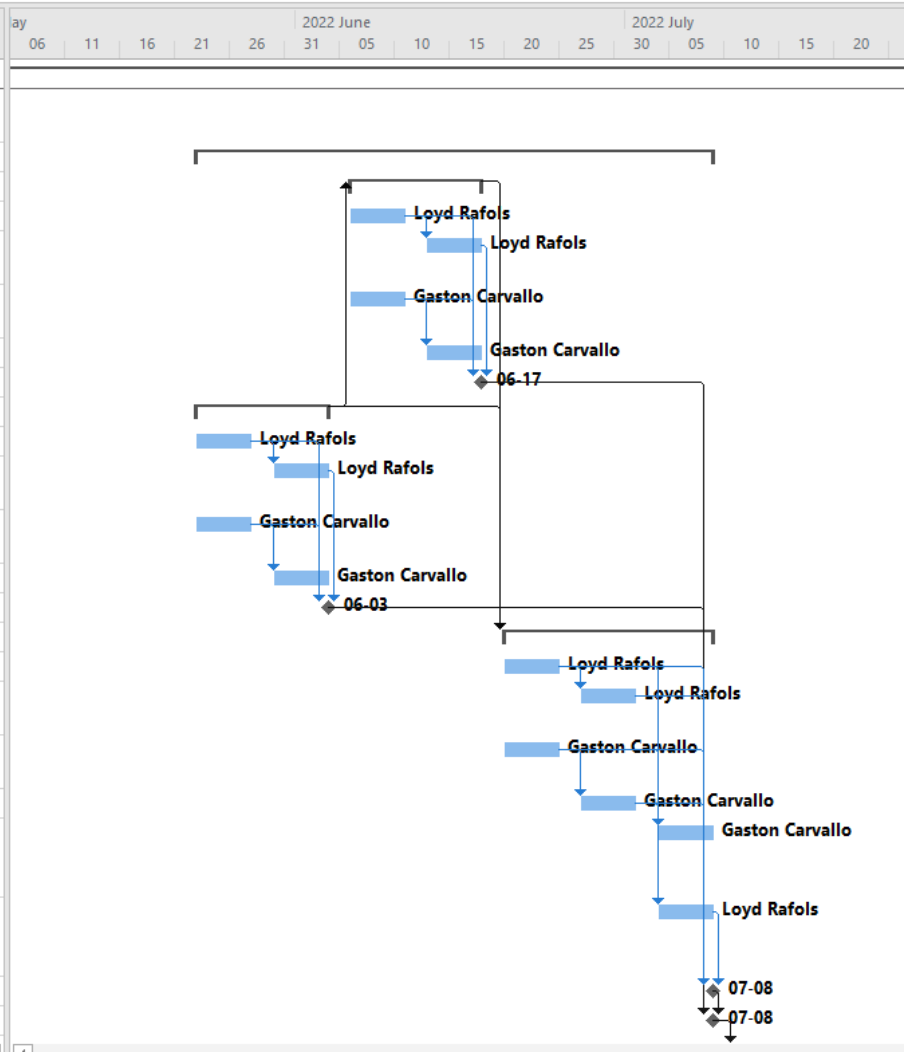
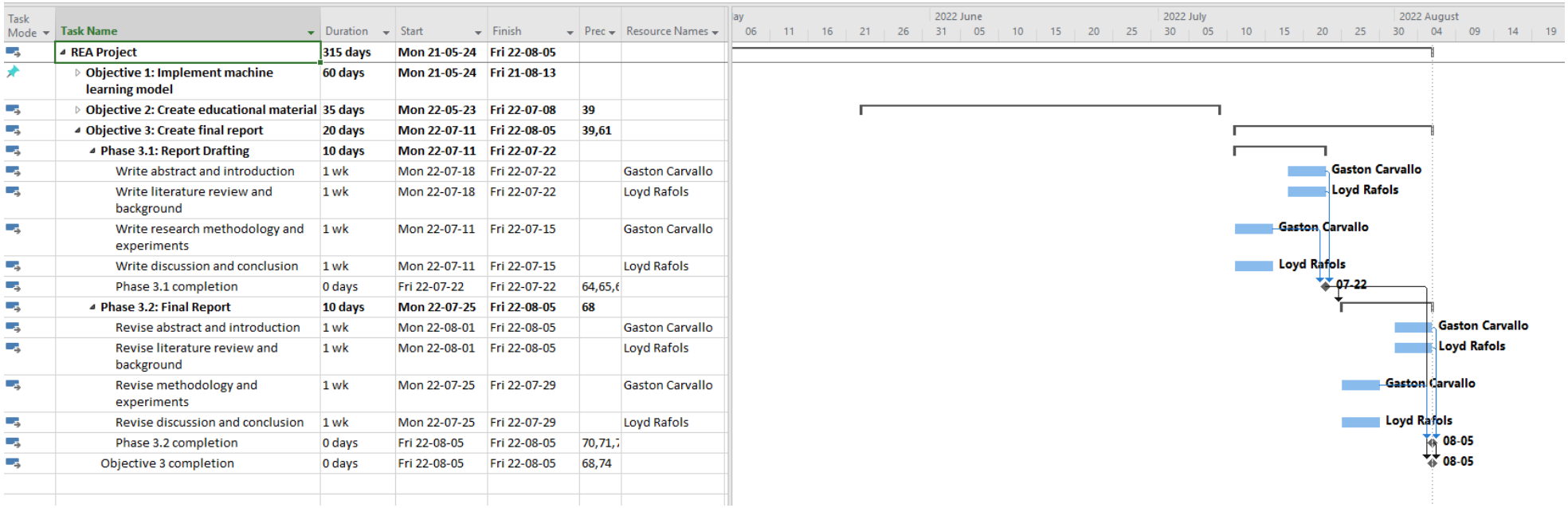


Figure 7 Detailed Gantt Semester 2 (part 1)



Task Mode	Task Name	Duration	Start	Finish	Prec	Resource Names
	REA Project	**315 days**	**Mon 21-05-24**	**Fri 22-08-05**		
	Objective 1: Implement machine learning model	**60 days**	**Mon 21-05-24**	**Fri 21-08-13**		
	Objective 2: Create educational material	**35 days**	**Mon 22-05-23**	**Fri 22-07-08**	**39**	
	Objective 3: Create final report	**20 days**	**Mon 22-07-11**	**Fri 22-08-05**	**39,61**	
	Phase 3.1: Report Drafting	**10 days**	**Mon 22-07-11**	**Fri 22-07-22**		
	Write abstract and introduction	1 wk	Mon 22-07-18	Fri 22-07-22		Gaston Carvallo
	Write literature review and background	1 wk	Mon 22-07-18	Fri 22-07-22		Loyd Rafols
	Write research methodology and experiments	1 wk	Mon 22-07-11	Fri 22-07-15		Gaston Carvallo
	Write discussion and conclusion	1 wk	Mon 22-07-11	Fri 22-07-15		Loyd Rafols
	Phase 3.1 completion	0 days	Fri 22-07-22	Fri 22-07-22	64,65,6	
	Phase 3.2: Final Report	**10 days**	**Mon 22-07-25**	**Fri 22-08-05**	**68**	
	Revise abstract and introduction	1 wk	Mon 22-08-01	Fri 22-08-05		Gaston Carvallo
	Revise literature review and background	1 wk	Mon 22-08-01	Fri 22-08-05		Loyd Rafols
	Revise methodology and experiments	1 wk	Mon 22-07-25	Fri 22-07-29		Gaston Carvallo
	Revise discussion and conclusion	1 wk	Mon 22-07-25	Fri 22-07-29		Loyd Rafols
	Phase 3.2 completion	0 days	Fri 22-08-05	Fri 22-08-05	70,71,7	
	Objective 3 completion	0 days	Fri 22-08-05	Fri 22-08-05	68,74	

Figure 8 Detailed Gantt Semester 2 (part 2)