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. 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.

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.

Current industry trends seem to indicate that machine learning will be more prevalent in detecting malware as it traverses or is downloaded into a network. However, the concept of machine learning is lost on many students, and many (ourselves included) feel overwhelmed when trying to get into understanding the fundamentals of machine learning, partially due to their lack of practical experience with these techniques, and partially because without a curriculum to simplify the introduction and reduce the apprehension to approach machine learning, it makes the topic be seen as an enigma that is difficult to enter.

What we want to achieve with this project is the creation of a practical environment and curriculum, designed to serve as an introduction to the fundamentals aspects of applying a machine learning model to a security problem. In particular, how machine learning can be used to detect malware. Our main objective is to provide a self-paced introduction to practically applying machine learning in the form of labs, lowering the barrier of entry, and granting students that capability of experimenting further with machine learning and figuring out different ways it could be applied in

information security. These labs would describe all the necessary steps to create and implement a machine learning model. To accomplish this, we will implement a model that uses static analysis to classify detected files into malware/benign, covering the whole process, including setting up the test environment, how to create the sample data, how to train and validate a model, and finally how the model can be implemented.

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 malicious executables to illustrate the process. We will create our own dataset by generating the malicious samples using the Metasploit Framework and in-the-wild malware samples, extract the features using CAPA[1], training 4 classifiers: Random Forest, C4.5, 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

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

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 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

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.

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 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.

[4] A method of encoding non-rankable items into a numeric order (such as colors)
[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.

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 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

[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
[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

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.

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

[12] Application Programming Interface - a means for software to allow interaction with itself through predefined functions or tasks.

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 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-

[13] A means of running one or several virtual computers on one or more physical computers
[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

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 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).

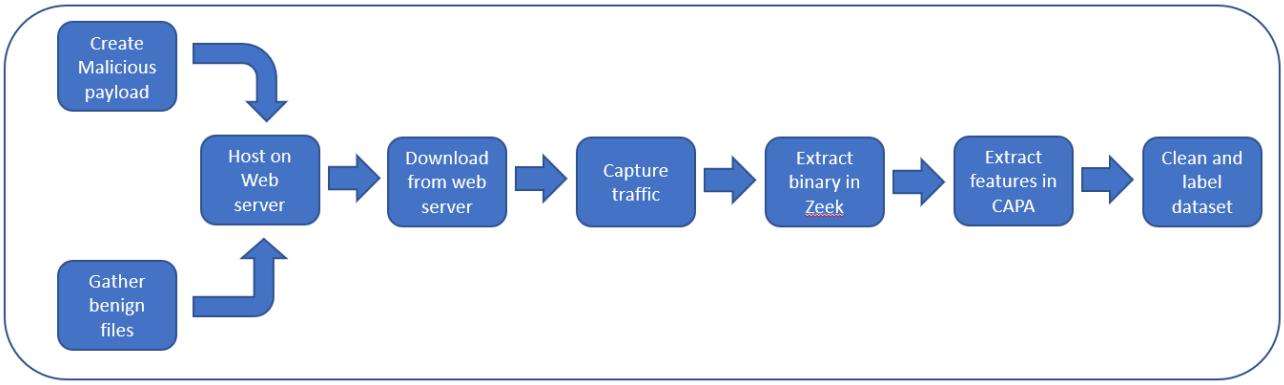

Figure 1: Dataset creation workflow

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).

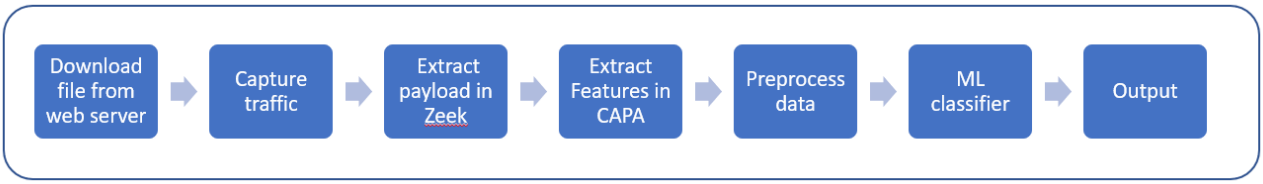

Figure 2: ML implementation

The educational component will consist of 6 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, 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.

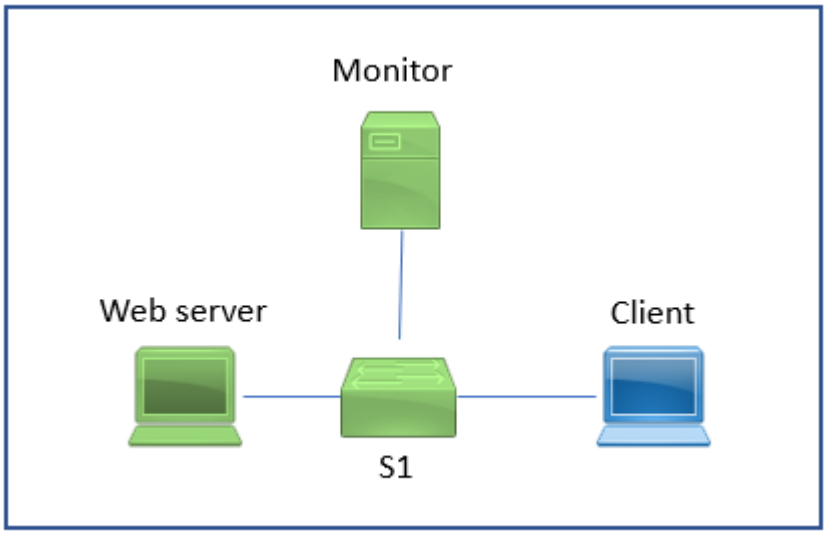

Figure 3: Project Testbed

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. the script “Malicious Payloads” (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 the script “Benign Payloads”, it will also document the file name, md5 hash, and label.

All payloads will be hosted on the web server using the script “Host Files” and the script “Download Files” will download all the payloads from the web server.

On the monitoring server, Zeek will be configured to extract the binaries. The script “Analyse File” 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 “Pre-process dataset” and the script “Train and validation”.

One classifier will be selected and implemented through a script called “ML implementation” 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. This will be done in 6 labs (Table 4), where each build upon the previous one, adding complexity. The first lab will be about building the

environment, this will include creating a testbed and installing the necessary software packages. The second lab will cover how to create the dataset. The third lab will cover how to train and validate the random forest classifier. The fourth lab will cover how to train and validate the C4.5 and Naïve Bayes classifier but will additionally include how to improve feature selection. The fifth lab will use the 4 classifiers (Random Forest, C4.5, Naïve Bayes and SVM) and will include how to compare the models and select the best one. The last lab will cover how to perform Hyperparameter tuning and improve the classifier selected on the Model Evaluation Lab (Lab 5).

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 Malicious Payloads, Benign Payloads, Host Files and Download Files) • Test and debug dataset automation ((scripts Malicious Payloads, Benign Payloads, Host Files and Download Files)) • Run scripts (scripts Malicious Payloads, Benign Payloads, Host Files and Download Files) • Create script “Analyse File”. • Test and debug script “Analyse File” • Generate dataset
Model selection	2.1: Model selection	• Data pre-processing • Train and validate ML using Random Forest • Train and validate ML using C4.5 • Train and validate ML using Naive Bayes • Train and validate ML using SVM • Select classifier to implement
Model Implementation	3.1 Model implementation	• Create script to implement classifier on monitoring server (ML implementation) • Test and debug the script “ML implementation”
Model Implementation	3.2 Experiments	• Design experiments • Perform experiments
Create educational material	4.1: Labs	• Create Setup Lab tutorial • Create Dataset Lab tutorial • Create Random Forest Lab tutorial • Create C4.5/Naïve Bayes Lab tutorial • Create Model Evaluation Lab tutorial • Create Hyperparameter Tuning Lab tutorial
Create educational material	4.2: PowerPoint Presentations	• Create presentation for Setup Lab tutorial • Create presentation for Dataset Lab tutorial • Create presentation for Random Forest Lab tutorial • Create presentation for C4.5/Naïve Bayes Lab tutorial • Create presentation for Model Evaluation Lab tutorial

		• Create presentation for Hyperparameter Tuning Lab tutorial
Create educational material	4.3: Revise	• Revise Setup Lab tutorial • Revise Dataset Lab tutorial • Revise Random Forest Lab tutorial • Revise C4.5/Naïve Bayes Lab tutorial Revise Model Evaluation Lab tutorial • Revise Hyperparameter Tuning Lab tutorial • Review and provide feedback of labs and presentations
Create final report	5.1: Report Draft	• Write abstract and introduction • Write Literature Review and Background • Write Methodology and Experiments • Write Discussion and Conclusion
Create final report	5.2: Final Report	• Revise abstract and introduction • Revise Literature Review and Background • Revise Methodology and Experiments • Revise Discussion and Conclusion

TABLE 3: LIST OF SCRIPTS

ID	Name	Description
1	Malicious Payloads	Creates malicious payloads, document name, MD5 hash, and label on a file.
2	Benign Payloads	Selects benign payloads, document name, MD5 hash, and label on a file.
3	Host Files	Hosts file on web server, creates pages or file structure, as necessary.
4	Download Files	Download files from web server.
5	Analyse File	Extract features from binaries using CAPA.

6	Pre-process dataset	Pre-process dataset using scikit-learn, there will be multiple variants of the script using different methods.
7	Train and validation	Model training and validation. There will be multiple variants, one for each classifier.
8	ML implementation	ML implementation, checks for binaries, extract features using CAPA, run it through the classifier and records results to a log.

TABLE 4: LIST OF LABS/TUTORIALS AND PRESENTATIONS

ID	Lab/Presentation	Description
1	Setup Lab/Presentation	Setting up the environment, include networking and installing required software packages.
2	Dataset Lab/Presentation	Creating the dataset, includes data collection and feature extraction
3	Random Forest Lab/Presentation	Classification using Random Forest, includes data pre-processing and training the model
4	C4.5/Naïve Bayes Lab/Presentation	Classification using C4.5 and Naïve Bayes, it includes how to improve feature selection
5	Model Evaluation Lab/Presentation	Compare the previous 4 classifiers and select the best one
6	Hyperparameter Tuning Lab/Presentation	Hyperparameter tunning of the classifier selected in lab 5

4.2 Research methodology

Our research methodology will be applied to two separate series of experiments, in which we will use different methods to evaluate our ML model implementation and demonstrate the limitations of it. However, pending discussion with faculty administration, we may perform additional experiments as required with different parameters or metrics that will be benchmarked.

The goal of the first series of experiments is to benchmark the ML model implementation's effectiveness of differentiating from malicious executables from benign executables. The purpose of

this experiment is to establish our ML model implementation's True Positive Rate and True Negative Rate (TPR/TNR), in comparison with contemporary malware detection solutions such as Malwarebytes or VirusTotal. The experiment process will follow the same workflow of model evaluation. We will provide samples generated through our malicious payload automation script, malware samples from DAS MALWERK, and benign samples found on various FOSS and download websites such as download.com. After the model provides its final binary classification on a particular sample, we will compare the results given by Malwarebytes and VirusTotal against our own implementation. In this experiment, our central metric will revolve around the binary classification generated by our implementation (that is, if an executable is malicious or not). We hypothesize that the results of this experiment will result in an adequate TPR/TNR rate (with estimations being around at least 80%), but not decisively better than Malwarebytes or VirusTotal.

Our second series of experiments will be designed toward measuring the resources necessary to run the ML implementation, i.e., these metrics being: how much memory it needs, the CPU usage of the implementation, and how long it takes to analyze a file from initial extraction to final classification. We hypothesize that our implementation will not have an adverse effect on the Monitor server, as the requirements to run the necessary components of the ML model is insignificant compared to the requirements needed by Zeek to function optimally for the scale of this learning environment (typically equal to or less than 100 Mbps). To run the experiments, random samples of benign and malicious files will be hosted on the web server and downloaded to the client, in a similar fashion to the first experiment, however sensors will be running on the monitor host, capturing how much resources the ML is using and how long it takes to analyze a file in correspondence to the metrics stated previously. Different scenarios will be run, including downloading several files at the same time, to see how this affects the performance of the implemented classifier.

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 6 and 7, 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 5: 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.

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

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. 24-29. 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.

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 6: 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 Malicious Payloads, Benign Payloads, Host Files and Download Files)	Loyd Rafols		1										

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Test and debug dataset automation ((scripts Malicious Payloads, Benign Payloads, Host Files and Download Files))	Loyd Rafols			1									
Run scripts (scripts Malicious Payloads, Benign Payloads, Host Files and Download Files)	Loyd Rafols			1									
Create script “Analyse File”.	Gaston Carvallo	1											
Test and debug script “Analyse File”	Gaston Carvallo		1										
Generate dataset	Gaston Carvallo				1								
Phase 2.1: Model Selection						4							
Data pre-processing	Gaston Carvallo					1							
Train and validate ML using Random Forest	Gaston Carvallo						1						
Train and validate ML using C4.5	Loyd Rafols						1						
Train and validate ML using Naive Bayes	Loyd Rafols							1					

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Train and validate ML using SVM	Loyd Rafols								1				
Select classifier	Gaston Carvallo									1			
Phase 3.1: Model Implementation											2		
Create script to implement classifier on monitoring server (ML implementation)	Gaston Carvallo										1		
Test and debug the script “ML implementation”	Gaston Carvallo											1	
Phase 4.1: Labs					9								
Create Setup Lab tutorial	Loyd Rafols				1								
Create Dataset Lab tutorial	Loyd Rafols					1							
Create Random Forest Lab tutorial	Gaston Carvallo							1					
Create C4.5/Naïve Bayes Lab tutorial	Gaston Carvallo								1				

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Revise Setup, Dataset, Random Forest, and C4.5/Naïve Bayes Lab tutorials	Loyd Rafols									1			
Create Model Evaluation Lab tutorial	Gaston Carvallo									1			
Revise Model Evaluation Lab tutorial	Loyd Rafols										1		
Create Hyperparameter Tuning Lab tutorial	Loyd Rafols											1	
Revise Hyperparameter Tuning Lab tutorial	Gaston Carvallo												1

TABLE 7: PROJECT WORK PLAN REA820

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Phase 3.2 Experiments		**2**											
Design experiments	Loyd Rafols	**1**											
Run experiment 1	Gaston Carvallo		**1**										
Run experiment 2	Loyd Rafols		**1**										
Phase 4.2: PowerPoint Presentations				3									
Create Setup Lab tutorial presentation	Loyd Rafols			1									
Create Dataset Lab tutorial presentation	Loyd Rafols				1								
Create Random Forest Lab tutorial presentation	Gaston Carvallo			1									
Create C4.5/Naïve Bayes Lab tutorial presentation	Gaston Carvallo				1								
Create Model Evaluation Lab tutorial presentation	Loyd Rafols					1							
Create Hyperparameter Tuning Lab tutorial presentation	Gaston Carvallo					1							

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Phase 4.3: Presentation and tutorial revision							3						
Revise Setup and Dataset Lab tutorials	Loyd Rafols						1						
Revise Model Evaluation Lab tutorial	Loyd Rafols							1					
Revise Random Forest and C4.5/Naïve Bayes tutorial	Gaston Carvallo						1						
Revise Hyperparameter Tuning Lab tutorial	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 5.1: Report Draft										2			
Write abstract and introduction	Gaston Carvallo									1			

PHASES & TASKS	DURATION INVOLVEMENT	1	2	3	4	5	6	7	8	9	10	11	12
Write literature review and background	Loyd Rafols									1			
Write research methodology and experiments	Gaston Carvallo										1		
Write discussion and conclusion	Loyd Rafols										1		
Phase 5.2: 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 Name	Duration	Start	Finish	Predece
REA Project	**320 days?**	**Mon 21-05-17**	**Fri 22-08-05**	
Project Kickoff	0 days	Mon 21-05-17	Mon 21-05-17	
Objective 1: Create dataset	**60 days**	**Mon 21-05-17**	**Fri 21-08-06**	
Phase 1.1: Deploy testbed	**4.5 days**	**Mon 21-05-17**	**Fri 21-05-21**	
Phase 1.2: Create dataset	**20 days**	**Mon 21-05-17**	**Fri 21-06-11**	
Objective 1 completion	0 days	Fri 21-06-11	Fri 21-06-11	11,19
Objective 2: Model selection	**21 days**	**Mon 21-06-14**	**Tue 21-07-13**	
Phase 2.1: Model Selection	**21 days**	**Mon 21-06-14**	**Tue 21-07-13**	**20**
Objective 2 completion	0 days	Tue 21-07-13	Tue 21-07-13	29FF
Objective 3: Model implementation	**255 days?**	**Mon 21-07-19**	**Fri 22-07-08**	
Phase 3.1: Model implementation	**10 days**	**Mon 21-07-19**	**Fri 21-07-30**	**30**
Phase 3.2 Experiments	**40 days?**	**Mon 22-05-16**	**Fri 22-07-08**	**35**
Objective 3 completion	0 days	Fri 22-05-27	Fri 22-05-27	35FF,40FF
Objective 4: Create educational material	**300 days**	**Mon 21-05-17**	**Fri 22-07-08**	**20**
Phase 4.1: Labs	**60 days**	**Mon 21-05-17**	**Fri 21-08-06**	
Phase 4.2: PowerPoint Presentations	**15 days**	**Mon 22-05-30**	**Fri 22-06-17**	**40**
Phase 4.3: Presentation and tutorial revision	**15 days**	**Mon 22-06-20**	**Fri 22-07-08**	**61**
Objective 4 completion	0 days	Fri 22-07-08	Fri 22-07-08	69,61,53
Objective 5: Create final report	**20 days**	**Mon 22-07-11**	**Fri 22-08-05**	**20,70**
Phase 5.1: Report Drafting	**10 days**	**Mon 22-07-11**	**Fri 22-07-22**	**70**
Phase 5.2: Final Report	**10 days**	**Mon 22-07-25**	**Fri 22-08-05**	**77**
Objective 5 completion	0 days	Fri 22-08-05	Fri 22-08-05	77,83

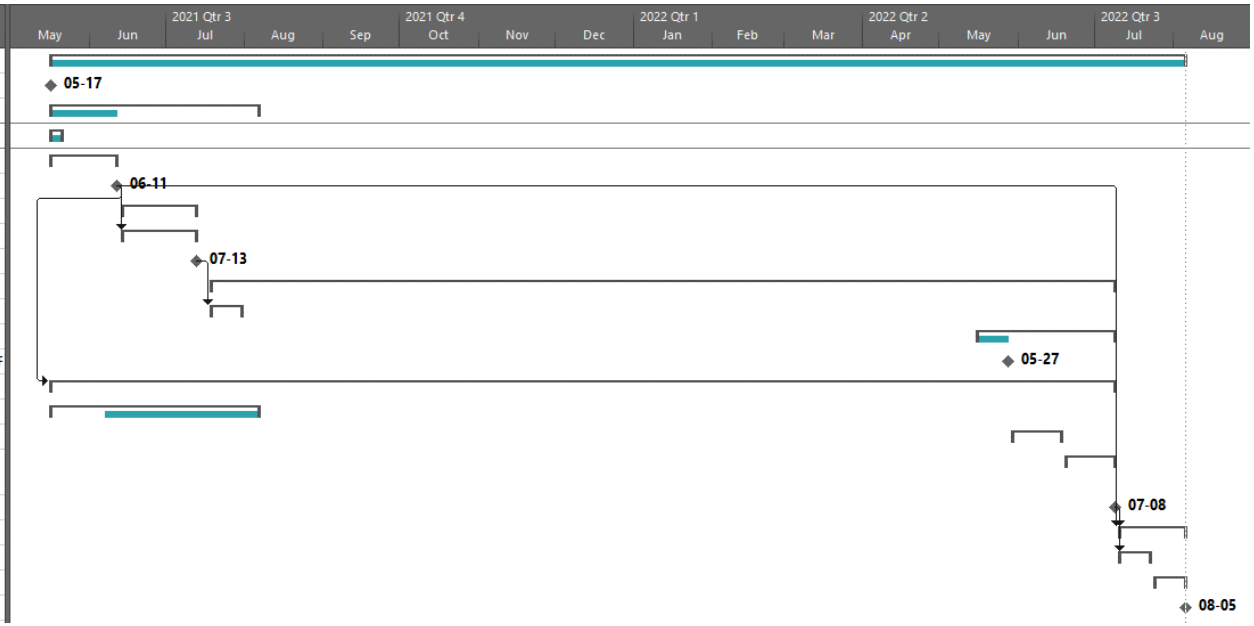

Figure 4: High Level Gantt

Task Name	Duration	Start	Finish	Predece
REA Project	**320 days?**	**Mon 21-05-17**	**Fri 22-08-05**	
Project Kickoff	0 days	Mon 21-05-17	Mon 21-05-17	
Objective 1: Create dataset	**60 days**	**Mon 21-05-17**	**Fri 21-08-06**	
Phase 1.1: Deploy testbed	**4.5 days**	**Mon 21-05-17**	**Fri 21-05-21**	
Deploy Kali (attacker)	1 day	Mon 21-05-17	Mon 21-05-17	
Deploy Windows MS3 (victim)	1 day	Tue 21-05-18	Tue 21-05-18	5
Deploy Ubuntu monitoring server, configure Zeek	1 day	Wed 21-05-19	Wed 21-05-19	6
Configure Zeek to extract files	1 day	Thu 21-05-20	Thu 21-05-20	7
Configure port mirroring through managed vSwitch on ESXi hypervisor	0.5 days	Fri 21-05-21	Fri 21-05-21	8
Validate network connectivity and configuration	0.5 days	Fri 21-05-21	Fri 21-05-21	5,6,8
Phase 1.1 completion	0 days	Fri 21-05-21	Fri 21-05-21	10
Phase 1.2: Create dataset	**20 days**	**Mon 21-05-17**	**Fri 21-06-11**	
Develop automation for creating dataset (scripts Malicious Payloads, Benign Payloads, Host Files and Download Files)	1 wk	Mon 21-05-24	Fri 21-05-28	11
Test and debug dataset automation ((scripts Malicious Payloads, Benign Payloads, Host Files and Download Files))	4 days	Mon 21-05-31	Thu 21-06-03	13
Run scripts (scripts Malicious Payloads, Benign Payloads, Host Files and Download Files)	1 day	Fri 21-06-04	Fri 21-06-04	14
Create script "Analyse File"	1 wk	Mon 21-05-17	Fri 21-05-21	2
Test and debug script "Analyse File"	1 wk	Mon 21-05-24	Fri 21-05-28	16
Generate dataset	1 wk	Mon 21-06-07	Fri 21-06-11	15,17
Phase 1.2 completion	0 days	Fri 21-06-11	Fri 21-06-11	18FF
Objective 1 completion	0 days	Fri 21-06-11	Fri 21-06-11	11,19

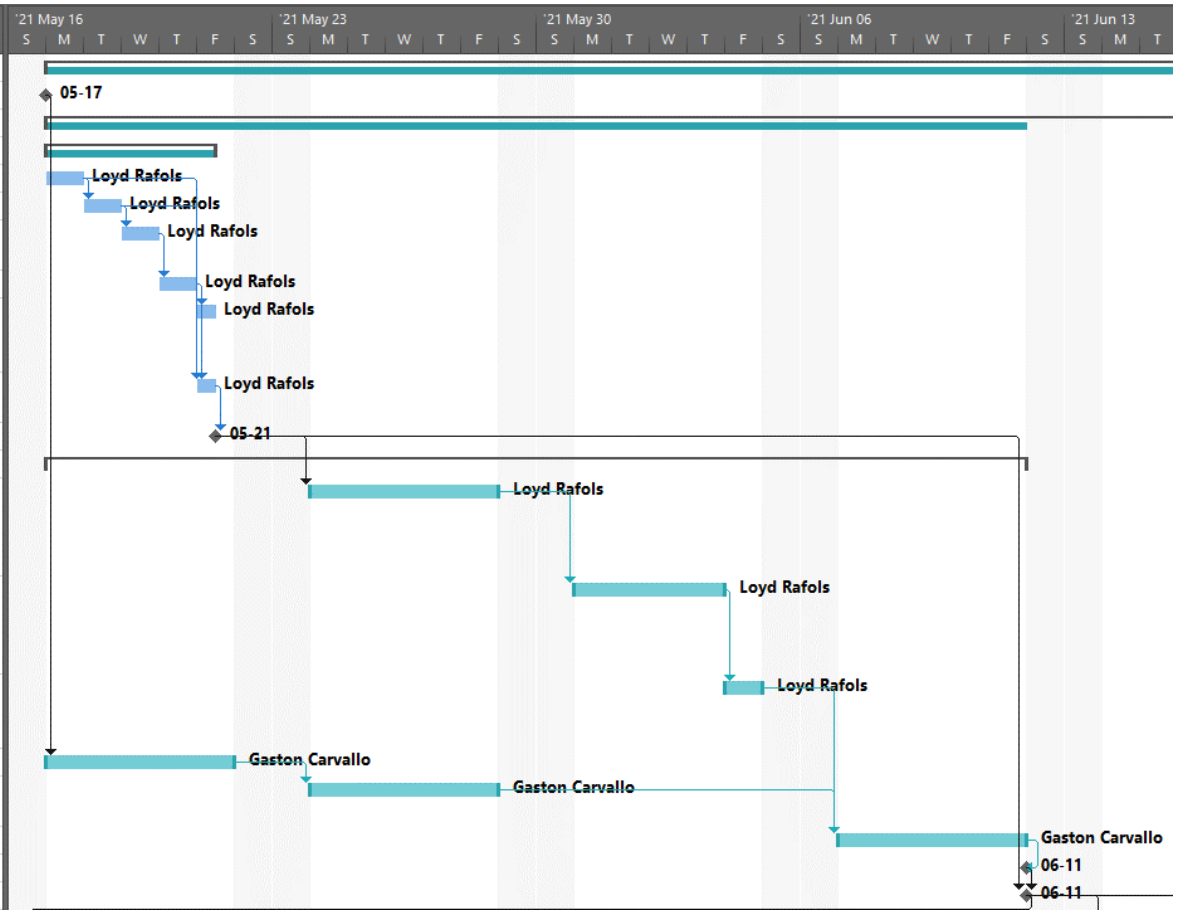

Figure 5: Detailed Gantt - Objective 1

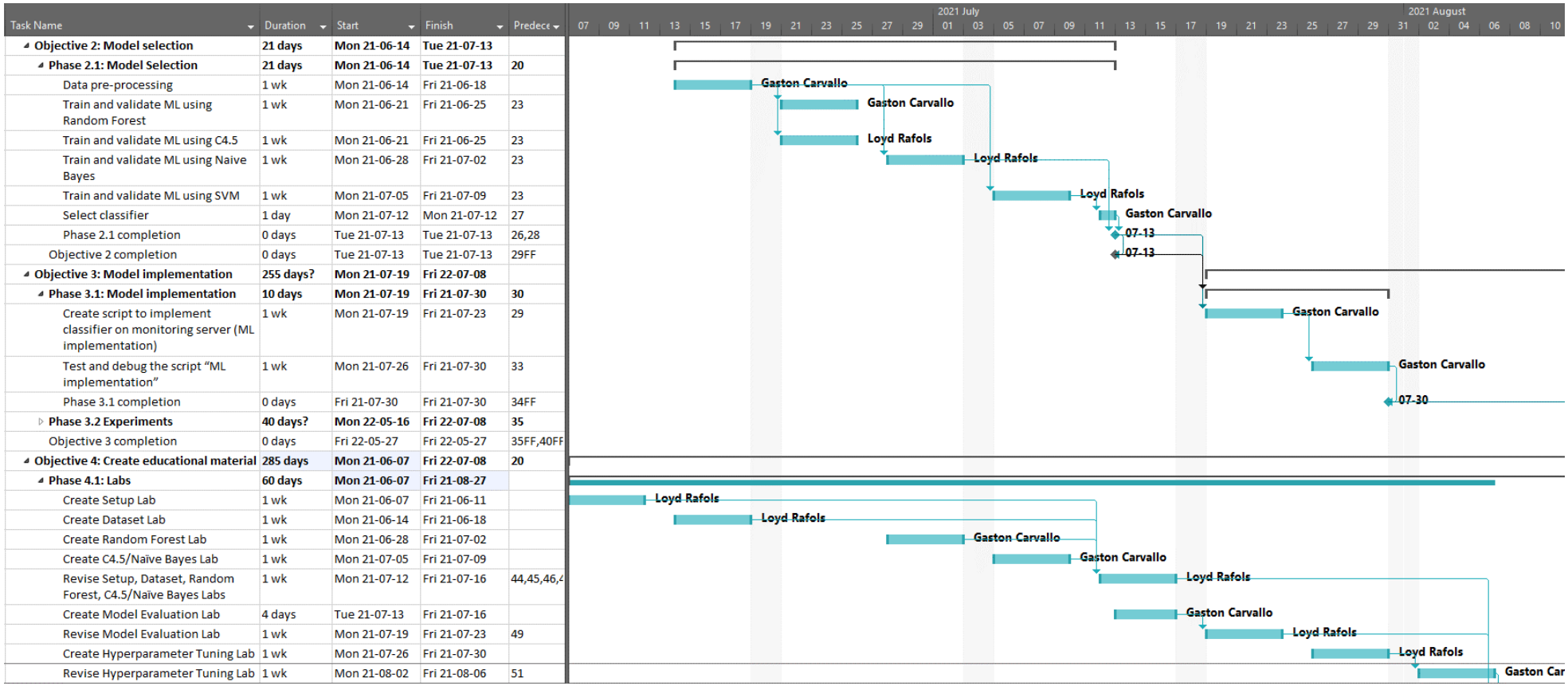

Figure 6: Detailed Gantt - Objectives 2-4 (Phase 4.1)

Task Name	Duration	Start	Finish	Predece
Phase 4.2: PowerPoint Presentations	**15 days**	**Mon 22-05-30**	**Fri 22-06-17**	**40**
Create Setup Lab tutorial presentation	5 days	Mon 22-05-30	Fri 22-06-03	
Create Dataset Lab tutorial presentation	5 days	Mon 22-06-06	Fri 22-06-10	55
Create Random Forest Lab tutorial presentation	5 days	Mon 22-05-30	Fri 22-06-03	
Create C4.5/Naïve Bayes Lab tutorial presentation	5 days	Mon 22-06-06	Fri 22-06-10	57
Create Model Evaluation Lab tutorial presentation	1 wk	Mon 22-06-13	Fri 22-06-17	56
Create Hyperparameter Tuning Lab tutorial presentation	1 wk	Mon 22-06-13	Fri 22-06-17	58
Phase 4.2 completion	0 days	Fri 22-06-17	Fri 22-06-17	55,56,57,5
Phase 4.3: Presentation and tutorial revision	**15 days**	**Mon 22-06-20**	**Fri 22-07-08**	**61**
Revise Setup and Dataset Lab	1 wk	Mon 22-06-20	Fri 22-06-24	61
Revise Model Evaluation Lab	1 wk	Mon 22-06-27	Fri 22-07-01	63
Revise Random Forest and C4.5/Naïve Bayes Lab	1 wk	Mon 22-06-20	Fri 22-06-24	61
Revise Hyperparameter Tuning Lab	1 wk	Mon 22-06-27	Fri 22-07-01	65
Review and provide feedback on other group member's labs and presentations	1 wk	Mon 22-07-04	Fri 22-07-08	63,64
Review and provide feedback on other group member's labs and presentations	1 wk	Mon 22-07-04	Fri 22-07-08	65,66
Phase 4.3 completion	0 days	Fri 22-07-08	Fri 22-07-08	63,64,65,6
Objective 4 completion	0 days	Fri 22-07-08	Fri 22-07-08	69,61,53

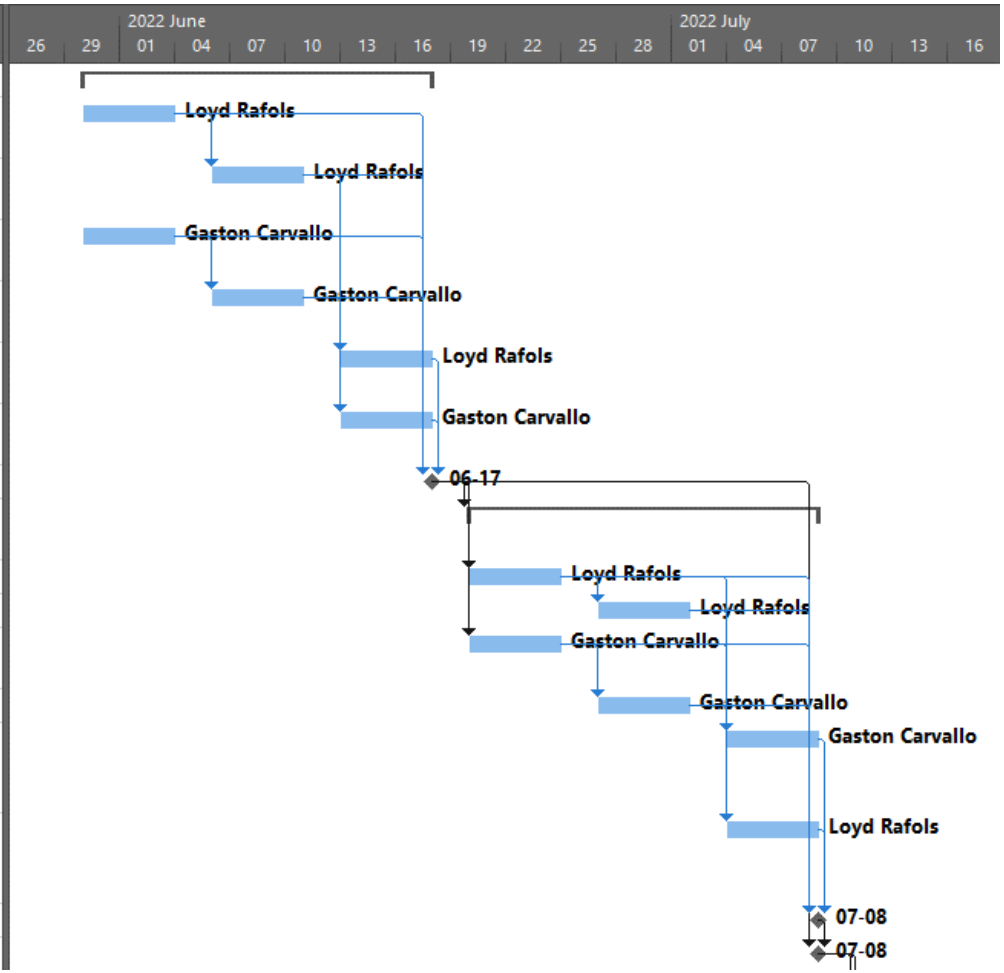

Figure 7: Detailed Gantt - Objective 4 (Phase 4.2, 4.3)

Task Name	Duration	Start	Finish	Predece
Objective 2: Model selection	**21 days**	**Mon 21-06-14**	**Tue 21-07-13**	
Objective 3: Model implementation	**255 days?**	**Mon 21-07-19**	**Fri 22-07-08**	
Objective 4: Create educational material	**285 days**	**Mon 21-06-07**	**Fri 22-07-08**	**20**
Objective 5: Create final report	**20 days**	**Mon 22-07-11**	**Fri 22-08-05**	**20,70**
Phase 5.1: Report Drafting	**10 days**	**Mon 22-07-11**	**Fri 22-07-22**	**70**
Write abstract and introduction	1 wk	Mon 22-07-11	Fri 22-07-15	70
Write literature review and background	1 wk	Mon 22-07-11	Fri 22-07-15	70
Write research methodology and experiments	1 wk	Mon 22-07-18	Fri 22-07-22	73
Write discussion and conclusion	1 wk	Mon 22-07-18	Fri 22-07-22	74
Phase 5.1 completion	0 days	Fri 22-07-22	Fri 22-07-22	73,74,75
Phase 5.2: Final Report	**10 days**	**Mon 22-07-25**	**Fri 22-08-05**	**77**
Revise abstract and introduction	1 wk	Mon 22-07-25	Fri 22-07-29	77
Revise literature review and background	1 wk	Mon 22-07-25	Fri 22-07-29	77
Revise methodology and experiments	1 wk	Mon 22-08-01	Fri 22-08-05	79
Revise discussion and conclusion	1 wk	Mon 22-08-01	Fri 22-08-05	80
Phase 5.2 completion	0 days	Fri 22-08-05	Fri 22-08-05	79,80,81
Objective 5 completion	0 days	Fri 22-08-05	Fri 22-08-05	77,83

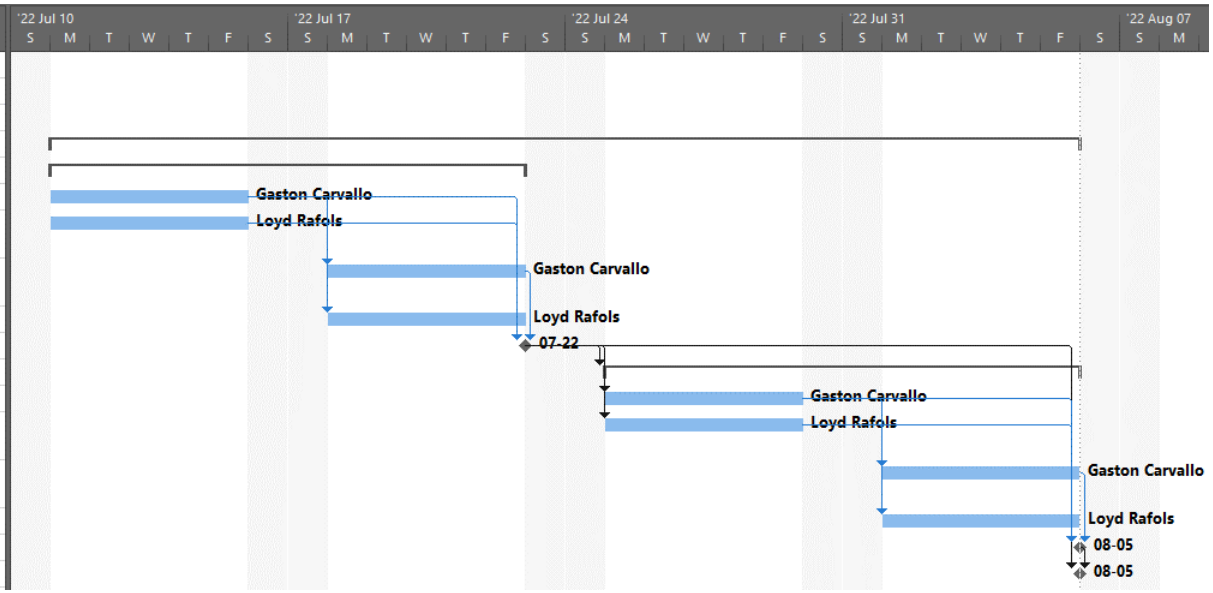

Figure 8: Detailed Gantt - Objective 5