Malware Analysis with Machine Learning Methods

Final Year Dissertation Deliverable 1

Cedric Ludo F. Ecran
Supervisor: Dr. Hani Ragab Hassen

B.Sc. (Hons) Computer Systems

Declaration

I, Cedric Ludo F. Ecran confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of references employed is included.

Signed: Cedric Ludo F. Ecran

Date: 22/11/2019

Abstract

With the growing amount of data in cyberspace so too does the number of malicious files, seeking to harm unsuspecting users. Malware can be very tricky to avoid as the expanding collection and complexity are overwhelming the current signature-based solutions. Machine Learning could be the future with promises of efficiency and a higher detection rate of zero-day malware then current anti-viruses. In this final year dissertation, we seek to access and implement research-based proposals on malware analysis utilizing machine learning methods.

Table of Contents

1. Introduction

1. 1. Context

The rate of malware is ever-expanding at an alarming rate, Kaspersky reported that in the first quarter of 2019 alone there were more than 150 million malware attacks in just the META region, averaging out to approximately 1.6 million per day [1]. These figures indicate a 108% increase over the number of malware attacks in the first quarter of 2018 [1]. The unprecedented scale of malware attacks highlights the need for machine learning-based detection mechanisms that operate effectively and efficiently to combat the ever-growing influx.

1. 2. Aim

The dissertation aims to critically review and implement a selection of machine learning-based malware detection research proposals to evaluate the results achieved. We will focus on static based analysis research proposals with the primary factor being the achieved accuracy of each reviewed proposal.

1. 3. Objectives

The objectives of the dissertation are the following:

- Construct an extensive dataset of both malicious and benign files, with all files passing validation tests from current anti-virus solutions
- Perform feature extraction and reduction on the dataset
- Apply various machine learning techniques from the selection of research proposals with the extracted features from the dataset
- Evaluate the achieved results between the implemented machine learning techniques

1. 4. Introduction to Malware and Machine Learning

1. 4. 1. Malware

The term malware is a combination of the words malicious and software hinting heavily to its definition. The definition of malware, also known as malicious software, is described as any software that brings harm to either the user, computer, or network that it is executed on [2]. Malware can adopt a wide variety of forms including viruses, worms, rootkits, and trojans although the basic principle stands for all variants. In this dissertation project all known types of malware, in the Windows executable format, will be utilized for training and testing purposes.

1. 4. 2. Machine Learning

Machine learning (ML) grew out of the broader field of Artificial Intelligence whereby machines attempt to mimic human intelligent abilities [3]. In Machine Learning algorithms the "learning" element involves training from examples to perform their task rather than traditional ways of having pre-defined rule sets [4]. Machine learning aims to predict, and correctly classify/label unseen data based on experience/training [3]. There are two types of learning unsupervised and supervised, unsupervised revolves around detecting either anomalies within the input data or hidden regularities [3]. In supervised learning the algorithm attempt to assign a label to each example to classify based on the previously seen examples [4]. In unsupervised the data has no labels or meanings to the algorithm whereas supervised does, this labeling allows for the effective classification of malware by supervised learning machine learning algorithms. We will focus on supervised learning techniques for training and testing for detecting malicious files.

2. Literature Review

2. 1. Background Theoretical Research

2. 1. 1. Malware Analysis

The goal of malware analysis is to extract information from a given malware sample with the intent to identify and possibly classify. The reason for doing so usually involves accessing the damage dealt by it, examining the security breach on the target system, and identifying defense vulnerabilities that were exploited [5]. The process involves two main techniques: Static analysis where malware is examined in a “dead” state and Dynamic analysis where the malware is executed to be observed [5].

Static analysis is normally the faster and easier approach to malware analysis due to none of the overhead that comes with executing malware. Static analysis involves two main techniques the first, and simplest, involves examining the Portable Executable (PE) a file format containing information used by the OS loader to execute the code that it is wrapped around [6]. The most useful information, from a malware analysis perspective, located in the PE are the import functions that the executable calls. These imports allow a program to call functions stored in a different program. This becomes particularly important once you consider code libraries are linked in this fashion, thereby allowing the malware creator to call functions without the code to perform that function being present within the malware [6]. The import functions can be implemented in several ways, one of which is statically whereby the linked library is loaded into the executable before any code is run. The more commonly utilized way is by dynamic linking whereby the libraries are not loaded into the executable, rather the functions are run within the libraries themselves, and are only run once called upon [6]. This is useful in two ways for the malware, it keeps the malware's size limited as it does not load the libraries thus keeping efficiency high. Secondly by not loading in any libraries the malware does not change thereby possibly helping to avoid detection from file fingerprinting. File fingerprinting is the process of hashing a file, performing some operation on/with it, and then once complete, rehashing it and comparing it to the original hash to ascertain if changes have occurred to the file [5]. The use of dynamic link libraries (DLLs) in malware design is widespread not only because of the efficiency and possible obfuscation of the true nature of the malware, but additionally due to DLL's being very widespread and common. Within the Windows OS it allows malware engineers to reuse them thereby allowing for code reusability and efficiency when creating additional malware [6]. The other, more advanced, static analysis technique requires disassembling the given malware to reveal the operations it will execute. Once disassembled the operation codes (opcodes), instructions to the CPU detailing operations that need to be performed, are commonly examined as they are the lowest level building blocks of any executable, being the literal commands passed to the CPU [6].

Dynamic analysis allows the malware to execute to observe and possibly alter the malware as it executes [5]. Dynamic analysis has a distinct advantage over static analysis due to being able to observe certain runtime features and operations that may not have been detected with static analysis. However; dynamic analysis requires a greater amount of setup and caution, due to the presence of active running malware. This will usually involving a safe environment (virtual machine), a process monitor [5], and a debugger to view the runtime state of the malware [6]. Although extremely exhaustive static analysis could, in theory, reveal all possible functionalities of a given malware, this would be extremely difficult and time-consuming [5]. In modern malware analysis elements from both static and dynamic analysis are used to gain a complete view of the analyzed malware [6].

2. 1. 2. Machine Learning

Within the field of machine learning problems can be labeled as one of two types of problems, classification or regression [3]. In a classification problem, the labeling of the done by the algorithm of the input data is discrete, an example being black or white, whereas with a regression problem the labels are more complex or real-valued [3]. The scope of use for this dissertation revolves around the classification problem of files, with their label being either malicious or benign. There are a wide variety of classification algorithms within the machine learning field; however, the focus for this dissertation will be on the most commonly used, based on researched papers on malware analysis using machine learning.

The k-Nearest Neighbor classification involves the training data being stored within a dimensional space in the form of points/positions, one for each piece of data. When the testing data is classified as its data, points are also classified into this space. Based on the distance to k amount of training data points classification is given to the test data [3]. This method is likely the simplest classification method to understand and has, on average, a very low error rate but the memory storage requirements and high computational workload make it not optimal for large datasets [3].

Another classification method is decision trees whereby, from training data, a tree is constructed based on partitioning the input space [3]. The tree is constructed from the training data by constantly selecting the best/pure discerning features for each of the nodes of the tree [3]. The tree keeps expanding if input from the training data cannot be successfully classified by traversing down the given tree. Classification of the test data is also done by traversing down the tree, with the leaf nodes containing the classification labels [3]. Decision trees work best with smaller datasets with low dimensionality, as they do not scale well when increasing these factors and being to require a large amount of memory storage and increase the risk of overfitting [3] [5]. To reduce these effects we can either limit the size that the tree can grow to, or perform pruning on the completed tree, whereby nodes are replaced with either sub nodes or leaves based on a bound or estimate [5]. Another implementation of decision trees by J. Quinlan is named C4.5 and is commonly used within malware analysis using machine learning research [3].

Further development of decision trees is the random forests whereby a large amount, unusually in the hundredths, of decision trees, are created using some part of the training data [5]. For each tree, only a random subset of the feature list is considered from the training data [4]. Once unseen data, test data, is passed to the random forest, each iteration of the decision tree will classify the data [4]. Based on the results of classification, each tree votes for classification of the input data with the majority classification being adopted [4].

The above methods within this section ensured that classification was clear and definitive for its constructed model but if new unseen data, which is somewhat different from the training data used, are used they may either fail or become computationally infeasible [5]. The possible solution is to use algorithms that allow for generalization, the abstractification of data, one of which is support vector machines (SVM) [3]. In SVM the training data is mapped into a feature space where the data points are separated by a large margin hyperplane [3]. This hyperplane needs to have a large margin, the smallest distance between the hyperplane and a data point, to classify new data [5]. New data will be mapped into the feature space, due to the large margin the new data will either be on one of two sides of the hyperplane thereby classifying it [3]. The exact distance from the hyperplane is not considered as it allows for generalization of the data [5].

Another algorithm that allows for generalization is boosting whereby linear predictors, the output from other classification algorithms, are weighed and then combined to create a final prediction [3] [5]. This generally results in significant performance improvements over the standalone algorithms [3]. The primary boosting algorithm that will be focused on is Adaptive Boosting (AdaBoost), the very first practical implementation of boosting [5].

2. 2. Related Works

Liu Liu et al. [2] proposed using static analysis with the use of multiple feature extraction process, greyscale imaging, operation code (Opcode) features, and dynamic link library calls to more accurately locate the frequently used functions within the malware. Then extract the Opcode features with an n-gram model, performing text feature extraction, combined with a control flow graph (CFG). The CFG allows for relationships between the functions to be considered as it shows a high-level overview of the malware whereas the n-gram model searches localized.

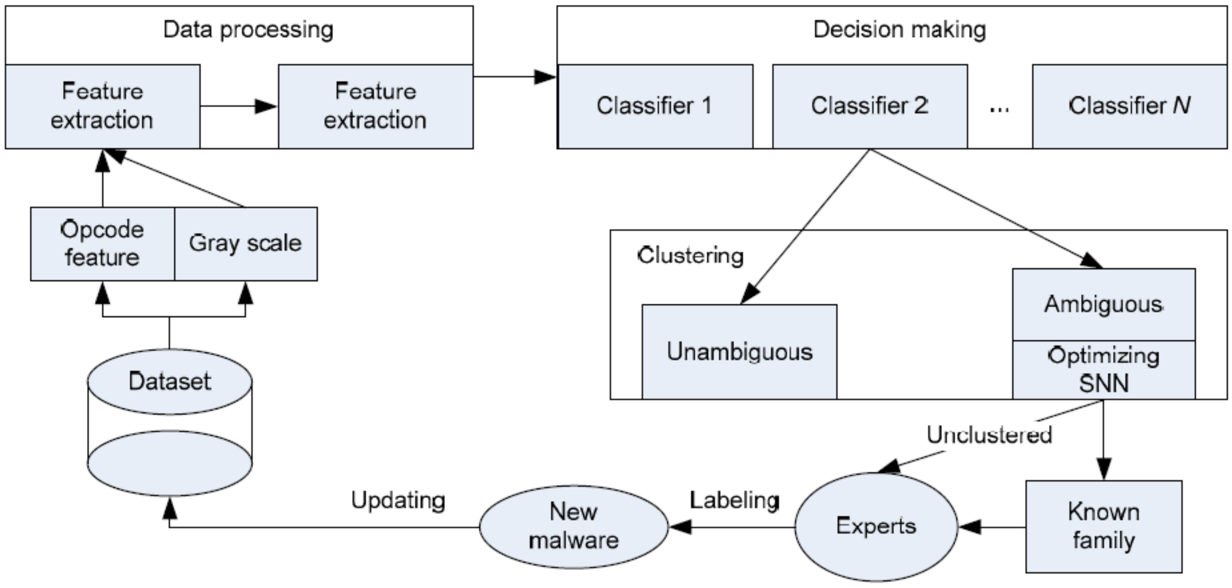


Figure 1 showcases a high-level overview of source [2] implementation

The malware collection used totaled 21,740 instances consisting of nine different malware types. After feature extraction a variety of machine learning classifiers consisting of random forests (RF), K-nearest neighbor (K-NN), Gradient-boosting (GB), Naïve Bayes (NB), logistic regression (LR), support vector machine-poly (SP) and decision trees (DT) were applied on known malware but only with varying levels of n-gram feature extraction. The results indicated that 2- and 3-gram feature extraction were most accurate with the 3-gram RF approach achieving a 94.7%. The next stage was to combine the greyscale imaging and the 3-gram feature extractions which, when tested overall classifiers, averaged a 95.7% accuracy compared to the 3-gram average of 84.3% and the greyscale imaging average of 90.7%. The highest combined feature extraction accuracy was the RF classifier, which managed an accuracy of 98.9%.

After classification of the known malware, clustering was performed on the data and an estimated eight clusters were formed. Then another testing sample was tested but this time the sample consisted of 900 samples, of which 90 were new unknown samples but of the same family. Each classifier would vote on the likelihood of the file being suspicious, therefore malware. The experiment showed that the new malware was classified accurately 86.7% of the time with a new cluster being formed.

Critical Analysis

The malware dataset could be bigger, and we are also not told the size of the benign dataset, as this could inflate the accuracy if the benign dataset is very small. There is no mention as to the origin of the dataset, nor if any validation with existing anti-virus tools was performed. The missed opportunity of including windows API calls into the feature extraction as this may have resulted in higher accuracy. They could also have included dynamic analysis to possibly integrate the results like [6] to possibly improve accuracy but to combat anti-analysis techniques. When performing their classification, they should have performed the k-fold cross-validation technique to increase the validity of their results. The lack of false negative and false positive data is unfortunate as it would give greater insight into the performance of the implemented machine learning method. The interesting element of this paper is the use of greyscale imaging combined with the more traditional static analysis approaches of Opcode and DLL calls. The use of greyscale could be better served by comparing before and after the execution of the program to see how much has changed and if that has a strong correlation to the program being malicious.

Jingling Zhao et al. [7] combined both static and dynamic information extraction before applying feature extraction on the combined information. In static extraction, they used Interactive Disassembler Professional (IDA) and IDA Python to disassemble their malware samples. The focus on static features was on the types of string’s present and the DLL import functions with their frequencies being noted. The dynamic information extraction was done in the Windows operating system with the malware being monitored with Intel PIN. Intel PIN provides varying levels of program execution information such as a function level, instruction level, etc. The monitoring of the malware samples was focused on the assembly instructions the malware used and the API calls initiated during execution.

The feature selection consisted of two algorithms Information Gain and N-gram algorithm. Information gain looks at the amount of information that a certain feature adds to the classification system. The N-gram algorithm was used on the malware Opcodes performing a sliding window with a size of N. Two ML classification algorithms were tested with the now extracted feature set, one being Naïve Bays and the other Support Vector Machines (SVM).

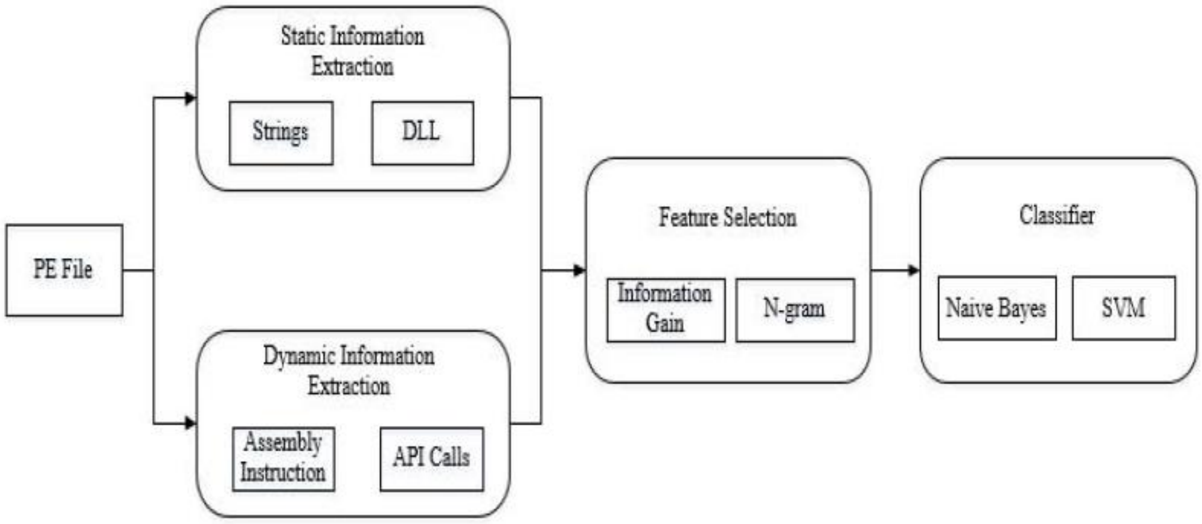


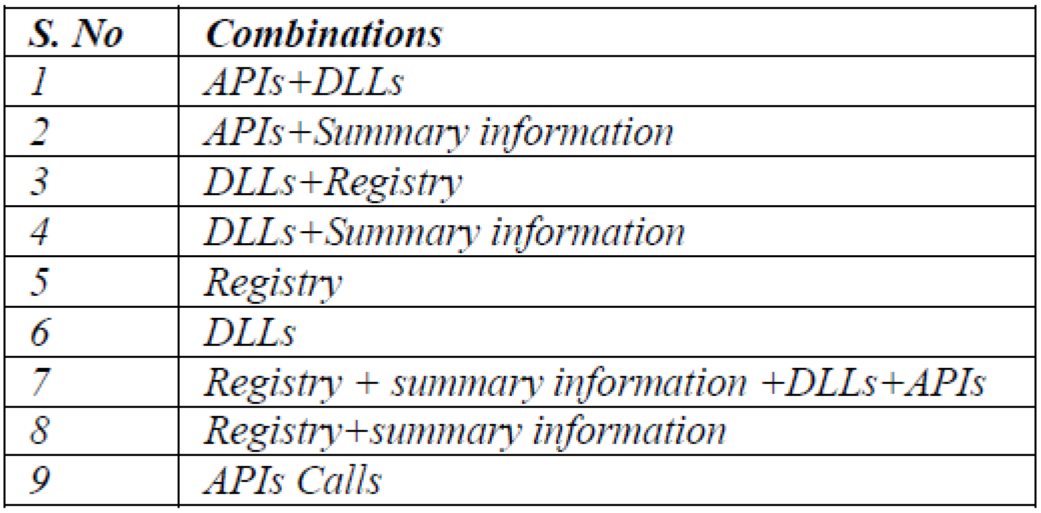
Figure 2 is the flowchart of the source [7] malware detection system

The dataset used consisted of 3,548 malware and 1,628 benign programs, the malware programs were collected from a website named Virrussign. The benign files were interestingly not a Windows OS executable collection, as they believed this to have a too high degree of similarity between the files, rather a collection was acquired from malwr.com. The results showed that the feature extraction/reduction obtained the top 200 API functions and strings, along with the top 100 DLL imports. The achieved accuracy using the Naïve Bays classification algorithm was 97% and for the SVM they achieved 98%.

<u>Critical Analysis</u>

The dataset is small but the ratio of benign to malware is good. The use of only two classification algorithms is puzzling, certainly when it is suggested from most research papers on the matter that random forests usually perform very well with mid-sized datasets like the one used here. When performing their classification, they should have performed the k-fold cross-validation technique to increase the validity of their results. There is no mention of the false negative/false positive rates, these should be included to give a better indication of the performance of implemented machine learning methods. The integration of static and dynamic features is similar to [6] implementation, but they each have separate reasons. An interesting note from this paper is the use of non-Windows OS executable collections as the OS due to similarity concerns. This could be a valid argument as the files usually do exhibit similar features and therefore be boosting the accuracy of detecting them.

Muhammad Ijaz et al. [8] compared static and dynamic malware analysis, and combinations of multiple dynamic features, to determine which achieves the highest attainable malware detection accuracy. The extraction of static features was done using a python library called Portable Executable PEFILE. PEFILE provided PE file header from which many features were recorded such as section size, size of data, and time and date stamps. Dynamic feature extraction was done using Cuckoo sandbox whereby registry keys, file state, API calls, and IPS and DNS queries were monitored and recorded. They then decided on combining multiple variants of the dynamic features, all of which can be seen in figure 3 below.



S. No	*Combinations*
1	*APIs+DLLs*
2	*APIs+Summary information*
3	*DLLs+Registry*
4	*DLLs+Summary information*
5	*Registry*
6	*DLLs*
7	*Registry + summary information +DLLs+APIs*
8	*Registry+summary information*
9	*APIs Calls*

Figure 3 shows the dynamic feature combinations tested

The ML classification algorithms used were Logistic Regression, Decision Tree, Random Forest, Bagging Classifier, AdaBoost Classifier, Tree Classifier, and Gradient Classifier. The non-combined dynamic results showed that Gradient Classifier achieved the highest accuracy of 94.64% with a false positive of 5.85% and a false negative of 13.96%. This proved to be the lowest false positive percentage with the AdaBoost Classifier achieving the lowest false negative result of 10.91%. The next set of results are for the combined dynamic features using the same ML classification algorithms. The results showcase only the highest achieved accuracy with a given ML classification algorithm and with no false positives or negatives. The top accuracy was achieved by the combination of API's and summary information (S. No 2 in figure 3) with an accuracy of 95.86%. The final set of results are for the static analysis features, once again utilizing the same ML classification algorithms. The highest achieved accuracy with static features was 99.36% using the Gradient Classifier, this resulted in a false positive percentage of 3.25% and a false negative of 2.73%. The lowest false positive percentage was achieved by Random Forest with 3.09% and the lowest false negative by the Bagging Classifier reaching only 0.60%.

Critical Analysis

Within the paper, they state that the malware dataset was "small", not a very descriptive answer and therefore I can only conclude that it would be a very small number. This is possibly why they got

99.36% using gradient classifiers with static analysis. The k-fold cross-validation technique would certainly have shown a more accurate accuracy as it would make the most out of the “small” dataset. The accuracy improvement between the single dynamic feature and the combinations was rather small of approximately 1.2%. The results they achieved for decision trees using static analysis features have a possible indication of overfitting as the false negative it achieved was a minuscule 0.665%. Their method for extracting the static features, via PEFILE, is a tool we are considering for this dissertation. In the future work section, they theorize that by allowing the program to execute in a dynamic environment you mitigate most of the anti-analysis issues that come with static analysis as described in source [6]. Once the program has executed then you would perform the static analysis to obtain very accurate and efficient results.

Usukhbayar Baldangombo et al. [9] propose combining data mining methods with machine learning to achieve high classification accuracy. The dataset includes 247,348 files all in the Windows operating system executable format of which 236,756 are malicious and 10,592 benign. The malicious files were constructed from several public domain sources one of which is the VX Heavens Virus Collection. The benign section of the dataset was obtained from Download.com and are Windows system files. The authors developed their portable executable parser to perform static analysis named PE Miner which extracted all the PE header information, API function names, and DLL names located in the PE. Feature extraction revealed the optimal features as the top 88 PE header features, top 130 frequent DLL names, and seven groups of frequently used API’s by the malicious files that translates to 2,453 API functions. The classifiers used were support vector machines, decision tree-J48 (java specific version of C4.5), and Naïve-Bays. Cross-validation was performed 10 times to assist in validating the robustness of the technique.

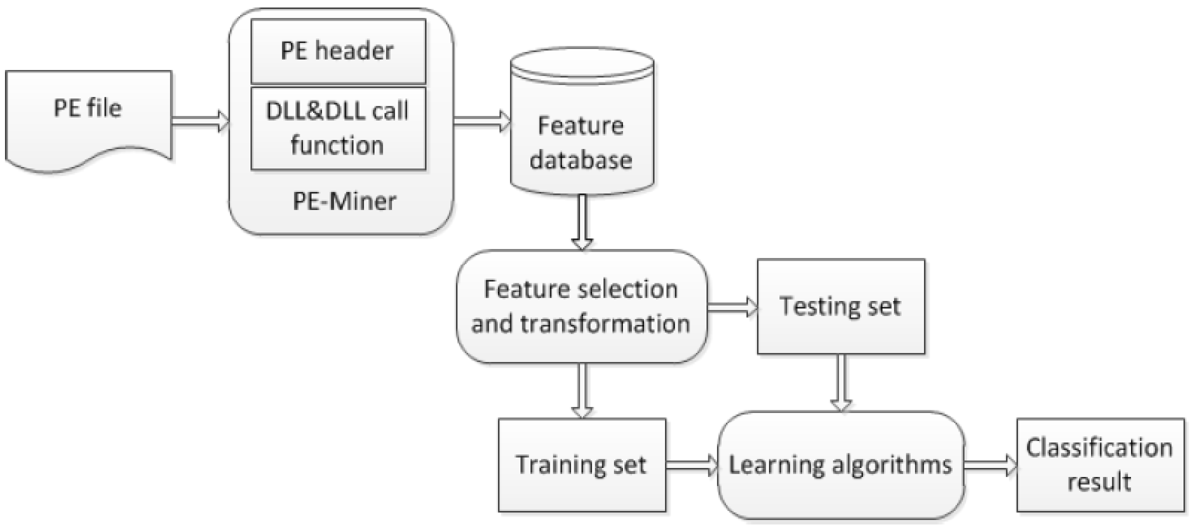


Figure 4 from source [9] showcases the malware detection system architecture

The first set of results was to see what impact that Principal Component Analysis (PCA) had on the direct detection rate of the J48 classifier. PCA is a data mining function, when used on the features, drastically reduced them by as much as 87%. By performing PCA on the selected features the detection rate of the J48 classifier increased for all three feature categories, with a maximum improvement of 1.3%. The final set of results paired all three classifiers with each of the feature types, and two hybrids the first is PE header and DLLs, the second PE header and API functions. The results showed that the highest direct detection rate came from using the PE header as feature type and the J48 classifier obtaining a 99.5%. This combination also obtained the lowest false positive rate of only 2.7% and the second-highest overall accuracy at 99%. The highest overall accuracy was achieved by combining API functions as the feature type with the J48 classifier, this achieved a 99.1% accuracy.

Critical Analysis

A very well sized dataset but could use more benign files as there is currently some oversampling going on in favor of malware. In-depth static analysis was performed but the lack of information on exactly how their feature selection was done is questionable. The addition of the data mining technique, PCA, brought surprising results by increasing the already very accurate J48 classifier with PE headers by another 0.4%. Their results show that the hybrid implementations did not achieve the same high levels of accuracy as the single feature ones. This could be due to the increase in complexity as algorithms with more generalization will perform better with higher complexity. This is also validated when looking at the SVM results as they are the only ones that increase with the hybrid implementation of the feature selection.

Akash Kumar Singh et al. [6] attempt to design a machine learning method for the detection of malware that has been designed with anti-analysis features by integrating both static and dynamic analysis. The dataset is in Windows OS executable format and has various types of malware present. The size of the dataset is 109 including 25 benign and 84 malicious files. The virtual machine used for dynamic analysis is NORIBEN. The static analysis is combined with the output of the various anti-analysis features that the file may include, these are determined during the execution of the program with dynamic analysis. The first module is the anti-VM module whereby we check via the Interrupt Descriptor Table Register (IDTR) whether the file detects that it's running in a virtual machine. The second is the anti-debugging module whereby looking at certain windows API calls, like IsDebuggerPresent, if the malware has detected a debugger. The third is the packer detection module, here we check if the file has been packaged and in what way, this is done by using PEiD. The final module is the URL analysis module where URLLIB is used to extract the URL connection, strings, and file information.

The data from both the static and dynamic analysis are integrated to ensure that even if the malware avoids static analysis detection the dynamic analysis results will still distinguish it as a malicious file. A free machine learning tool was used to perform a classification called WEKA. Three classifier methods were implored, Naïve Bays, support vector machines, and random forests. 10-fold cross-validation was used to provide robustness to the results of the paper. The classification was conducted on static analysis features, dynamic analysis features, and the integrated analysis features. The static analysis features results showed that SVM obtained the highest accuracy at 71%. The dynamic analysis features results indicated that random forests have the highest accuracy of 63.3%. The results using integrated analysis features showed random forests to have the highest accuracy of 73.4%.

Critical Analysis

The extremely small dataset of only 109 files total makes the results questionable at best. The use of k-fold cross validation does attempt to help but the dataset is so small there may not be much of an improvement to the result validity. The research paper looks at the anti-analysis design for certain malware types and offers some interesting solutions to detect them. The feature selection and integration of both the static and dynamic analyses attempted to show the improvement over single feature use. This is similar to the implementation in source [7] but the idea behind it is different. Overall accuracy did improve one the integrated features were used with the classifiers showcasing that their ML method could produce more accurate results, but with such a small dataset more training and testing needs to be done.

2. 3. Critical Reflection

Table 1 research papers on malware analysis with machine learning methods

Source	**Static/Dynamic Analysis**	**Dataset Size**	**Feature Extraction**	**Classification Algorithms**	**Highest Obtained Accuracy**	**False Negative Rate**
[2]	Static	21,740	Greyscale, Opcodes, DLL's	RF,K-NN,GBC,NB,LR,SVM,DT	RF 98.9%	NA
[7]	Static & Dynamic	5,176	Strings, DLL's, Assembly Instructions, API Calls	NB, SVM	NB 95%	NA
[8]	Static & Dynamic	"small"	API Calls, DLL's, Registers, Summary Information	LR, DT, RF, BC, AD, TR, GB	Static GB 99.36%	2.73%

[9]	Static	247,348	PE Header, DLL’s, API Calls	NB, SVM, C4	API Calls C4 99.1%	NA
[6]	Static & Dynamic	109	(Anti-VM, Debugging, Packer, URL Outputs), Functions, API Calls	NB, SVM, RF	Integrated RF 73.47%	NA

Index: Random Forests (RF), K-nearest neighbor (K-NN), Gradient-boosting (GB), Naïve Bayes (NB), Logistic Regression (LR), Support Vector Machine (SVM) and Decision Trees (DT), Bagging Classifier (BC), AdaBoost Classifier (AD), Tree Classifier (TR), C4.5/J48 (C4)

In conclusion, we have discussed a multitude of research papers regarding malware analysis using machine learning methods. Some papers also included topics such as data mining techniques for feature reduction [9] and clustering to create clusters or groups of varying types of malware [2].

Table 1 above displays the prominent attributes across multiple papers. A common feature is the presence of DLL’s in the feature extraction of all the papers. Another common factor, apart from source [9], are the small or very small dataset sizes that question the validity of the above results. A multitude of the papers discussed integrated feature selection/reduction from both static and dynamic analysis. These results have yielded promising improvements over just using one set of features but more in-depth research needs to be conducted in that field before concrete conclusions can be made. Overall of the results from the various papers, random forests performed very well when compared to the other more traditional classification algorithms. In terms of the large margin algorithms SVM was used in nearly all papers but generally, the boosting classifiers, Gradient and AdaBoost, did outperform it when they were employed. [8]

This literature review has showcased in-depth knowledge with regards to malware analyses and machine learning in the context of this dissertation. We have also explored multiple research papers based on the topic. We can now clearly see the need for more research into the field and have indications/trends in terms of what features and classifiers produce superior accuracy.

3. Requirements Analysis and Research Methodology

3. 1. Introduction

The requirements analysis focusses on the requirements necessary for the implementation of the various malware detecting machine learning techniques. This involves exploring the various Functional (FR) and Non-Functional (NFR) Requirements. Functional Requirements explore what desired functionality a system should be capable of achieving whereas Non-Functional Requirements focus on how, and to what degree, this will be achieved.

The research methodology links directly to the objectives and aims of the dissertation with the evaluation of the results obtained from implementing the malware detecting machine learning techniques. The major quantitative discerning factor will be the achieved accuracy of each technique but also the calculated false positive and false negative percentages. This will allow for a more holistic and real-world performance indication performance comparison aiming to locate the optimal technique for machine learning-based malware detection utilizing static analysis.

3. 2. Requirements

ID	**Type**	**Priority**	**Requirement**
FR 1	Dataset	Must have	The dataset must have two separate sections of malware and benign files for testing purposes and all files must be validated to their respective category
FR 2	Features	Must Have	Feature extraction and reduction must be executed on all files from the dataset used for testing purposes
FR 3	ML Technique	Must Have	Training of the various ML techniques must be performed on the training dataset
FR 4	ML Technique	Must Have	The program must produce a confusion matrix stating the achieved accuracy, false positive, and false negative percentiles
NFR 1	Dataset	Must Have	The dataset must contain at minimum 25,000 examples of recent malware designed for the Windows OS, in windows executable format
NFR 2	ML Technique	Must Have	The program will be trained and run on a computer with at least quad-core CPU running above 2.0GHz

NFR 3	Evaluation	Must Have	Identification of a matrix for the evaluation of the research papers

3. 3. Research Question

The primary research question expected to be explored in this dissertation is which Machine Learning technique achieves the highest real-world performance for malware analysis based on static analysis. The various techniques are selected from a variety of explored research papers on the matter, selecting those with promising results. The real-world performance is measured from a combination of accuracy, false positives and false negatives that each of the techniques produces. False positives are files that get classified as malicious when in fact they are not. False-negative will be files that get classified as benign when they are malware. False negatives are relatively more serious as it constitutes to security breaches whereas false positives, although not a security breach, do lead to a lot of false alarms and possibly to operator complacency in the real world. These metrics combined with the primary measurement factor, accuracy, should provide string indications on the machine learning techniques feasibility in real-world systems. The relatively large, up to date, and validated dataset will assist us in replicating real-world testing conditions.

4. Preliminary Implementation

The construction on the dataset is well underway as we quickly identified this to be a major time-consuming task that needed to be started as soon as possible. The source that we are using to construct the malicious part of the dataset is the most recent malicious file collections from the site Virus Share [12]. The site allows its members to access repositories of live malware from its extensive collections, access to the site is invitation only and is limited to security professionals/researchers [12]. Once we had obtained the collections the process of validation began, this involves checking each file to ascertain if it is a malicious file (malware) and what type of malware it belongs to. This is done by using existing Anti-malware tools, anti-viruses, which utilize signature-based approaches to classify files as malicious or benign [6]. The process involves hashing the uploaded file and comparing that hash to that specific anti-virus's "master" malware list if a match is found then the file is flagged as malware.

To expedite this process, as multiple anti-viruses need to be used to gain an accurate confirmation on the state of the file, we used a site called VirusTotal [13]. VirusTotal is a site that provides a one-stop anti-virus file scanning capability with over 70 different anti-viruses being linked to the site [13]. In addition to providing very capable identification functionality files uploaded to VirusTotal are also shared with their anti-virus partners and premium customers, usually security professionals or organizations, thereby helping to maintain and increase the global IT security level [13]. The preliminary implementation using VirusTotal has not been very efficient as the publicly available API keys do not have a very high priority to VirusTotal resources and have hard caps on the number of files scanned in each time frame. Those hard caps are 4 files per minute, 1,000 per day, and 30,000 a month or the public access API key. We have requested a higher capacity API key but have not yet received one, once a higher capacity key is acquired it will only take a matter of days to assemble a large and detailed dataset.

Another crucial step in our dissertation process is to identify and extract the features from the files in the dataset that will be used to train the ML algorithms. The dataset will only contain Window

PE files and as noted before in section II a 1 the import functions and executable calls are both present, these will be used for feature extraction as it is a common and efficient process. So far two tools have been tested for this purpose the first being Dependency Walker a program designed to show all import functions and DLL's that a given program calls [14]. The other program that has been tested is an Online disassembler (ODA) that allows for x86 disassembly programs/files to gain a far greater understanding of the internal workings, execution cycle, and memory loads of the program/file [15].

5. Evaluation Strategy

The objective of this dissertation is to identify out of a selection of proven machine learning static malware analysis methods, the highest obtained effectiveness. To obtain these results, and ultimately ascertain an answer to the research question, evaluation of both our proposed requirements and the possible research papers techniques is required to determine to what extent our results/conclusion is accurate.

Functional Requirements Evaluation:

- FR 1 - The dataset must have two separate sections of malware and benign files for testing purposes
  - To confirm that our dataset has indeed two separate sections, all the sourced possible malicious files are and will continue to be, checked via VirusTotal to confirm that they are known examples of malware, as mentioned in section IV. The benign files within the dataset will consist of confirmed benign windows operating system files from varying versions of the OS.

- FR 2 - Feature extraction and reduction must be executed on all files from the dataset used for testing purposes
  - The machine learning algorithms require input data whereon they will build their models on, create testing data sets, and training datasets on. This input data needs to be a specific feature of the files (operation codes, static import/export functions, DLL's) as it would be computationally infeasible to use all the features. Once a feature set is selected, again due to computational infeasibility, a reduction, or narrowing, of that feature set is required to allow the machine learning algorithms applied to function relatively efficiently.

- FR 3 - Training of the various ML techniques must be performed on the training dataset
    - Machine learning stems from the idea that algorithm will learn, learning in the context of inductive reasoning, whereby one learns by observing examples and extrapolating a statistical pattern which can be used to sort new unseen data. This learning element is of critical importance to the algorithm to achieve peak performance, in our case accuracy. The number of training cycles will be influenced by the selected research papers methodologies, to achieve their results or to establish a baseline.

- FR 4 - The program must calculate and output the achieved accuracy obtained from each of the ML techniques used
    - As our primary measure of performance is the algorithms achieved accuracy of detecting malicious files from the test data, it seems only natural that such results should be easily accessible to use in our final evaluation. The other major measure of performance, resonating more with real-world performance, studied in this dissertation is the number of false negatives and false positives each machine learning algorithm produces. The reasoning behind this is laid out in section III c of this report. These results will be displayed in a confusion matrix.

Non-Functional Requirements Evaluation:

- NFR 1 - The dataset must contain at minimum 25,000 examples of recent malware designed for the Windows OS, in windows executable format
    - As discussed in section IV only the most recent collections from Virus Share are selected for the dataset, as of September 2019, and all files within these collections are in the windows executable format.

- NFR 2 - The program will be trained and run on a computer with at least quad-core CPU running above 2.0GHz
    - To limit the effects of computational infeasibility, and with the time restrictions placed on the duration of this dissertation, the use of a modern computer is crucial in order to perform the required decompress the possible malicious files before verification, possible disassembly and/or feature extraction of all the files, and training and testing of the various machine learning algorithms on the dataset. A quad-core running above 2.0GHz, while not being a true dedicated ML server, should still provide enough computational power to allow for large feature sets or larger iterations of decision trees.

- NFR 3 - Identification of a matrix for the evaluation of the research papers
    - In selecting the techniques that will be explored and tested in this dissertation we must develop a matrix to compare tested methods from various research papers to one another. This is to access the strengths of various machine learning algorithms in varying conditions such as dataset size and variation, and features extracted and used.

6. Project Management

6. 1. Methodology

The selected project management methodology to assist with the execution of the proposed dissertation plan is SCRUM. SCRUM is an agile methodology utilizing short 2-4 weeks of development periods for new software capability. The purpose behind SCRUM is to allow for more clarity and flexibility as opposed to the traditional waterfall model, as SCRUM allows for developers to change or modify existing plans/schedules depending on the current situation [16]. Due to the short development cycles SCRUM allows for changes to deadlines or deliverables per each cycle, thus allowing for greater flexibility.

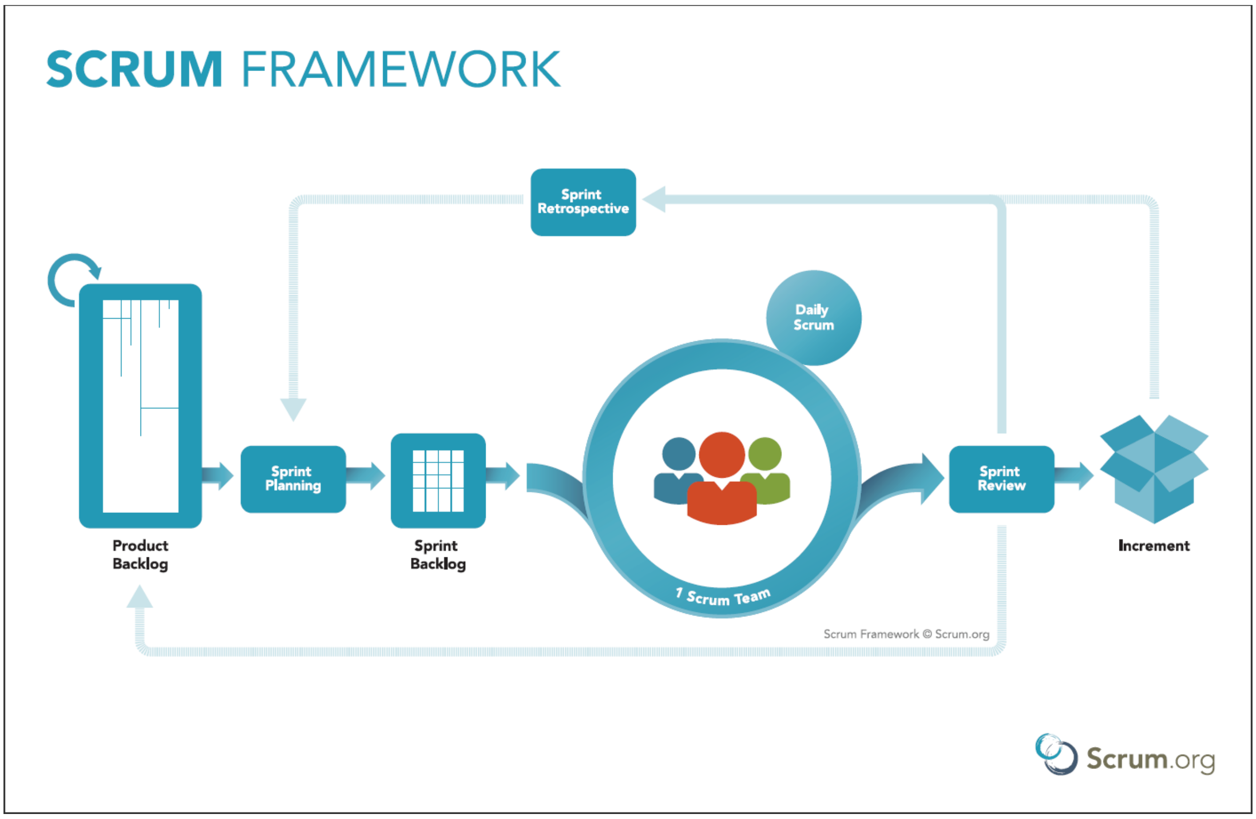
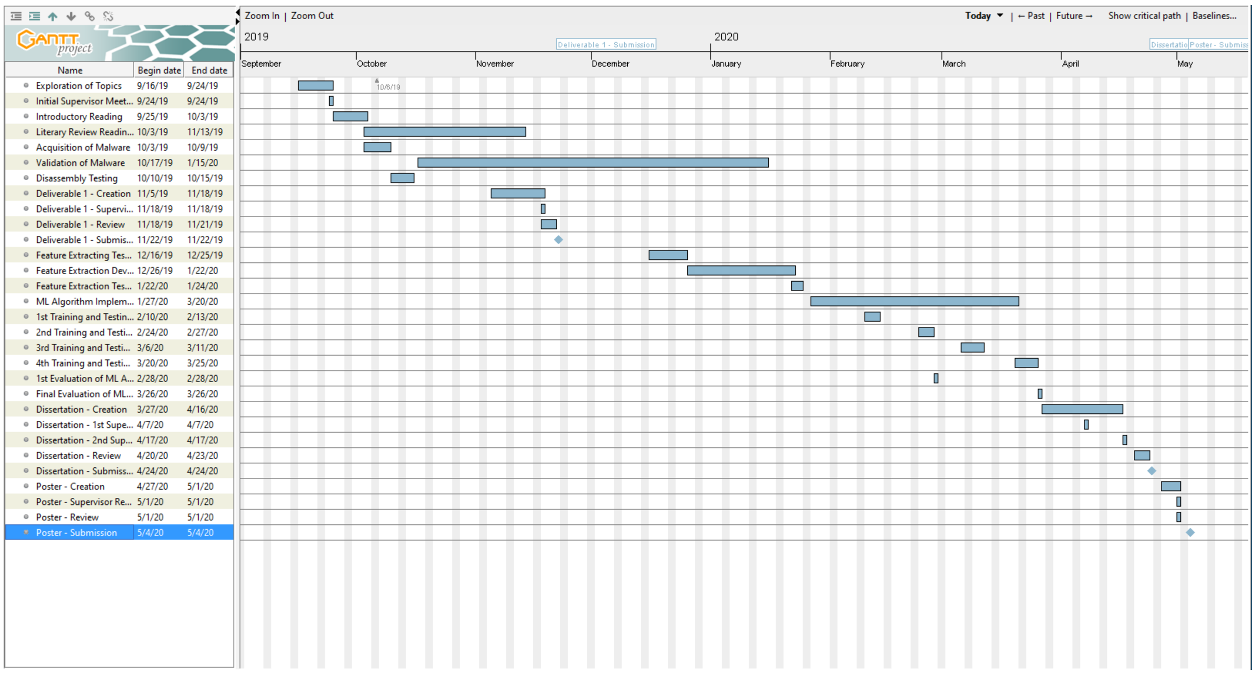


Figure 5 The SCRUM methodology framework [16]

Each development cycle in SCRUM is called a sprint, SCRUM also includes daily sprints to further assist with organizing the daily task that needs completing. As illustrated in figure 5 the product backlog is the remaining tasks/work that is required on the product, dissertation in our context. The sprint planning, where tasks are divided into the sprint's, produces the sprint backlog listing all the currently planned sprints to accomplish the goal of delivering the full product. One major change that we will implement on the SCRUM methodology is the requirement of having daily SCRUM meetings between the development team and the supervisor. This will be modified to be between the student and their supervisor, and the frequency will be reduced from daily to either weekly or bi-weekly depending on development progress. After each sprint, the sprint review is conducted whereby the current situation is inspected and from these changes can be made to either the future sprints or even to the product specifications.

6. 2. Project Plan



Name	Begin date	End date
Exploration of Topics	9/16/19	9/24/19
Initial Supervisor Meet...	9/24/19	9/24/19
Introductory Reading	9/25/19	10/3/19
Literary Review Readin...	10/3/19	11/13/19
Acquisition of Malware	10/3/19	10/9/19
Validation of Malware	10/17/19	1/15/20
Disassembly Testing	10/10/19	10/15/19
Deliverable 1 - Creation	11/5/19	11/18/19
Deliverable 1 - Supervi...	11/18/19	11/18/19
Deliverable 1 - Review	11/18/19	11/21/19
Deliverable 1 - Submis...	11/22/19	11/22/19
Feature Extracting Tes...	12/16/19	12/25/19
Feature Extraction Dev...	12/26/19	1/22/20
Feature Extraction Tes...	1/22/20	1/24/20
ML Algorithm Implem...	1/27/20	3/20/20
1st Training and Testin...	2/10/20	2/13/20
2nd Training and Testi...	2/24/20	2/27/20
3rd Training and Testi...	3/6/20	3/11/20
4th Training and Testi...	3/20/20	3/25/20
1st Evaluation of ML A...	2/28/20	2/28/20
Final Evaluation of ML...	3/26/20	3/26/20
Dissertation - Creation	3/27/20	4/16/20
Dissertation - 1st Supe...	4/7/20	4/7/20
Dissertation - 2nd Sup...	4/17/20	4/17/20
Dissertation - Review	4/20/20	4/23/20
Dissertation - Submiss...	4/24/20	4/24/20
Poster - Creation	4/27/20	5/1/20
Poster - Supervisor Re...	5/1/20	5/1/20
Poster - Review	5/1/20	5/1/20
Poster - Submission	5/4/20	5/4/20

Figure 6 Gantt chart showcasing the project plan and deliverable dates

The Gannt chart above illustrates the entire project plan from September 16th, 2019 until the final submission of the poster on May 3rd, 2020.

6. 3. Risk Management Plan

The risk management plan is an essential part of project management as understanding the risks involved with conducting this dissertation and have a plan to identify and possibly avoid/resolve if necessary is crucial. The primary goal of this dissertation is to complete all testing and deliverables by the required deadlines. This section will be divided into two parts, the first dealing with risk identification and analysis and the second with risk planning and monitoring.

6. 3. 1. Risk Identification and Analysis

Metrics associated with risk:

Probability

Very Low	Low	Moderate	High	Very High

Impact

Very Small	Small	Medium	Large	Severe

Priority

Low	Medium	High

Type

Technology	Organization	Estimation	People	Tools	Requirements

Table 2. Lists all the risks with a full analysis

ID	Risk	Type	Probability	Impact	Indication
🟥 R1	Dataset corruption with loss of all data	Technology	Low	Severe	Unresponsive storage device dataset is located on
🟨 R2	Dataset size does not grow to a sufficient size	Tools	Low	Large	Not processing enough files through Virus Total
🟥 R3	Malware leakage onto host computer from the virtual machine	Technology	High	Severe	Machine portrays un-commanded behaviors
🟥 R4	Loss of access to ML algorithms packages	Technology	Very Low	Severe	Inoperable classification methods
🟩 R5	Computational Infeasibility	Tools	High	Moderate	Extreme processing time of ML classification algorithms
🟨 R6	Drastic modifications to requirements	Requirements	Low	Large	Discovery of required additional requirements
🟨 R7	Unable to meet deliverable deadlines	Organization	Low	Severe	Incomplete/missing elements of the dissertation
🟩 R8	Incorrect estimation of time to complete feature extraction	Estimation	Medium	Large	Milestone as stated in the project plan not being met

6. 3. 2. Risk Planning and Monitoring

The risks associated with our dissertation have been identified and analyzed in the previous section. Now that the risks are known we need to develop plans to deal with the risks going forward. There are three main strategies to help accomplish this goal, avoid, minimize, and plan a contingency. Avoiding is the first step where we attempt to reduce the chance of the risk occurring, minimizing is for when avoiding fails and where we attempt to reduce the impact of the risk. Finally, the contingency is a plan for how to deal with the risk once it occurs.

Table 3 lists the planning and monitoring for each risk

ID	Avoidance	Minimization	Contingency	Monitoring
🟥 R1	Keep the data storage unit separated and only used for the dissertation.	Obtain a larger API key so that the build-up of a new dataset can proceed quickly.	Redownload malicious files and benign files and re-commence validation through Virus Total	Perform regular checks on the data storage health.
🟨 R2	Perform dataset validation continuously with an automated script	Adjust expectation accordingly to the results achieved, as the smaller dataset may not provide accurate results	Locate a new source of publicly available and validated malware dataset.	Perform regular checks on the validation process and throughput
🟥 R3	Ensure the virtual machine is configured correctly to completely self-sufficient.	Obtain a Windows OS installation image for the host machine.	Complete re-installation of host machine operating system and setting up a new virtual machine	Perform checks on background process, registers and network activity
🟥 R4	Obtain all necessary packages immediately	Look into using other ML algorithms that are accessible	Change the project scope to allow for implantation of different ML algorithms	Perform regular building and execution of the machine learning models
🟩 R5	Maintain strict feature reduction and prevent oversized trees or too many trees (for random forest)	Attempt to locate more powerful hardware, or prepare to reduce the number of features	Utilize more powerful hardware and/or reduce the number of features and/or trees being used.	Record processing times for all runs to attempt to estimate run time when upscaling complexity.
🟨 R6	Ensure the initial proposal is well-thought and confirm all proposed requirements with the supervisor.	Construct a plan of action, with your supervisor, for incorporating any changes/additional requirements.	Use the constructed plan to execute the changes/additions.	Meet with your supervisor and carefully examine the proposed requirements.
🟨 R7	Perform daily and weekly planning to stay on track	With your supervisor prepare a set of	Attempt to accomplish these alternatives	Prepare doable weekly schedules in

🟨	with the deliverable schedule.	alternative activities for the deliverables to still meet the goals of the dissertation.	activities to still meet the goals of the dissertation.	advance and have weekly meetings with your supervisor to help you stay on track
🟩 R8	Create a realistic project plan, with a Gantt chart, with as high as possible tolerances for error	Discuss with your supervisor an alternative plan prioritizing critical components	Implement this new plan	Monitor the Gantt chart to see progress and have an outlook on possible delays

6. 4. Professional, Legal, Ethical and Social Issues

This dissertation to the best of my knowledge has no professional, ethical and social issues. The legal issue is the sourcing of many malicious files that could be used for their intended purpose, as certain malware can be utilized in cyber-crime/warfare [6]. To ensure this does not occur the data storage device on which the dataset is located will be stored securely and access to that device will be limited to the student and their supervisor.

7. Bibliography

[1] Kaspersky, "Digital Dangerscape: Kaspersky Lab Spotlights Cybersecurity Trends in the Middle East, Turkey, and Africa," 30 April 2019. [Online]. Available: https://me-en.kaspersky.com/about/press-releases/2019_digital-dangerscape-kaspersky-lab-spotlights-cybersecurity-trends-in-the-middle-east-turkey-and-africa. [Accessed November 2019].

[2] L. Liu, W. Bao-sheng, Y. Bo and Z. Qiu-xi, "Automatic malware classification and new malware detection," Frontiers of Information Technology & Electronic Engineering, Changsha, 2017.

[3] G. R¨atsch, "A Brief Introduction into Machine Learning," Friedrich Miescher Laboratory of the Max Planck Society, T¨ubingen, 2004.

[4] H. S. Joshua Saxe, "Chapter 6: Understanding Machine Learning-Based Malware Detectors," in *Malware Data Science*, San Francisco, no starch press, 2018, pp. 89-117.

[5] K. Kris and M. Chad, "Practical Malware Analysis," Black Hat, 2014.

[6] S. Michael and H. Andrew, Practical Malware Analysis, San Fransisco: No Starch Press, 2012.

[7] S. S. Shai and B. D. Shai, Understanding Machine Learning:, New York: Cambridge University Press, 2014.

[8] K. S. Akash and J. Aruna, "Integrated Malware Analysis Using Machine Learning," in *International Conference on Telecommunication and Networks*, Noida, 2017.

[9] Z. Jingling, Z. Suoxing, L. Gohan and B. Cui, "Malware Detection Using Machine Learning Based," in *10.1109/ICCCN.2018.8487459*, 2018.

[10] I. Muhammad, H. D. Muhammad and I. Maliha, "Static and Dynamic Malware Analysis," in *IBCAST*, Islamabad, 2019.

[11] B. Usukhbayar, J. Nyamjav and H. Shi-Jinn, "A Static Malware Detection System Using Data Mining Methods," National University of Mongolia, 2013.

[12] "VirusShare.com," [Online]. Available: https://virusshare.com/. [Accessed 22 November 2019].

[13] "Virus Total," [Online]. Available: https://www.virustotal.com/gui/home. [Accessed 22 November 2019].

[14] "Dependency Walker 2.2," [Online]. Available: http://www.dependencywalker.com/. [Accessed 22 November 2019].

[15] "ODA," [Online]. Available: https://onlinedisassembler.com/odaweb/. [Accessed 22 November 2019].

[16] "Scrum.org," [Online]. Available: https://www.scrum.org/. [Accessed 22 November 2019].