TECHNICAL COMPUTING

DISSERTATION RESEARCH PROJECT PROPOSAL

Investigating Machine Learning Models in the Classification of Potentially Malicious Code

Ben Eriksson

S1709402

Dr. Adam Gorine

Supervisor

University of Gloucestershire

School of Computing and Technology

2019

Declaration

The following research project and associated proposal is the sole production of myself.

Any documentation associated that is not of my origin, has been correctly cited and credited to the original Author(s).

Despite the investigation of potentially malicious software, this research project does not infringe upon the ethical research principals outlined in the relevant University of Gloucestershires' Handbook.

Any potentially malicious code has been legally obtained from the gratuity of Industry experts, whom produce such content for an equal goal of investigate learning within the community.

Extensive efforts have been made to ensure the collected data remains isolated from any other Information Technology devices that aren't within scope.

Acknowledgements

I would like to thank my family for their continued support throughout my studies, as well as Dr. Gorine for his council throughout this research project. Finally, the liberal individuals within the Industry who share their datasets for communal use.

Abstract

The prevalent risk of malicious infection is an ever-increasing threat to the availability of World-wide Information Technology infrastructure, whom form as pillars to day-to-day society as we know it.

Presenting in many forms, Malware variants have a variety of common objectives from an attacker's perspective. One category known as Ransomware, is a variant of malicious code whom prohibits a User from accessing a system and/or its data until a ransom is paid - a modern twist on the traditional extortion schemes used by criminal syndicates.

This is an especially noticeable limitation in current methodologies for malignant detection, using signature-based detection, which uses pre-determined rule sets to identify code as malignant or not.

The following research project investigates and experiments with applying machine-learning models to be able to score the maliciousness of code within datasets based upon their heuristics, where the models have no prior-knowledge of the file presented.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1. Overview

Whilst the idea of combining Computing resources and technical knowledge to solve both human social and scientific issues isn't a new concept, the introduction of Machine Learning and the use of Artificial Intelligence to solve problems of a grandeur scale is at an unprecedented level.

The applicability of AI can be seen as implemented at all scales of suitability. Companies such as Facebook and Twitter use Artificial Intelligence to learn more about their Users at an individual level, tailoring the content that is delivered that is relevant to their interests. Or from a business perspective, deliver customised advertising that will bring a much higher rate of ad-revenue because of the Users interests.

Considering, this use of AI is arguably trivial when comparing other implementations of the ground-breaking technology. Ranging from self-driving Cars to progressing the very forefront of Science with Quantum Computing and bio-medical prediction. (Chui et al., 2019)

This research project will investigate and apply various machine-learning algorithms and models against a collated data-set of Portable Executable's, henceforth referenced as **PE** files for identification of their behaviour and intent to a host Operating System

Microsoft Windows' PE file is a standard of file-formatting, insisting the structure of various objects across the family of Windows Operating System releases.

This PE structure is a prevalent rule-set that can be found across many file-types; ranging from executable files for applications, to text-font files and object files - all essential for functionality of the Operating System.

The Windows PE File comprises of eleven sections. Whilst application Authors can populate these sections as desired, each section will contain metadata, informing the Operating System of how the file should be processed within the application (e.g. defining it as a executable file rather than an object file) (Windows Developer Center, 2019)

Whilst the Windows Operating System family has evolved, the PE file has continued with little variation.

Microsoft's Windows Operating System has done a remarkable job at maintaining compatibility with previous generations of the Windows Operating System. The **MS-DOS** header within a PE file is an excellent example of these efforts of backwards compatibility. This header enables the Operating System to process the file and check for compatibility across the whole file. Where there is no compatibility, rather than just crashing as the first editions of MS-DOS used too, the Operating System will now just inform the User that there is no current compatibility.

There can be optional parameters, which are often populated during the development process of the software attributed to the PE file. These are not essential for a PE file, for example, an icon that is a graphical way of identifying the application to the user. This is illustrated in Figure 1.

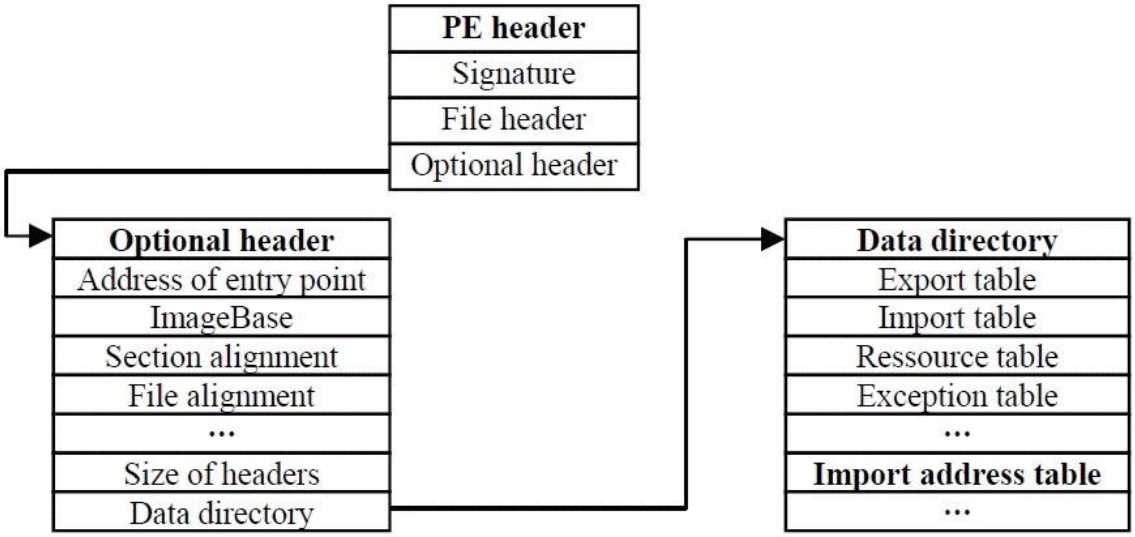


Figure 1: The Portable Executable (PE) structure with optional parameters (Belaoued, M., Mazouzi, S., 2016)

1.2. Problem Statement

This research project will highlight the limitations of current mitigations in place to combat the threat of Malware. Signature-based identification is the first-line of defence against the war on Malware.

Signature-based protection solely relies upon the malignant intent of a sample file already being identified. This could be through various channels such as manual inspection from an Analyst investigating the files behaviour to the pre-cursor knowledge of the behaviours the file exhibits, i.e. the file has performed malicious actions on a previous device.

Due to the biological characteristics of malicious code, the identification of brand-new or “zero-day” variants are an on-going "Cat and Mouse" game between both Malware Analyst and Malicious threat actor. These threat actors can create a substrate of a pre-existing Malware variant in very little time - in comparison to the resources needed for a Malware Analyst to determine the heuristics and behaviours of such generated code.

As Signature-based detection effectively compares a file’s “fingerprint” or signature against a predetermined list. These signatures are created from a bit-to-bit accuracy, where any deviation of for example, a character will attribute the file with a new signature. This is extremely problematic in combatting Malware, as although the malicious intent of the file remains the same, due to fact there is a slight deviation, an anti-virus engine using Signature-based detection will treat it as a safe file.

In the context of Artificial Intelligence, classification is a quintessential concept. From detecting animals to the likes of human expression. The proposed research project explores pre-existing classification concepts to create an Artificial Intelligence model capable of detecting new substrates of malicious code “on the fly” as opposed to traditional, current capabilities.

1.3. Research Questions

- What attributes can be found within malicious code that can be used to identify its intent?
- Do variants of malicious code have similar attributes, and can these common characteristics be applied to the same machine-learning model?
- Can this identification be accurate enough for true-positive classification, in comparison to alternative methods such as signature-based detection?
- What signatures and characteristics do current Anti-Virus engines use to identify and classify variants of malicious code?
- Can any machine-learning model be used to combat new substrates of malicious code from procreating and propagating before infection?

1.4. Research Objectives

To achieve the aim laid out for this research project, the following objectives will be investigated:

Table 1: Overview of the Research Objectives

1	To investigate the characteristics of malicious code and how these attributes can be used by a Machine Learning model for identification.
2	To assess pre-existing Machine Learning Models and Malware identification techniques and understand their effectiveness in identifying malicious code.
3	To evaluate the reliability and accuracy of a variety of developed Machine Learning Models and their performance in detecting malicious heuristics of code, identifying and preventing further infection.

1.5. Scope

With Microsoft's Windows Operating System holding a majority **86%** market share (Netmarketshare., 2019) of the consumer-level market, this research project only investigates Malware samples that are compatible with the Microsoft Windows NT Operating System (Windows 10, Windows 7) as Malware poses a potential fatal risk to Consumers especially - whom may not have the expertise to implement an effective disaster-recovery plan.

Additionally, due to the very nature of Malware, it is sensible to take a precautionary view when classifying Malware. This consideration is a prevalent thought throughout the basis of this research project. It is extremely risky to assume the classification is always correct – despite the amount of training and accuracy scores, Malicious files may be mis-categorised by the model and may lead to infection.

In essence, submitted samples will be point-scored based upon the presence of malicious features– rather than categorising them definitively via the various scoring points across the random-forest.

Finally, the state-changes exhibited from Malware when the program has executed will not be investigated and is out-of-scope for this project.

1.6. Ethical Issues

The destructive capabilities of Malware are of grave concern to the integrity of this research project. Equalising Malware to a biological virus whom have various means to replicate itself and infect other hosts, Malware too has various means to infect other devices. Although a means of infection isn't by a characteristic of being airborne like a traditional virus, Malware depending upon its variant can infect other devices over a Network - especially in a Local Area Network (LAN) environment.

Although published datasets containing malware is originated from genuine Malware samples, they are un-weaponized, meaning the malicious elements to the sample have been removed. However, still contains the characteristics of the variant it is produced from - making it a perfect example for entry-level Malware Analysts to investigate without much risk.

1.7. Conclusion

The following research project illustrates the threat that Malware poses, as well as the daily battle imposed on Analysts from the Authors of Malware. Whilst Analysts have introduced methods to combat this, for example, signature-based analysis, whom is an effective approach at prevent infection against already classified malware. The current common implementations today struggle to identify the heuristics of never seen before samples.

Researchers have investigated into using the common use-cases of Artificial Intelligence such as classifiers. Traditional implementations involve detecting between different animals and objects. This research project uses the idea of these traditional aims, but rather then detecting different animals, will be investigating the features that can be used to detect the maliciousness of software on computing Operating Systems.

Data used to train the model as well as compare results with will be collected through various means, one being a mixture of high interaction Honeypots consisting of vulnerable services whom are frequently probed and attacked by threat actors such as Remote Desktop and Samba, a file sharing service for Windows. Finally, published datasets from other researchers whose samples are collated through very similar interactions of vulnerable services.

2. Literature Review

Malware analysis is a complex and ever-changing topic, containing many features to train a machine learning model upon. As (Schultz et al., 2000) proposes, “Eight to ten malicious programs are created every day, and most cannot be accurately detected until signatures have been generated for them” indicating a huge necessity for real-time analysis to be made to prevent infection on host devices.

This is supported by Symantec, an industry revolutioniser in commercial Anti-Virus engines, whom report of “[an increase] in the first six months of 2017, Symantec blocked just over 319,000 ransomware infections” (Symantec, 2017) evidently, Malware, especially ransomware is on the dramatic increase. Case studies such as WannaCry “with infections recorded across 150 countries globally” (Nominet, 2017) lightly demonstrate the global and non-target-specific gripe the threat poses.

Thankfully, there has been some continued investigation into circumventing the limitations that signature-based detection retains. For example, (Schultz et al., 2000) created a framework whom "automatically extracted a binary profile from each example in [their] dataset" using "properties … such as byte sequences". This is an effective alternative to other frameworks proposed by researchers such as (Tesauro, Kephart and Sorkin., 1996) who only achieved a small detection rate due to only investigating PC Boot-sector viruses, of which "PC boot sectors are 512 bytes long". With this limited amount of code, there is a very minimal expectation of the features that can be extracted.

When larger datasets containing a lot of features are created, such as that in (Mohaisen, Alrawi and Mohaisen, 2015) Consisting of a much larger scale of "115,157 malware samples".

Whilst the study "used only a total of 65 features for classification and clustering", their best performing algorithm achieved a 85% accuracy rate. However, it should be noted that their performance scale was calculated on both the highest yet most time-efficient algorithm. Other explored frameworks may have achieved a higher accuracy score, however at a much higher systemresource cost.

This is a noticeable improvement over alternatively suggested frameworks and algorithms such as those of (Bayer et al, 2009), whom suggest that "aggressive approximate clustering techniques may need to be employed [with much larger datasets]" resulting in a loss of accuracy due to generalization. The dramatic decrease in accuracy as cluster – hence dataset sizes increases is shown in Figure 2

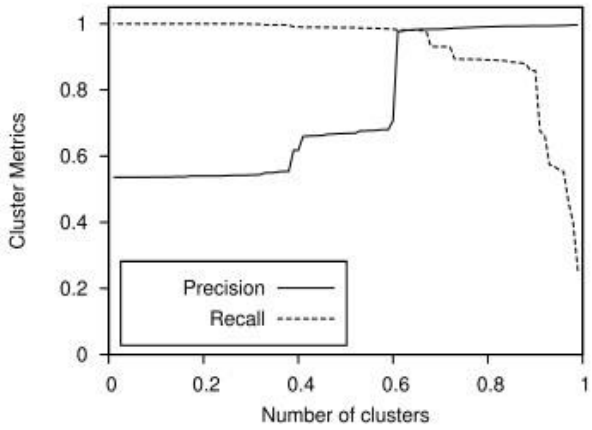


Figure 2: The dramatic decrease of accuracy Vs. precision in increase of clustering methods (Bayer et al, 2009)

Arguably, system-resource usage is an expendable commodity in order to achieve a higher classification detection rate, especially in a corporate environment. Although, a much larger corporate environment such as an Enterprise who faces a vast quantity of advanced persistent threats from malicious actors daily.

Binary classification appears to be a popular and successful approach to solving the issue of Malware classification. Binary classification, within the context of extracting information of Malware involves the "process of classifying given document/account on the basis of predefined classes" (Kumari and Kr., 2017).

To extend the context of this, examples of binary classification is a fundamental basis in Malware analysis. Malware Analysts do this manually when investigating individual cases, and algorithms on a much-larger scale when classifying. Extracting behavioural patterns from object files such as **.dll**'s, or in the case of (Dhanyasree, Krishnan and Ambikadevi Amma, 2019) both non-verbal and verbal communication classes can be collated together to classify the legitimacy of a social-media user profile.

Combining different sets of classes that are extracted from the characteristics and information from the object files such as **.dll**'s and **.exe**'s will be a fundamental methodology for classification throughout this research project.

Alas, there are very noticeable constraints of statistical analysis of Malware. One simple but fatal example of this is code obfuscation. As (Moser, Kruegel and Kirda, 2007) suggests,

"The values of the arrays […] are crafted such that after [a] for loop, all bits of the first group have the correct, final value, while those of the second group depend on the random input"

Values such as integers for important functions can be populated based upon random number generation after execution. Simply, the state in which the malware initially presents itself may end up acting differently upon execution. This is a very frequent technique in which sophisticated Malware Authors use to bypass anti-virus engines.

To highlight the impact of this, a file that uses code-obfuscation may be classed as non-malignant. However, upon execution, if the file was sent through the classification model again, would be classed as malignant.

Another common – albeit sophisticated approach to avoiding the anti-virus signature-based detection is the use of mutation, known as polymorphic code. As (Selamat, Mohd Ali and Abu Othman, 2016) proposes, Malware can procreate with similar features and intentions, however contain enough differences within their code to appear as a completely new file – hence their similarities to the common biological virus.

Malware common achieves a mutation from a mutagen. This engine utilizes cryptography once a system is infected, new code is generated ready for further infection to another host. As this new code is randomly generated, the detected signature will not be akin to that of the code that caused the initial infection.

However, (Bazrafshan et al., 2013) introduces to the fact that although malicious code may be randomly generated, it does not avoid anti-virus detection by default. They propose that a common feature of Malicious code is random Variable name generation. They state that "[although] Changing the variable names may confuse human [this] has almost no effect for automated detection techniques"

The research published by (Selamat, Mohd Ali and Abu Othman, 2016) investigated into the identification of polymorphic malware. Using a framework such as that of Figure 3.

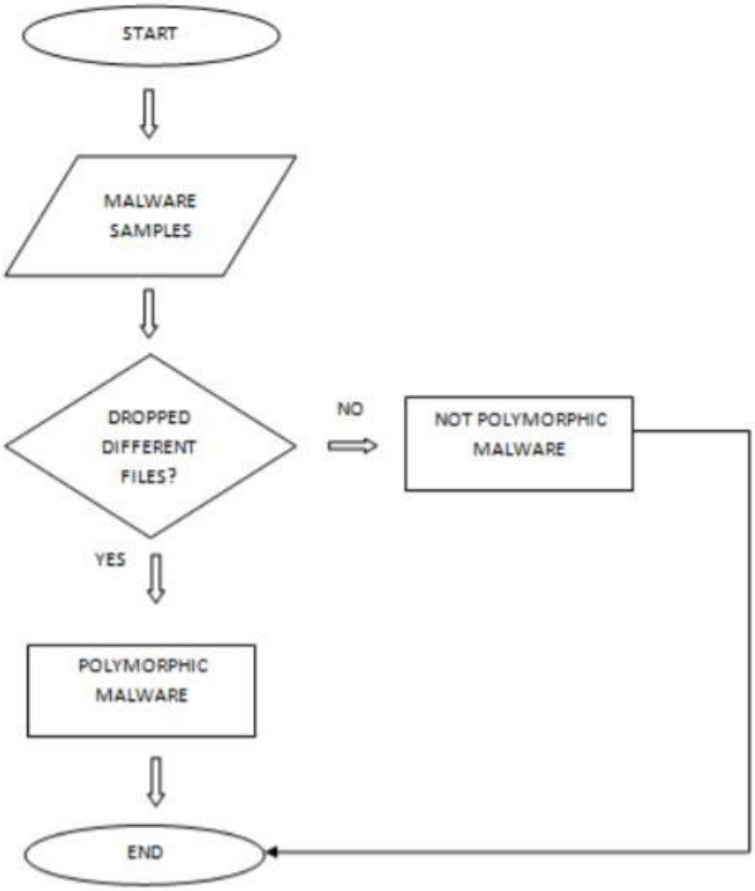


Figure 3: Flowchart illustrating the proposed framework for the identification of polymorphic Malware (Selamat, Mohd Ali and Abu Othman, 2016)

3. Methodology

3.1. Overview

When trying to solve a problem using Classification, there are two significant algorithmic approaches that are considered, each with beneficiaries and adversaries that may also go hand-in-hand.

For instance, the approach of using a decision tree-based model for identifying malicious code seems appealing – resembling a flowchart.

As Figure 4 below shows, a file can be inputted into the algorithm, where it is analysed for certain attributes that have been pre-determined as malignant or not.

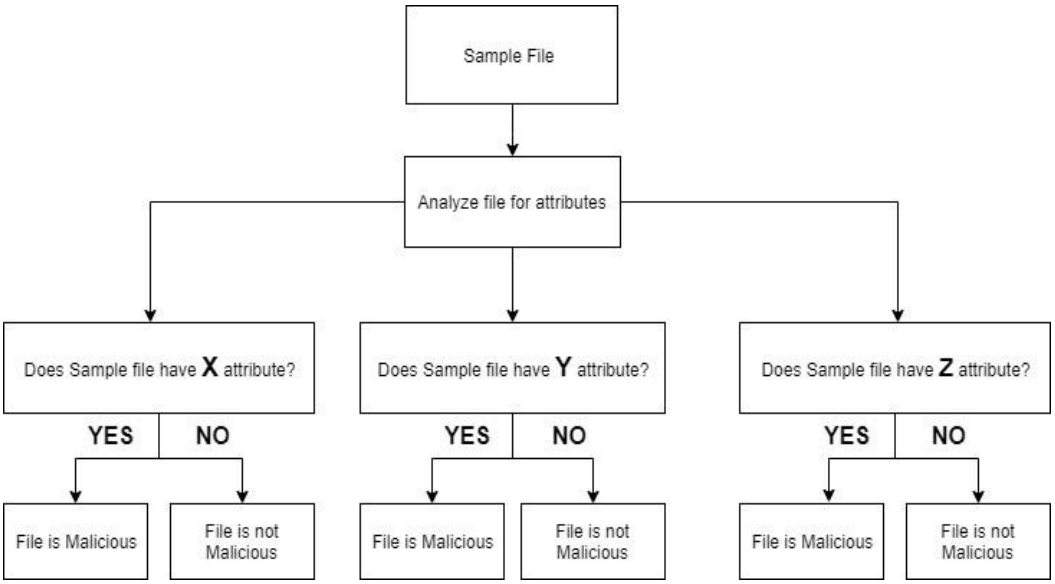


Figure 4: Illustrating the use-case of a decision tree-based machine learning model

Whilst this classification has many benefits, one being that it can deal with large datasets quickly as the algorithm essentially follows a checklist. Another being from an Author point of view, it is rather quick and easy to develop and train…Any changes made can be replicated across the model almost instantly. This would be extremely beneficial if the attributes of the sample file will change frequently, a few rule-sets will be changed and the model can adapt to the new parameters well. This type of model has no memory of previous-actions, this is a huge disadvantage for this case as it cannot self-improve.

Because of these features, the model will have to be taught the new attributes it is to search for, requiring a lot of human-intervention.

Additionally, not only the complexity of PE Files, but the fact that the PE file is a standardised format across all objects on a Windows Operation System, this classification model may prove to be very inaccurate, where accuracy is one of the most important influences in this project.

Figures 5 and 6 highlights that both “malicious” and “clean” files contain similar attributes due to the nature of the PE File structure. Specifically, creating a Decision Tree model to solely classify code as malicious or not - based upon the presence of a certain attribute such as “Imported Function” will

henceforth, classify all PE files as “malicious”. This is due to the standardised structure of a PE file whom will contain such “Imported Function” attribute, regardless of its intent.

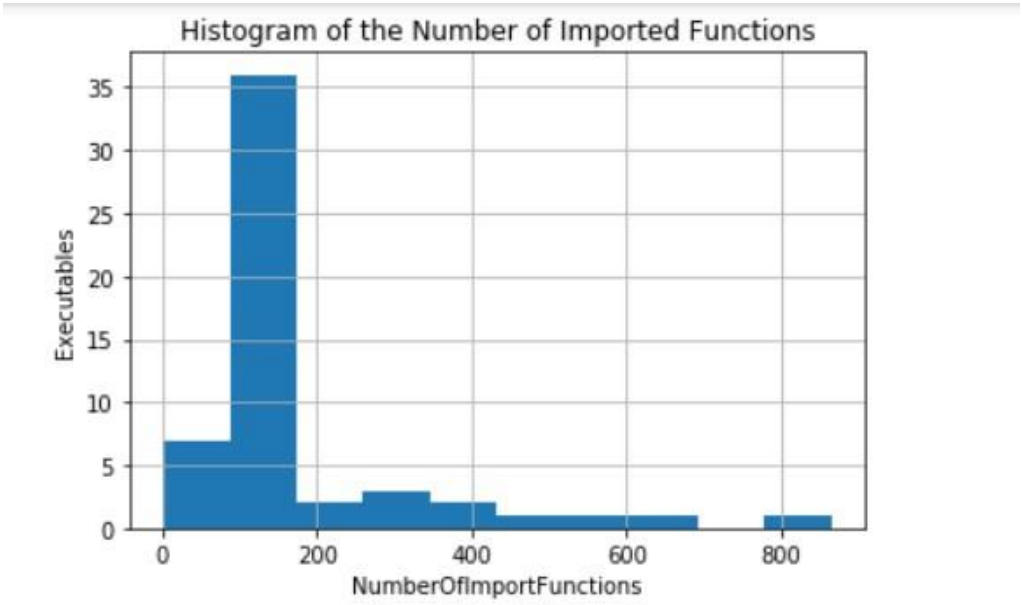


Figure 5: A Histogram illustrating an example function “clean” files contain

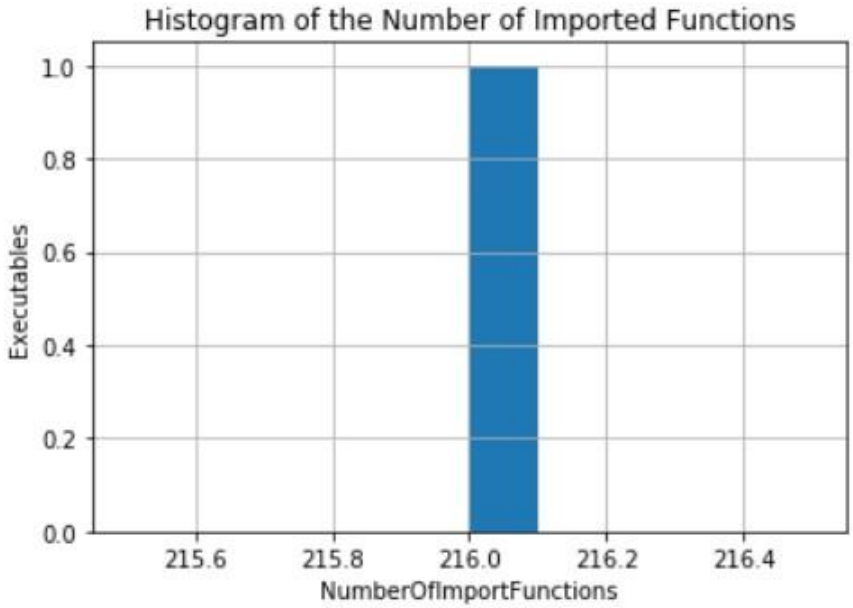


Figure 6: A Histogram illustrating an example of the same functions that a "malicious" file contains

As such, a Random Forest based algorithm is explored within this project. A Random Forest algorithm can be a self-teaching algorithm, in which previous performance indicators can be stored and revisited by the model to improve accuracy by itself, such as malicious intent, and no manual input is required.

This could lead to possible avenue where the model is capable of detecting maliciousness in never-seen before files based upon its own previous performance.

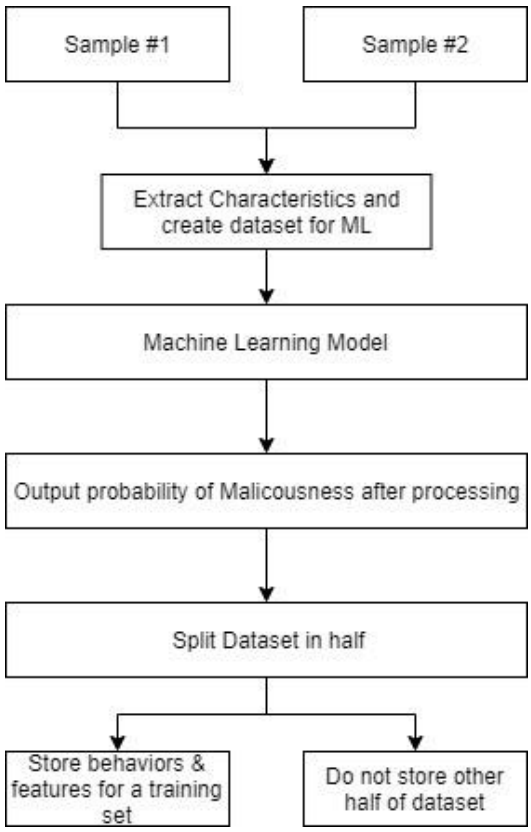


Figure 7: An implementation of Random-Forest to create a training sample set

A random-forest implementation has seen to have a high accuracy rate across few academics. For example, as seen in Figure 8.

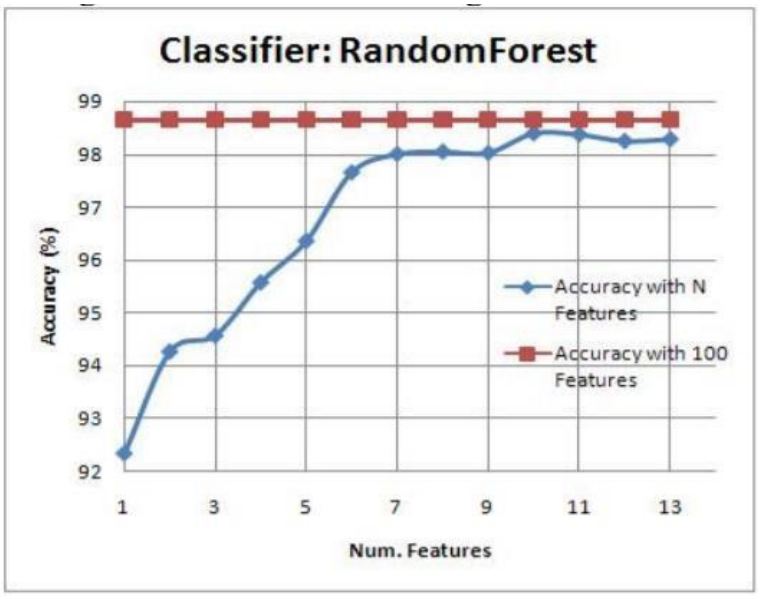


Figure 8: Line Graph of the accuracy achieved by a random-forest-based framework (K. Raman, 2012)

Figures 9 and 10 are an illustration of the proposed experimental algorithm, where we can see how the training set is devised from the collated samples and sent through the model multiple times.

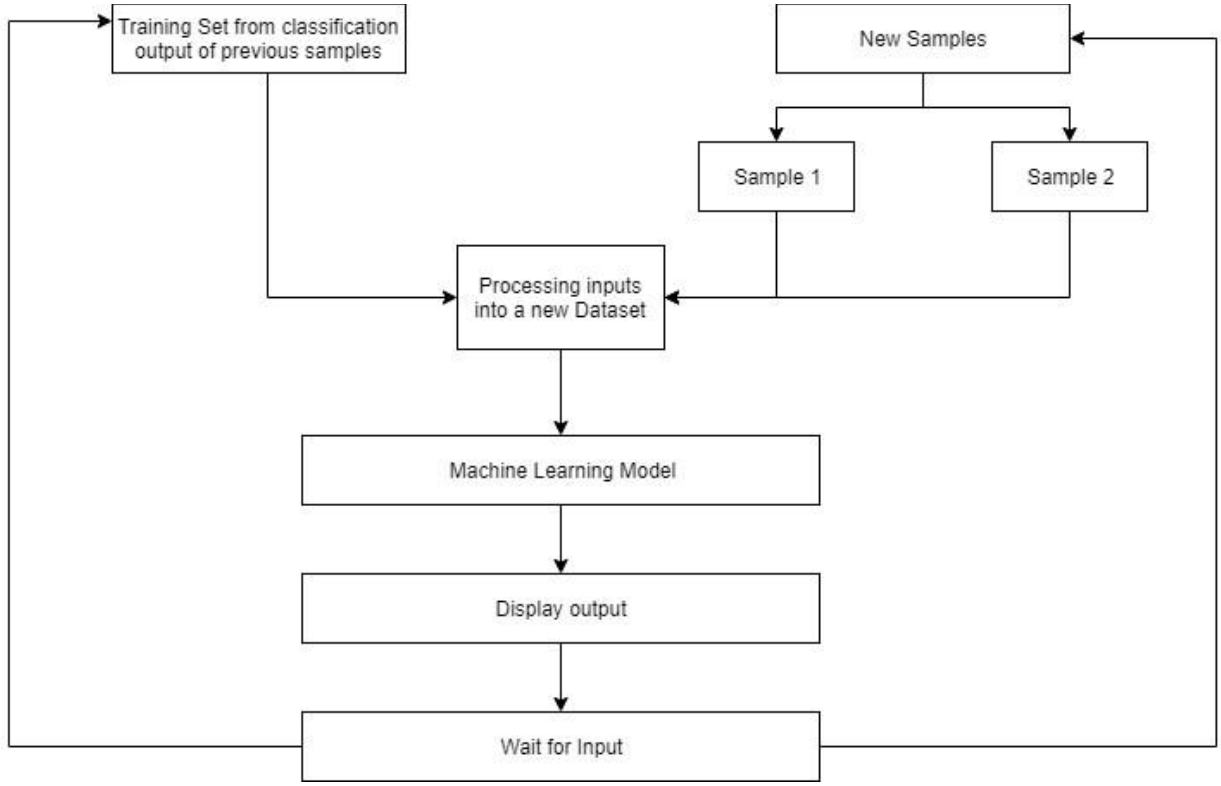


Figure 9: An illustration of how the model combines both new samples and previous performance from training set for continuous learning

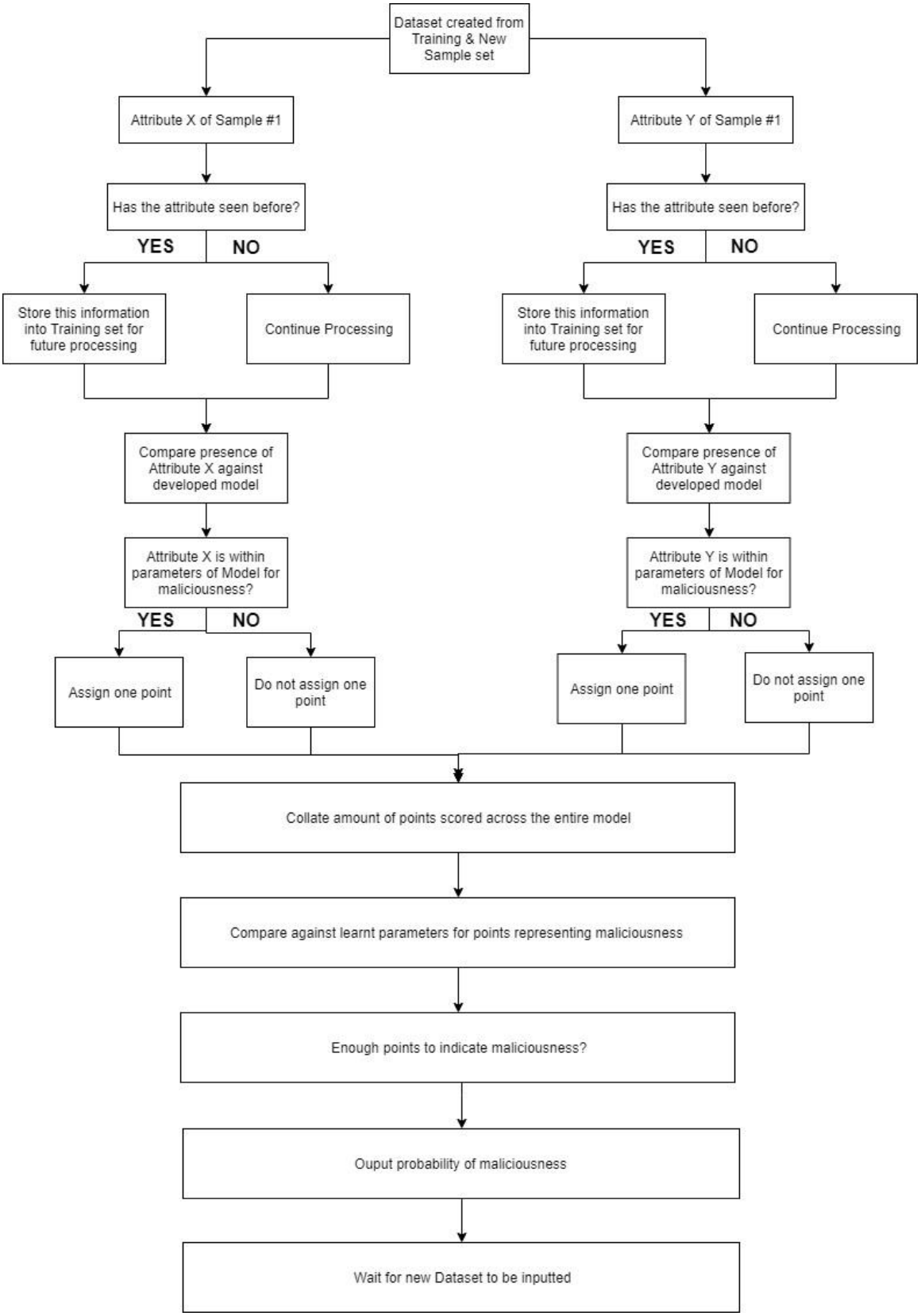


Figure 10: Illustrating two decision trees within the random-forest algorithm uses to identify maliciousness of samples

3.2. Dataset Sampling

Obtaining a dataset of PE files to use as a **clean** baseline is imperative for this research project. We assume a dataset as **clean** in the parameters of the samples are collated from a fresh-install of an Operating System – in this instance, Windows 7 Professional 32 bit.

Within this context, the source of the dataset is a "Virtual-Machine" using the commercial-platform "VMware Workstation Pro 15" on-top of a Windows 10 Operating System henceforth referred to as the Hypervisor.

A fresh-install of the Operating System onto the Host entails installing the Operating System using a digital-replica of the traditional installation disk, with no additions to the post-installation state. It is essential that no additional software is installed on this Host, as such action may render the dataset as tainted.

Figure 11: System Information for "clean" dataset source on Windows 7 Professional (32bit)

The efforts to ensure that the Host remains **clean** is continued throughout collection of data by ensuring the Host remain disconnected from the Internet. An expected procedure of a fresh-install of the Windows Operating System is to "call home" and check for security updates. The updates that are downloaded are varied based upon numerous factors, such as:

- Service Pack version
- License type e.g. Windows Home, Windows Professional
- 32bit or 64bit

This is problematic as these updates introduce a large quantity of variables when collecting the dataset. For this research project, we want to keep the baseline as generic - hence applicable as possible.

The samples collected are from “Windows” as this is the Operating System’s Users-abstraction of the Kernel-layer. The Operating system at “Levels 2” and “Level 3” is merely an abstraction of the Level 0 – the Kernel, who is not directly accessed by the User – but through the Operating System itself, as visualised in Figure 12.

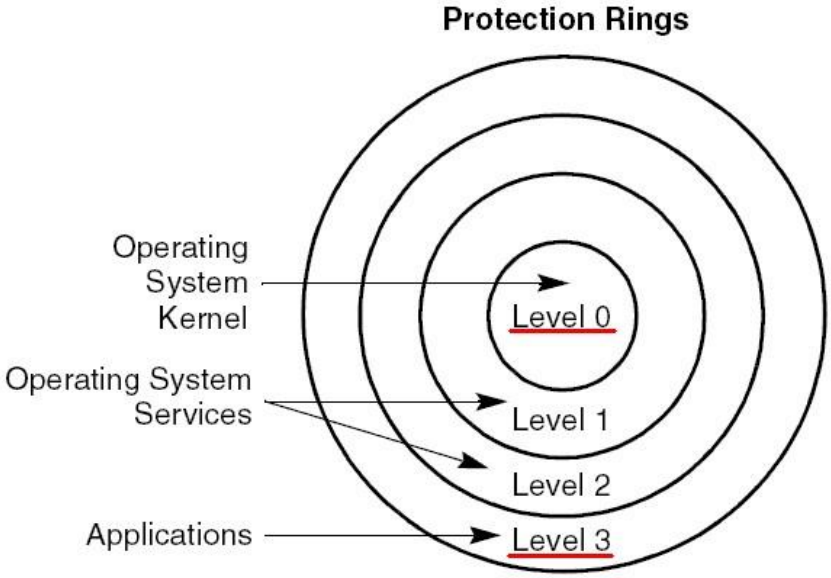


Figure 12: Operating System "Protection Rings" Diagram (Montana State University, 2005)

In the context of a post-install state, the Host contains an approximate *11,005* **.DLL** object files. The actual result may vary, as some object files are not visible or accessible to the User by the Operating System. These will be used as the baseline for the Random Forest model.

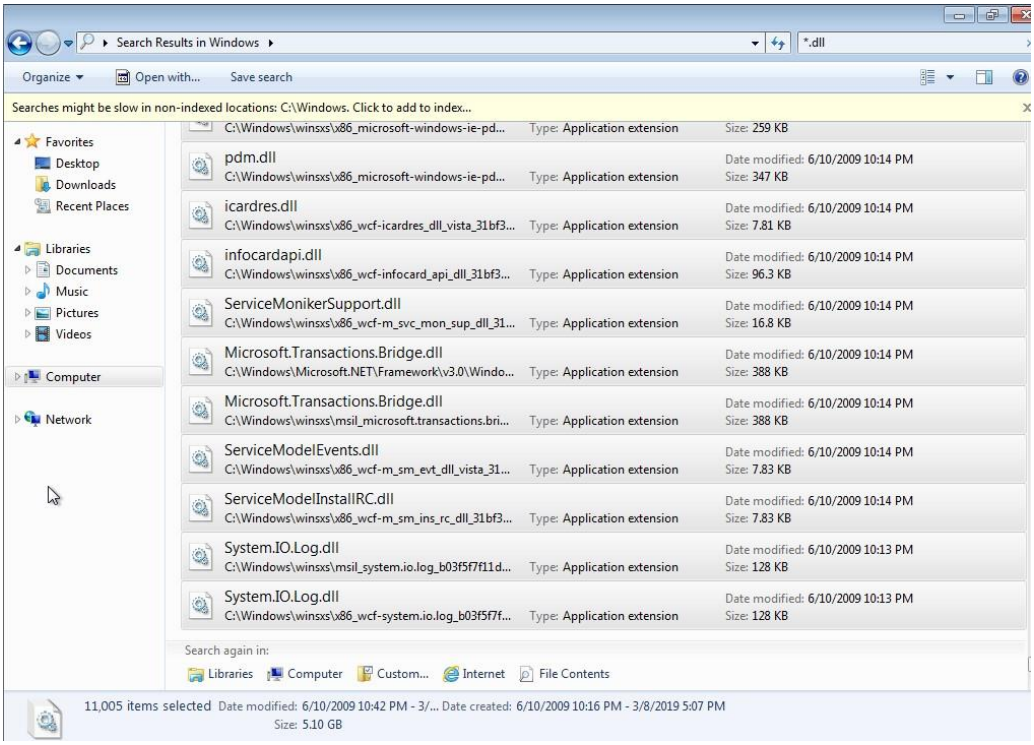


Figure 13: Graphical Representation of the "clean" dataset taken from a Windows 7 in Post-Installation state

After considering other research projects whom have explored into using machine learning to combat Malware, this is considered a suitable dataset sample for clean files. However, whilst “malicious” files contain similar functions, the amounts and presence of these can be varied in comparison to system files.

The *HoneyBow* toolkit, a high-level interaction Honeypot developed by (Zhuge et al., 2006) "integrates three malware collection tools ... [rather than emulating, uses] true vulnerable services as victims to lure malware infections" this method allows real-case and potentially never seen before zero-days to be captured for analysis, whom would truly bypass the traditional signaturebased malware.

Figures 13 and 14 illustrate the key conceptual differences between high and low level interaction Honeypots.

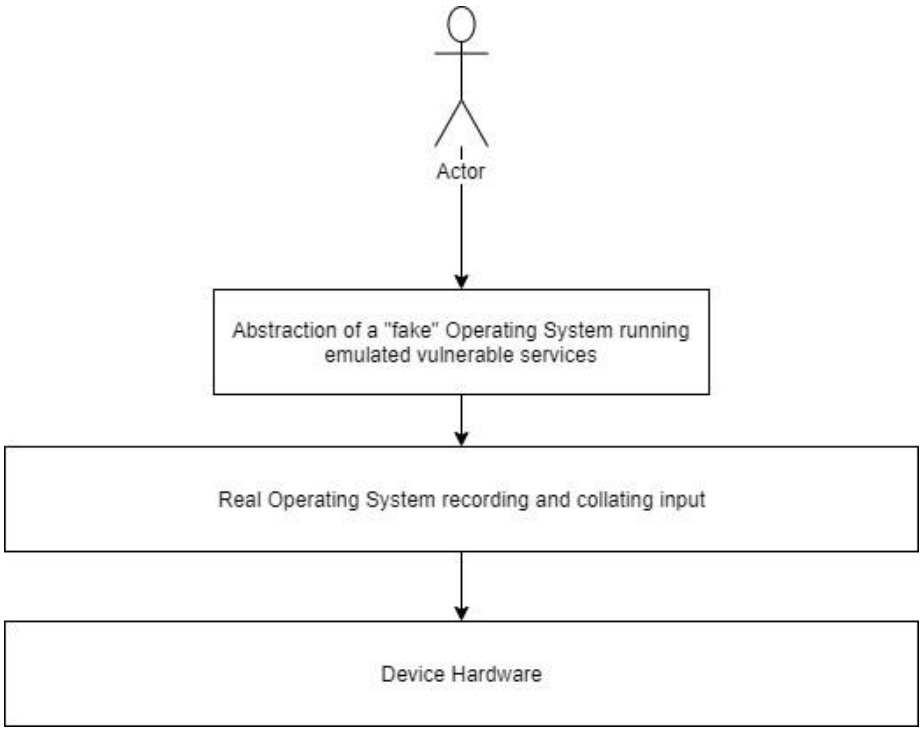


Figure 14: Visualisation of a low-level interaction Honeypot

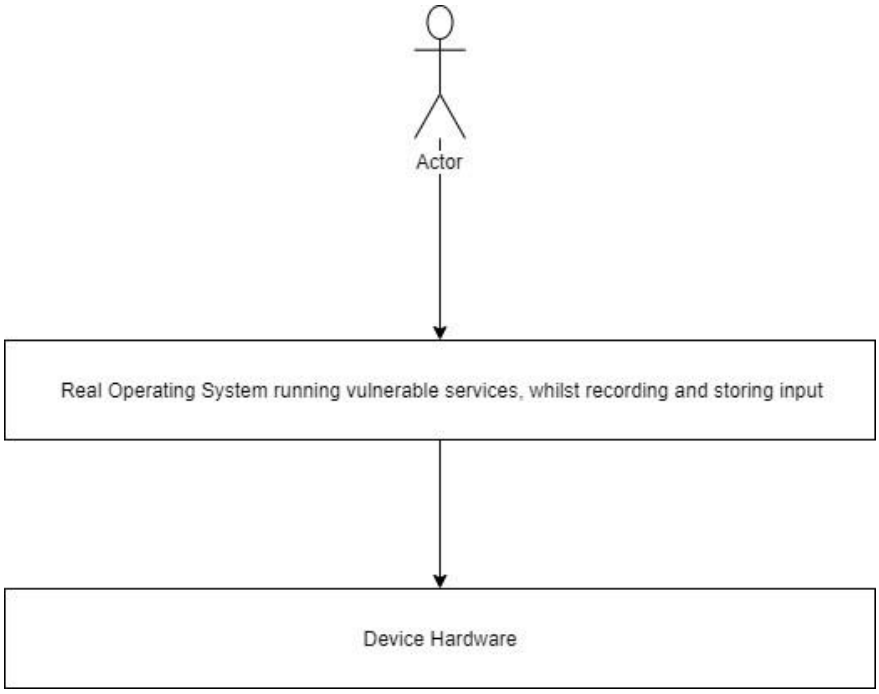


Figure 15: Visualisation of a high-level interaction Honeypot

Combining the use of honeypots and data sample sets from security researchers such as “The Zoo” project will provide a sufficiently sized dataset of both “clean” and potentially “malicious” files.

4. References

Alam, S., Horspool, R., Traore, I. and Sogukpinar, I. (2015). A framework for metamorphic malware analysis and real-time detection. *Computers & Security*, 48, pp.212-233. Retrieved from: https://www.sciencedirect.com/science/article/pii/S0167404814001576.

Bayer, U., Comparetti, P. M., Hlauschek, C., Kruegel, C., & Kirda, E. (2009). Scalable, Behavior-Based Malware Clustering. P 11-12

Bazrafshan, Z., Hashemi, H., Fard, S. and Hamzeh, A. (2013). A Survey on Heuristic Malware Detection Techniques. *Conference on Information and Knowledge Technology*, 5, pp.113-116. Retrieved from: https://ieeexplore.ieee.org/abstract/document/6620049.

Belaoued, M. and Mazouzi, S. (2016). A Chi-Square-Based Decision for Real-Time Malware Detection Using PE-File Features. *Journal of Information Processing Systems*. Retrieved from: http://jips.jatsxml.org/Article/12/4/644.

Chui, M., Manyika, J., Miremadi, M., Heneke, N., Chung, R., Nel, P. and Malhotra, S. (2019). *NOTES FROM THE AI FRONTIER INSIGHTS FROM HUNDREDS OF USE CASES*. McKinsey Global Institute, pp.3-20. Retrieved from: https://www.mckinsey.com/~/media/mckinsey/featured%20insights/artificial%20intelligence/notes%20from%20the%20ai%20frontier%20applications%20and%20value%20of%20deep%20learning/notes-from-the-ai-frontier-insights-from-hundreds-of-use-cases-discussionpaper.ashx.

Dhanyasree, P., Krishnan, S. and Ambikadevi Amma, T. (2019). Deception Detection in Social Media through Combined Verbal and Non-Verbal Behavior. *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(4), p.446. Retrieved from: https://pdfs.semanticscholar.org/6488/57a8208b1af246f17ad08894237aa4a00c3c.pdf.

Dolan-Gavitt, B., Hodosh, J., Hulin, P., Leek, T., & Whelan, R. (2015). Repeatable reverse engineering with PANDA. *ACM International Conference Proceeding Series*, *08-Decembe*. https://doi.org/10.1145/2843859.2843867

Fattori, A., Lanzi, A., Balzarotti, D. and Kirda, E. (2015). Hypervisor-based malware protection with AccessMiner. *Computers & Security*, 52, pp.33-50. Retrieved from: https://www.sciencedirect.com/science/article/pii/S0167404815000395.

Gavrilut, D., Cimpoesu, M., Anton, D. and Ciortuz, L. (2009). Malware detection using machine learning. *2009 International Multiconference on Computer Science and Information Technology*.

Islam, R., Tian, R., Batten, L. M., & Versteeg, S. (2012). *Classification of malware based on integrated static and dynamic features*. https://doi.org/10.1016/j.jnca.2012.10.004

Kumari, R. and Kr., S. (2017). Machine Learning: A Review on Binary Classification. *International Journal of Computer Applications*, 160(7), pp.11-15.

Mohaisen, A., Alrawi, O. and Mohaisen, M. (2015). AMAL: High-fidelity, behavior-based automated malware analysis and classification. *Computers & Security*, 52, pp.251-266. Retrieved from: https://doi.org/10.1016/j.cose.2015.04.001

Montana State University (2005). *Operating System "Protection Rings" Diagram*. Retrieved from: https://www.cs.montana.edu/courses/spring2005/518/Hypertextbook/jim/media/intelPrivilegeLevels.gif [Accessed 10 Dec. 2019].

Moser, A., Kruegel, C. and Kirda, E. (2007). Limits of Static Analysis for Malware Detection. *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007)*, 23, pp.421-429. Retrieved from: https://ieeexplore.ieee.org/document/4413008.

Mostfa Kamal, S. U., Abd Ali, R. J., Alani, H. K., & Abdulmajed, E. S. (2016). Survey and brief history on malware in network security case study: Viruses, worms and bots. *ARPN Journal of Engineering and Applied Sciences*, *11*(1), 683–698. Retrieved from https://www.researchgate.net/publication/301695496_Survey_and_brief_history_on_malware_in_network_security_case_study_Viruses_worms_and_bots

Netmarketshare (2019). *Operating System Market Share*. Retrieved from https://netmarketshare.com/operating-system-market-share.aspx

Nominet (2017). *TECHNICAL WHITEPAPER - TRACKING THE WANNACRY RANSOMWARE*. Nominet, p.3. Retrieved from: https://media.nominet.uk/wpcontent/uploads/2017/11/21123044/WannaCry-Whitepaper1.pdf.

Raman, K. (2012). Selecting Features to Classify Malware. Adobe Systems Inc, pp.1-5 Retrieved from http://2012.infosecsouthwest.com/files/speaker_materials/ISSW2012_Selecting_Features_to_Classify_Malware.pdf

Rathore, H., Agarwal, S., Sahay, S. and Sewak, M. (2018). Malware Detection Using Machine Learning and Deep Learning. *Big Data Analytics*, 11297, pp.402-411. Retrieved from: https://arxiv.org/pdf/1904.02441

Schultz, M., Eskin, E., Zadok, E. and Stolfo, S. (2000). *Data mining methods for detection of new malicious executables*. Oakland, CA, USA: IEEE, pp.1-4. Retrieved from: https://doi.org/10.1109/SECPRI.2001.924286.

Selamat, N., Mohd Ali, F. and Abu Othman, N. (2016). Polymorphic Malware Detection. *2016 6th International Conference on IT Convergence and Security (ICITCS)*. Retrieved from: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7740362.

Sihwail, R., Omar, K., & Ariffin, K. A. Z. (2018). A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis. *International Journal on Advanced Science, Engineering and Information Technology*, *8*(4–2), 1662–1671. https://doi.org/10.18517/ijaseit.8.4-2.6827

Symantec (2017). *ISTR Ransomware 2017*. Insider Threat Report. Mountain View, CA, USA: Symantec, pp.6-8. Retrieved from: https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istrransomware-2017-en.pdf.

Tesauro, G., Kephart, J. and Sorkin, G. (1996). Neural networks for computer virus recognition. *IEEE Expert*, 11(4), pp.5-6. Retrieved from: https://pdfs.semanticscholar.org/1b6a/47119f7fea9229f786684f9e83c1441c2075.pdf.

Ucci, D., Aniello, L. and Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security*, 81, pp.123-147. Retrieved from: https://www.sciencedirect.com/science/article/pii/S0167404818303808.

Windows Developer Center. (2019). *PE Format - Win32 apps*. Retrieved from: https://docs.microsoft.com/en-us/windows/win32/debug/pe-format

Zhuge, J., Holz, T., Han, X., Song, C. and Zou, W. (2006). *Collecting Autonomous Spreading Malware Using High-Interaction Honeypots*. Retrieved from: http://www.cs.ucr.edu/~csong/icics07.pdf.

Appendix A: Research Comparison & Flowchart

Comparison Table

Table 2: Comparison of Research Material

Author(s)	RT/F	MLF	ACC	MLC	FE	TS	MSD	DT	BC	HP
Alam et al., 2015	✗	✓	✓	✓	✗	✗	✓	✗	✓	✗
Bayer et al., 2009	✓	✓	✓	✓	✓	✓	✗	✗	✓	✗
Bazrafshan et al., 2013	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗
Belaoued, M., Mazouzi, S. 2016	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Chui et al., 2019	✓	✗	✗	✓	✗	✓	✗	✓	✗	✗
Dhanyasree, Krishnan and Ambikadevi Amma, 2019	✓	✗	✗	✓	✗	✓	✗	✓	✗	✗
Dolan-Gavitt et al., 2015	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗
Fattori et al., 2015	✗	✓	✓	✗	✗	✗	✓	✓	✓	✗
Gavrilut et al., 2009	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Islam et al., 2012	✗	✓	✗	✗	✓	✗	✓	✓	✓	✗
Kumari, R. and Kr., S., 2017	✗	✓	✓	✓	✓	✓	✗	✓	✓	✗
Mohaisen, A., Alrawi, O. and Mohaisen, M. 2015	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Moser, A., Kruegel, C. and Kirda, E., (2007)	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗
Nominet., 2017	✗	✓	✗	✗	✓	✗	✓	✗	✓	✗
Raman, K., 2012	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗
Rathore, H., Agarwal, S., Sahay, S. and Sewak, M., 2018	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Selamat, N., Mohd Ali, F. and Abu Othman, N., 2016	✗	✓	✓	✗	✓	✗	✓	✗	✓	✗
Sihwail, R., Omar, K., & Ariffin, K. A. Z., 2018	✗	✓	✓	✗	✓	✗	✓	✗	✗	✗
Symantec., 2017	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗
Tesauro, G., Kephart, J. and Sorkin, G., 1996	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
Ucci, D., Aniello, L. and Baldoni, R., 2019	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
Zhuge, et al 2006	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓

KEY: RT/F = Random Tree/Forest, **MLF** = Malware Feature(s) , **ACC**= Accuracy, **MLC**= Machinelearning Classification, **FE** = Feature Extraction, **TS** = Training Set, MSD = Microsoft-DOS, DT = Data Mining, **BC** = Binary Classification, **HP** = Honeypot

Research Methodology Flowchart

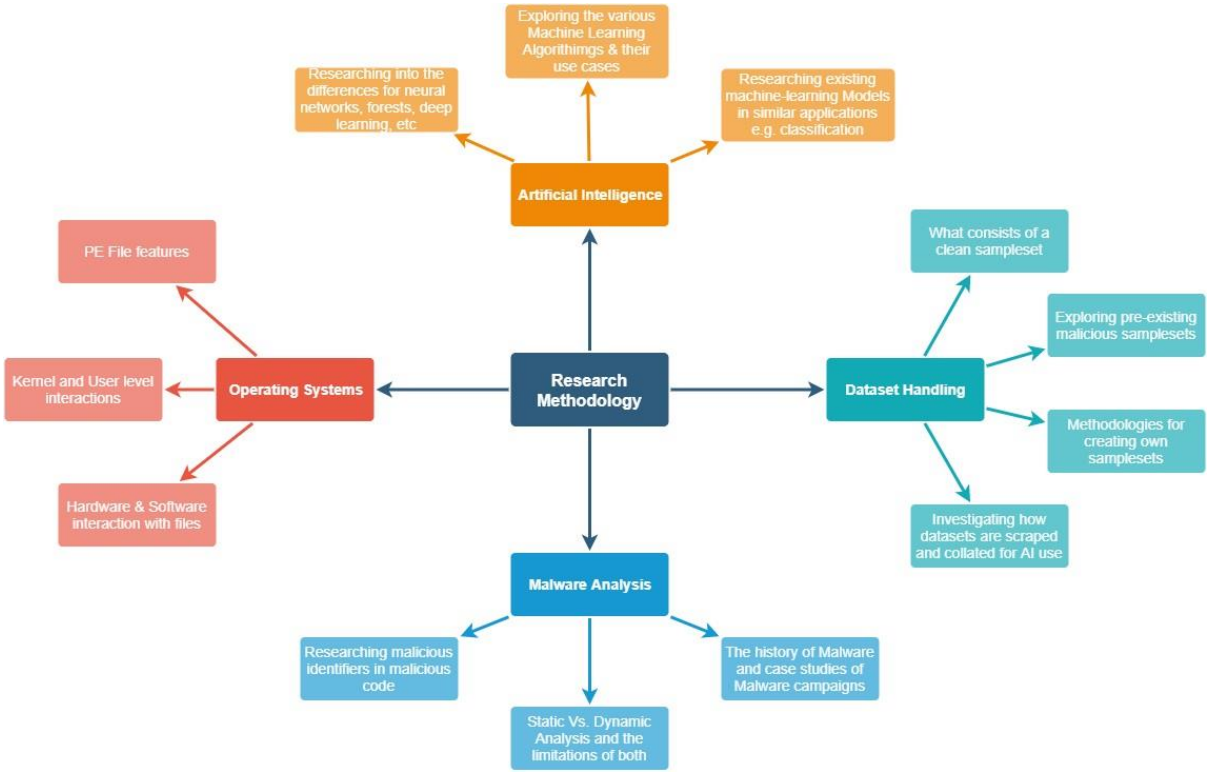
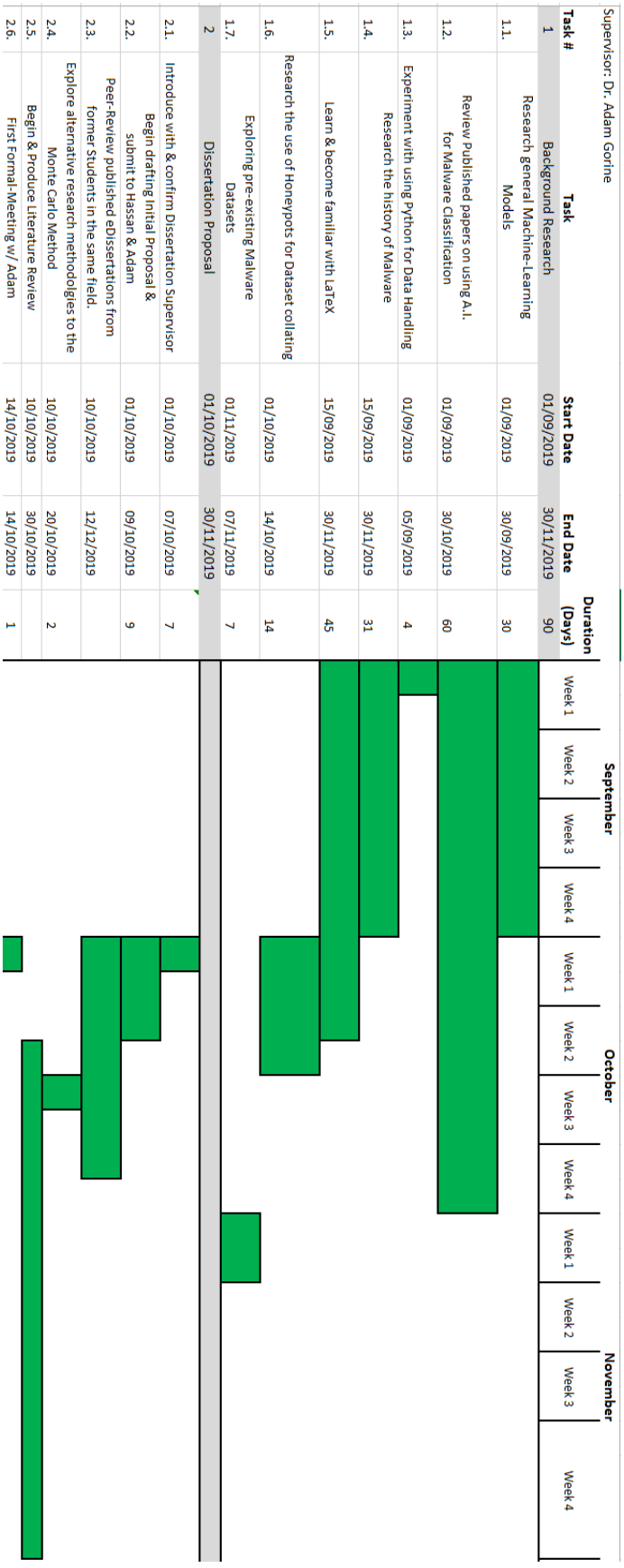


Figure 16: Research Methodology Flowchart

Appendix B: Gantt Chart & Dissertation Timeline

Gantt Chart Preceding Proposal Submission:



Expected Dissertation Timeline:

Table 3: Expected Dissertation Timeline

Task #	Task	Month Start	Month End	Progress
1	Research Machine-Learning Models & Frameworks	June 2019	December 2019	
1.1	Review academic publications on A.I. and Malware Classification	October	December	
1.2	Experiment with Python for Data Handling	September	October	
1.3.	Research into the history of Malware	June	September	
1.4.	Investigate the processes of static & dynamic malware analysis	June	December	
1.5.	Research the use of Honeypots for Dataset collating	October	November	
1.6.	Investigate pre-existing Malware Datasets	November	November	
1.7.	Learn & become familiar with LaTeX	October	December	
2	Dissertation Proposal	October 2019	December 2019	
2.1.	Propose idea & confirm Dissertation Supervisor	October	October	
2.2.	Begin drafting formal proposal & submit to Hassan & Adam	October	October	
2.3.	Peer-review published academic work from similar fields	November	November	
2.4.	Explore alternative research methodologies to Monte-Carlo	October	October	
2.5.	Produce Literature Review	November	November	
2.6.	First Formal-Meeting w/ Adam for reflection	October	November	
2.7.	Begin collecting example dataset	November	November	
3.	Dissertation	December 2019	May 2020	
3.1.	Reflection meetings w/ Adam for work before next Semester	December (2019)	December (2020)	
3.2.	Development of various classification frameworks in Jupyter	January	May	
3.3.	Reflect on & Introduce various Dataset sampling methods	January	May	
3.4.	Begin feeding Dataset into machine-learning framework	January	May	
3.5.	Consistent reflections on training output and improvements	January	May	
3.6.	Writing Research Project hand-in	December (2019)	May	
3.7.	Prepare for Dissertation Defence Session	April	May	