Classifying Advanced Malware into Families based on Instruction Link Analysis

Alsa Tabatabaei

School of Computing, Science, and Engineering

University of Salford

Manchester, UK

Submitted in Partial Fulfilment of the Requirements of the Degree Master of Philosophy, 2018

`

Table of Contents

`

`

`

`

List of Figures

`

`

`

List of Tables

`

This thesis is dedicated to the memory of my parents. I wish they could see this process through to its completion. I miss them every day and but I am glad to have their support to make this journey possible, as well as plenty of hearty encouragement. To them with love and eternal appreciation.

`

Acknowledgement

My two research journeys have been one of the most challenging and unforgettable times of my life. Fortunately, I met many people along the way. They have appeared in my life like a blessing, just at the right time and with purpose. I would like to take this opportunity to extend my sincere gratitude.

I am also grateful to the members of my committee for their patience and support in overcoming numerous obstacles I have been facing through my research. I am very grateful to Dr.Ali Dehghantanha for bringing me into the field of cybersecurity. He has taught me not just how to do research, but more importantly, the passion, the enthusiasm, the diligence, and the discipline of being a world-class researcher. Dr.Mohammad Saraee thank you for sharing your insights in Machine Learning. You have my immense gratitude. I am grateful to Dr.Tooska Darghahi for helping with the correction and renew access VirusTotalInteligence.

I would like to thank all my friends at the University of Salford, for their enjoyable company and discussions, both technical and non-technical. Thank you for accepting nothing less than excellence from me.

A very special gratitude goes out to the staff of PGR and school of computing for the last –minute favour and consideration.

I would like to acknowledge the support provided by the University of Salford. I am very grateful for their patience and support in overcoming numerous obstacles. Without their precious and continuous support, it would not be possible to finish my research journey.

I would like to thank my family for supporting me spiritually, mentally throughout my study and my life in general.

Last but not least, I am grateful to VirusTotal for being generous and supportive towards my research and providing their private key to help me build the datasets.

Thanks for all your encouragement!

`

Declaration

As the author of this thesis, I thereby confirm that no portion of the work presented in this thesis is submitted in support of an application for another degree or qualification at Salford or any other university.

`

Abstract

With the ever-increasing growth of network resources, a great number of organizations are extremely dependent on the internet for operational activities as such, exposing their sensitive and confidential information to intrusion or invasion by saboteurs and corporate theft leaving them exposed and vulnerable. This revolution has led to the fast and emerging growth of malware with high complexity which circumnavigates a lot of security asset to keep safe sensitive organizational data. The development of these complex malware has become a big threat in today's computing world such as Advanced Persistent Threats (APTs). APTs is customized for a specific target and can be subtly altered to avoid detection. In that, APTs attack is considered as a serious problem whose devastating effects cannot be overemphasized.

To combat this propagate, malware analysers have been deployed in Machine Learning and Data Mining techniques or the combination of both techniques to automatically spot malicious file. A lot of feature engineering approaches are explored to improve the performance of detection/classification system if feature engineering approach provides sufficient information of malware type for clustering purposes, then this indicates the possibility of developing learning method which performs better. In fact, there are motivations for incorporating feature selection in data classification employed on data from a machine learning perspective. The main focus of this research is on static analysis approach. To find the dominated features in one malware family, an experimentation with the association, link analysis, and segmentation algorithms are employed. The model performs on a publicly available dataset on Kaggle and GitHub. The experimental data gave supportive validation of the proposed feature selection model by Gaussian Mixer Model in R environment.

`

Chapter One

INTRODUCTION

1 Overview

This chapter presents the introductory part of the research which includes the background of the study and the other important keys pathways to give a general overview of the research work. It is categorised into various sub-section presented sequentially as follow:

1.1 Background

Malware is a malicious program that is intentionally developed to harm computer systems(Sharma and Sahay, 2016).The emerging growth of malware with high complexity has opened doors to a confidential and sensitive data breach by hackers on a daily bases. This can circumnavigate a lot of security asset set up to mask or keep safe sensitive organisational data. A big threat in today's cybersecurity era, the variant of complicated malware mutates their code by crafting to look different from each other even from the same malicious family to evade detection system. Finding samples of all the new advanced malware from different groups created every day is a big challenge. Although, creating new signatures and getting them deployed effectively is an even bigger one.

Cyberspace is a highly dynamic man-made domain with a high degree of uncertainty and incomplete data which must be transformed into knowledge to support precise and predictable cyber risk. Therefore, to combat malware effectively, it is desirable to identify the type of malware with strong structural similarities or dissimilarities. Technically, if an unseen malware crafted by some of the examined patterns; it is possible to detect suspicious file on the host in the real-time. As a result, this type of detectability can formalize cyber asset.

`

A sophisticated malicious program, like APTs campaign, generates a unique, new instance of a malware family for each victim, to create a new way of attack. One challenge faced by IT security teams is how to spot signs of an advanced malware. Static, dynamic analysis or combination of both techniques equip them to spot data exfiltration occurrence. APTs are particularly dangerous for enterprises, as hackers have ongoing access to sensitive company data. To skip pattern matching detection technique, malware creators employ several sophisticated obfuscation mechanisms to ensure they can break into systems. Obfuscation techniques preserve the program‘s semantic and functionality while, at the same time, make it harder to understand or read the program’s structure such as packing, polymorphism, and metamorphism. In which case, the signature-based detection techniques fail while dynamic-based detections are mainly time-consuming according to (Fraley, 2016) (Gandotra, Bansal and Sofat, 2016). One possible solution to this is to automatically classify malware types and assign those families in relation to opcodes similarities or dissimilarities factors. In this case, feature selection techniques will come to the picture. Feature selection is widely used so that a set of relevant features can be chosen without searching for all possible subsets according to certain relevance evaluation criteria.

By considering the fact that malware writers change malware appearance on each infection, though; they design functionally equivalent (Kaur and Singh, 2014) (Mathur, 2013). Matching instruction syllable into pre-exciting groups and identify their respective malware families becomes a potential solution. Thereby, the concept pattern discovery techniques have been utilized by cybersecurity researchers likewise in the health system for cancer detection and prediction. Discovering frequent sequential patterns and measuring the dissimilarity and similarity between discovered patterns have been studied by researchers (Ding *et al.*, 2014) (Fan *et al.*, 2016). Monitoring the opcode instructions of different malware groups and present opcode relationships by segment analysis is examined in this research.

In order to create an abstract description of each malware family; there is a need for a comprehensive study which consists of several stages. The first stage is data preparation. Data preparation in current research includes: collecting malware sample files, reversing the samples code, creating raw opcode dataset. To create the virus code extracted from variants of the same malware family as a raw dataset. To complete data preparation, the network between extracted instruction syllables from each individual malware type is generated. The generated graphs present features in the secondary datasets. The feature is an individual measurable property of a malware opcode being observed. The number of features will be used is called the dimension. Assessment of link Analysis results is done by data segmentation. Data segmentation captured the commonalities of all features within a group. It also explains how the variable and variable

`

levels contribute to the formation of clusters. Important variables are identified by segmentation analysis and the performance of its clustering will be evaluated by Expectation-Maximization algorithm.

On this assumption and motivation, a new feature selection model is developed. The proposed model not only to discover relationship and links between code instructions but also effectively regroup malware types to the respective family. The experimental evaluation in this research was designed to be much more thorough to pave the way toward deployment of the approach for use by cyber-security analysts.

As far as this research is concerned, this study is the first to establish a model based on link analysis and clustering to categorize malicious families. While there have been promising results on the usage of data mining and unsupervised learning techniques on data classification problems (Siddequi, Wang and Lee, 2008) (Ucci, Aniello and Baldoni, 2017) (Das et al., 2016) (Liu, Li and Zhao, 2003) (Bernardi et al., 2017)(V.SHARMILA, I.VASUDEVAN and 1ASSOCIATE, 2014)(Ding et al., 2014)(Ge, Wang and Xu, 2016) (Mohaisen, Alrawi and Mohaisen, 2015)(Pai et al., 2016).

1.2 Research Problem

There is a large amount of variously complicated malware and it is insufficient to compare them only as a single class against the signature repository. The intention of crackers is to reuse the previous chunk of codes, for instance, well-known loop performing infection. Therefore, while the main functionality of a malware sample remains immutable during its mutation. Malware types can be merged into groups of malware family with common functionality which is called malware families.

By knowing that ,frequent functions are used, the probability to trace the previous family based on the instruction parcel commonalities have been studied by security researchers (Le, Zincir-Heywood and Heywood, 2017)(Yewale and Singh, 2017)(Scott, Fellow and Technology, 2017)(Ye, 2017)(Deka, Sarma and Panicker, 2017)(Mohammadi and Hamzeh, 2017). However, code instruction commonalities come at the cost of an exponential increase in the dimensionality of the resulting feature vectors, required storage and computation time. Hence, previous work that uses opcode mining approach commonly chose small instruction parcel.

Part of the challenge in finding commonalities among samples is feature extraction/construction. The process of finding the most relevant features for each malware family is feature selection. Further yet which ones belong together as part of an attack is akin to finding needles in a huge haystack let alone to figure out which ones belong to the same knitting set.

The art of evaluating which subsets of codes is significant for detection purpose is still under investigation. Besides, which feature selection methods perform better on prediction model while alleviating dimensionality is another challenge to be addressed. Finding the most effective features and decreasing the computational load in cluttered scenes is another area that needs further perusal. The results of an efficient feature selection system could mean a substantial saving in terms of time for data collection and space for future data storage. There are many advantages of feature selection for malware detection; however, little effort has been devoted, until now, to apply feature selection methods of machine learning to deal with malware redundant and irrelevant features.

To remedy this situation, extracted feature compress by feature selection methods to produce a more normal distribution prior to clustering. Feature selection and classification accuracy for malware detection vastly have been investigated (Ahmadi and Giacinto, 2015)(Deepa, Radhamani and Vinod, 2015)(Zolotukhin and Hämäläinen, 2014). Classification analysis will both describe the separation between groups and assign new malware to a known group based on a list of selected features (Poulsen, 2013).

This will result in models that will produce a better fit. For clustering malware family's samples, many approaches are published. Different similarity measures are examined but without thoroughly discussing their choice on dissimilarity or link of structure code. For the purpose of identifying malicious families, it is needed to extract the pattern necessary for its detection. The complex malware samples that are similar to each other are grouped into the same cluster (Canfora *et al.*, 2015). Each cluster is characterized by a membership function with statistical mean and deviation. The extracted feature, corresponding to a cluster, is weighted and contained in the cluster. The dataset is grouped into clusters, based on similarity of the opcode. Reactions to cyber threats based on what has already been found are limited by the victim of APT attacks(Thomson, 2011).

1.3 Research Hypothesis

On this assumption and motivation, a new model in this thesis will be developed to combine static analysis, association rules and a clustering method. The outcome could be used to train a classifier over a decent size dataset to provide robustness against new unseen malware. If an unknown complex malware uses all or some of the same functionalities as are used by the training dataset then it is possible to detect this unseen malware. Based on the investigation, the following questions are considered in the research to figure out an effective method to a group known malware samples.

1.4 Research Questions

In this research, data mining method was used to identify groups of opcode patterns that are highly correlated with each other in a family. The following three research questions that reflect the main topics of the research have identified.

1. How can advanced malware opcode structures be characterized based on similarity?

It is logical to assume that different malware families will contain a different kind of functions code. The majority of current methods for measuring opcode similarity collect only frequency and sequential patterns. This may be a great way to classify malware, but it doesn't tell much about significant patterns. This would be comprehensive enough to identify a malware sample into the respective family. Thus, methods of feature engineering for providing the most informative features is needed to be explore.

2. How can link analysis and segmentation techniques be utilized for malware classification?

The second research question is focused on research into analyse opcode link for segmentation. By selecting rules with higher lift value and significant odds ratios, and considered individual each observation matched these rules will form a malware group. Our study adds to the growing body of research exploring the association between malware samples in a family.

3. How to select a smaller number of variables from a large pool of opcode structures as input variables, reducing the dimensionality of data for building better mode?

To define groups of malware samples with similar opcode links, the segmentation results in present similarity, distance, and distance metrics.

2. What possibilities does the similarity opcode analysis have for detecting advance malware?

The main goal of the third research question is to explore the benefits of the proposed method for feature engineering in detecting advanced malware. We will pay attention to the verification of different characteristics of similarity opcodes and to the various representation of malware families. Specifically, we will focus on different functions to measure the pattern distances and the identification of each group

`

based on the opcode graph. To build a predictive model for malware detection, segmentation results can be utilized.

1.5 Research Motivation

The present study designed to investigate the prevalence and patterns of opcode structures discriminate the similarity of each frame to malware groups in a cluster. Unsupervised learning techniques are applied to the execution traces, with the aim being to create a predictive model which can accurately and speedily differentiate between malware families. The motivation behind this research is shaped based on the proposed method by (Liu *et al.,* 2017)and (Hu *et al.,* 2013) for detecting variants malware with an opcode-based feature into clusters. Authors proposed a fast density-based clustering technique. Their experiment tackled feature selection and data dimension reduction, computational power, and storage required to find. Besides, the limitation that only convex decision regions are allowed is considered in their proposed method. Conversely from the research carried out during the course of this work, there is relatively limited research in literature concerned with the similarity between two groups of advanced malware. Therefore, this study aims to experimentally investigate the advance malware similarities and differences for attack indication purpose based on association rule mining.

1.6 Research Challenges

The potential challenge collated from the extensive literature review and preliminary investigations is classifying the various given potential advanced malware binary codes into various pattern features for clustering purpose. Thus, the challenge is to select a relevant set of features, so that, the classification model can be built in less time with high accuracy. Tackling this challenge will involve discovering malware signature by analysing and observing the patterns in the assembly level instructions of various x86 binary malware executables. Prioritize limited APTs malware resources is consider another difficult task of this research.

1.7 Justification, Aims, and Objectives

In this research, a comprehensive and robust approach will be used to understand and assess the mechanisms and factors that affect the efficiency of clustering.

1.8 Research Aims

- The primary goal of this research is performing similarity measures between different malware families for the same variant utilizing data mining concepts in conjunction with unsupervised algorithms.

- The secondary goal of this research is to evaluate unsupervised algorithms towards automated clustering of advanced malware into various groups accordingly.

1.9 Research Objectives

Through experimental and numerical simulation, the key objectives are listed below:

I. Study the evolution and current state of classification methods for advanced, as well as recent progress made to analyse and detect sophisticated obfuscated malwares families.
II. Develop static analysis to enable the detection of sophisticated obfuscated malwares
III. Data exploration using association mining, sequential pattern mining, relevance feature discovery on clustering
IV. Enhance the unknown pattern discovery to reduce the side effects of noisy patterns, low frequency problem.
V. Develop a technique aiming at better analysing malware opcode with particular emphasis on evaluating features to reduce the dimensionality of high dimensional opcode vector.
VI. Facilitate the categorization of complex malware by measuring the link instructions in order to improve the previous detection coverage.

To meet the research objectives: detectable patterns of opcode instructions firstly are discovered and then the most relevant of them within each family is uncovered. Afterwards, a collection of the most important predictor variables is ready for training purpose in classification technique. Specifically, objective III and IV can be achieved by usage of association rules mining and examining clustering techniques. Link analysis is the data mining technique that addresses this need. This model is used to generate data for future data-driven decision making on the multinomial dataset. The models are designed in SAS® Enterprise Miner. When all variables generated by link analysis have been fed into the segmentation node a desired number of clusters are formed automatically. As a result, important variables for each cluster will be retrieved. The validation partition is set aside and is used to test the accuracy and fine tune model.

1.10 Significant of the Study

The uniqueness of this study exists in dealing with uncertain labels and a reduction in the volume of features to mitigate misclassification. Hence it enables the introduction of link analysis in malware features construction for clustering purpose. This is in line with multiple interpretations conveyed by frequent sequential opcode patterns, which help to indicate zero-day attacks on the whole systems. Such a combined approach could leverage the relative strengths of classification to yield a stronger overall detector in terms of detection rate and speed. The strategy of considering multiple aspects to compare samples can leverage different facets of the malware, improve the overall classification rate, and potentially improve the robustness of performance.

1.11 Research Scope and Limitations

The proposed approach is presented under the following limitations:

- Due to time constraints, the ability to collect large datasets of malware samples from APTs groups was somewhat limited. However, the number of malware samples selected was arbitrary and the final 250 files were fully populated from a data completeness perspective.
- Fast-paced malware sample, will the datasets being used for the experiment will be questioned.
- Last resort, stick to the existing database, to free from any specific malware family to make sure the method will/could work with incoming, new malware.

Needless to say, for research purposed these issues are always wondered.

1.12 Research Methodology and Research Methods

Academic research is defined as the art of scientific investigation which aims to investigate the unknown knowledge. This chapter provides a detailed description of how the research is to be carried out.

1.12.1 Research Methodology

Research methodology is a systematic way to solve a problem. The nature of academic research requires a different kind of steps which is listed below:

- Problem discovery and research questions

- Formulation research problems and objectives
- Research design
- Literature survey and reference collection
- Assessment of the current status of the topic chosen
- Sampling plan and data analysis
- Interpretation of results
- Experiment report

These steps are depicted in Figure 1-1-2 and research architecture is illustrated in Figure 1-1.

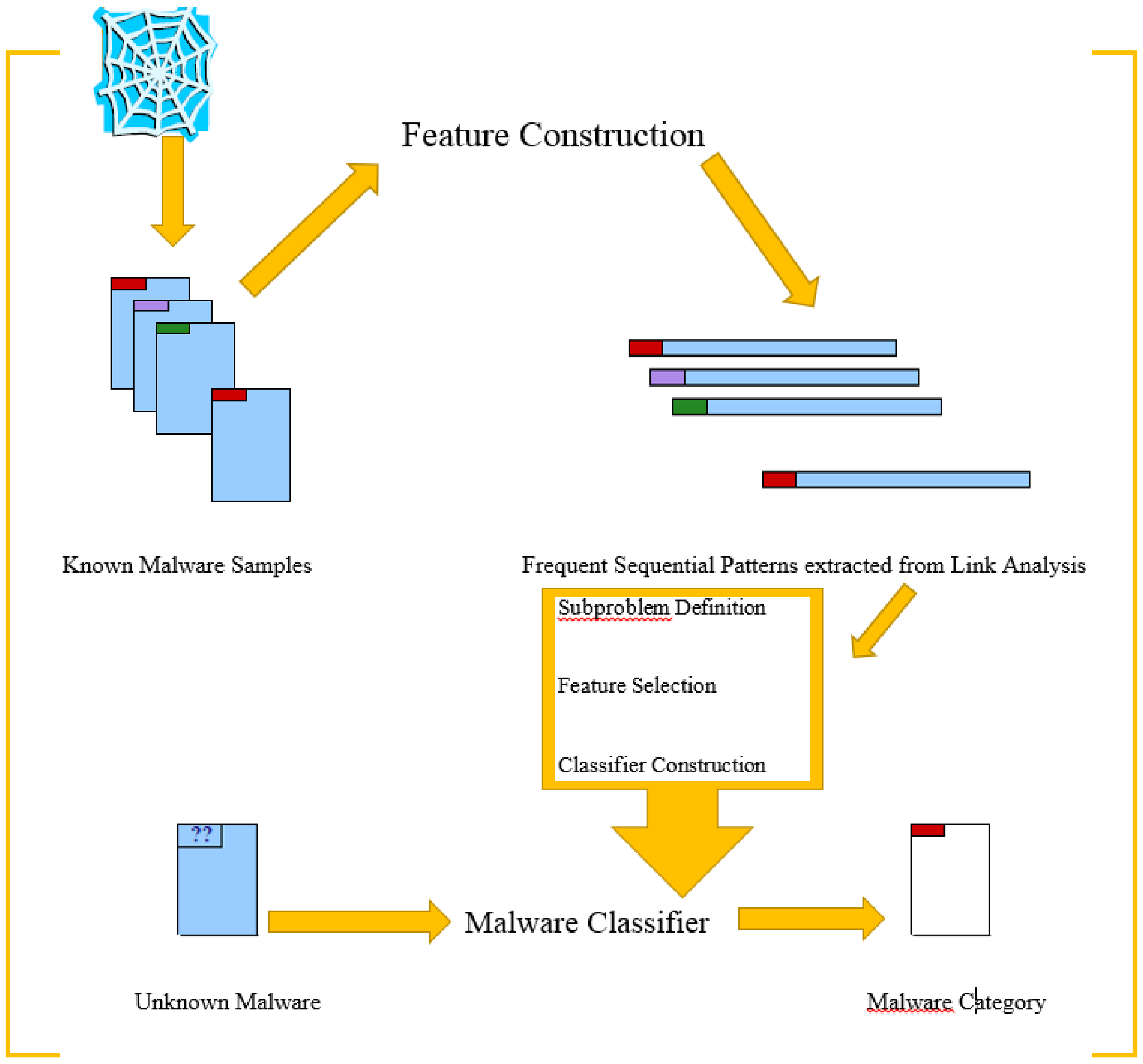


Figure 1-1Research Architecture

1.12.2 Research Methods

In order to get a solution to a problem research methods is concreated. The results are presented in tables and graphs. Once data of each case study are collected. Association rule mining applied to analyse patterns of malware family based on opcode graph.

In this case, a proper representation of each malware family will be pulled out. Hereafter, link analysis measurement will be run. Building segmentation analysis places observations into groups according to the natural association between the observations. The overview of the methodology is demonstrated in chapter 3 section 3.2.

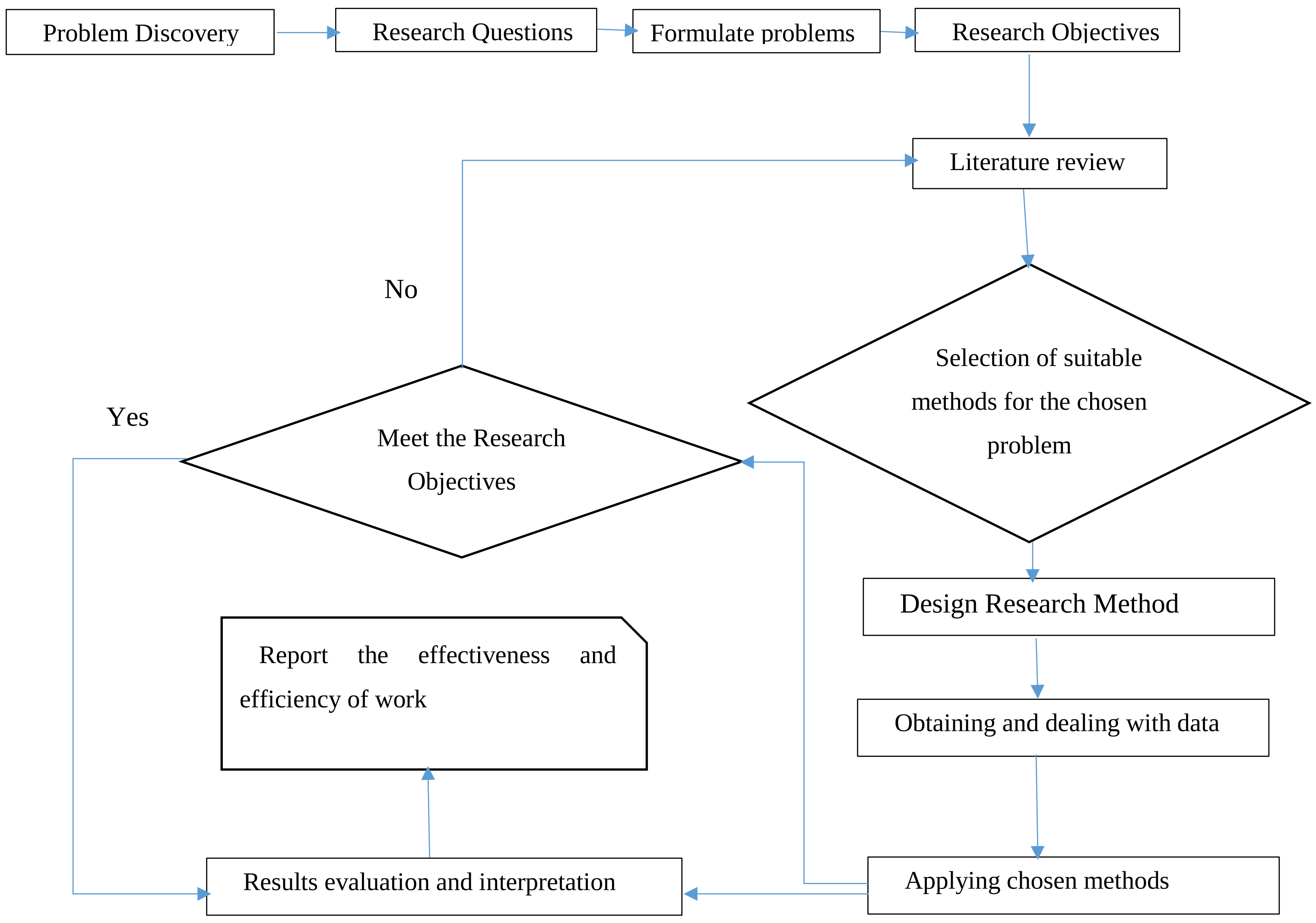

Problem Discovery
Research Questions
Formulate problems
Research Objectives
Literature review
No
Selection of suitable methods for the chosen problem
Yes
Meet the Research Objectives
Design Research Method
Report the effectiveness and efficiency of work
Obtaining and dealing with data
Results evaluation and interpretation
Applying chosen methods

Figure 1-1-2Research Methodology Flow

`

1.13Research Overview and Structure

This research presents a three-stage method to cluster malware into a group by comparing their opcode similarities to existing static traces and assigning it to the most similar group. First creating a shortlist of samples malware from different families. The second stage opcode extraction in order to estimate their similarity/dissimilarity distance between opcode. Finally k-means, hieratical, and pam clustering is run base on the combination of opcode interaction to find out which sample belongs to which family. A set of code influencers and aggregates them based on how close they are to each other is filtered. The results demonstrate that the proposed model provides useful clustering variables by segment analysis technique.

This research is organized as follow chapter one provide an introduction to the research gap and problem. An overview of the most important previous works based on machine learning and data mining discusses in chapter two. The proposed method in more detail and stating the detect mechanism is described in chapter three. The details the techniques discussed in this chapter too. Chapter four provided more information about research case studies and experiment. Chapter five presents the examination of results. Finally, chapter six wraps the whole thesis up by giving a conclusion. The recommendation and future work stated in the same chapter.

`

Chapter Two

Related Literature Review

2 Overview

As was mentioned in the previous chapter, this research presented malware classification that integrated the static analysis of malware with a data mining system using association rule and clustering techniques. Relevant background material related to malware classification and detection techniques specific to distinguish malware families is studied extensively. Furthermore, several existing zero-day sophisticated malware detection techniques are surveyed. In this section, a brief overview of the several topics that are relevant to the research aims is discussed. First, malware definition followed by APTs is considered. Then, malware analysis and ML in general, with emphasis on data mining techniques. It is followed by the discussion on Link Analysis and its potential role in malware detection, summarized by key findings. Consequently, the purpose of this chapter is to review the state of the art literature on the above-mentioned topics in such a way that rationalise their connections to the proposed research hypothesis.

2.1 Characterisation of Malware

Maliciousness is defined as the risks exposed to users, computers, networks or infrastructures -such as viruses, worms, Trojan horses, rootkits, (Yewale and Singh, 2017). Evidently, a malicious code scans for vulnerabilities of a host or network target. Once the malware finds a way to enter the system, takes control of it and it performs silently some unauthorized action, gathering sensitive information and etc. Sometimes the tension of a virus is not to cause damage, but to clone itself onto another host. The virus causes damage it is more likely to be detected. On the other hand, advanced malware with a very small footprint could remain undetected for a very long time (Shenwen, Yingbo, and Xiongjie, 2015).

There is unstoppable growth in the number and variant of attacks that lies on the continuous generation of refined and aggressive malicious code (Azmoodeh *et al.*, 2018). Malware can be classified into different categories based on their intent. Table 2-1 presents their capabilities as stated by (Gharacheh et al., 2016) (Deka, Sarma and Panicker, 2017) (Sharma and K. Sahay, 2014) (Fraley, 2016) (Thunga and Neelisetti, 2015).

`

Table 2-1Malware Variant

	Capabilities	Way of Defence
Malware variant	• Able to replicates by inserting itself into other programs • Able to gain access to the private system • Able to disrupt a computer system • Able to gather sensitive information	• Anti-virus
	• Able to skip traditional signature-based defences • Able to bypass inefficient security solutions. • Able to detects a VM environment and to hide and not perform malicious behaviour	• Sandbox • Firewall

Although any specific malware has its own capability; the internal state of malicious software remains the same. Since it is difficult to create a new virus from scratch; usually, malware writers reuse old virus. However, to hide detection the hacktivists they tend to change the obfuscations (syntax) more than the behaviour (semantic) of any specific malware. To be seen, malware creators simply modify internal components while preserving the code's functionality and behaviour. Therefore, the comparison of the functional blocks of code structure to each known malware might accommodate a zero-day malware (Kaur and Singh, 2014) (Ucci, Aniello, and Baldoni, 2017). Note that throughout this research, the term virus and malware are used interchangeably.

2.2 Understanding Advanced Malware

The evaluation of malware can morph and mutate like an infectious virus to bypass signature present in the antivirus database without leaving a footprint (Rathnayaka and Jamdagni, 2017). There are multiple transformation techniques, including code permutation, register renaming, expanding and shrinking code, and the insertion of garbage code or other constructs (Cosovan, Benchea and Gavrilut, 2015). Such a case every time the malicious file presents itself, it looks different as it can change itself or evolve according to the theory of Darwin that is considered as a self-propagate feature. Some advanced malware samples employ two or three of these techniques together. Therefore, the complexity of detection and remediation in hosts and networks will increase significantly. Notably, determined attackers target misconfigured and unpatched systems to withdraw confidential information. They also aim regulated data eventually by

`

evading security defence systems such as a virtual sandbox that many organizations rely on today (Ling, Putra, and Ling, 2017). In other words, the attack data is able to travel across the network in the form of packets. When the characteristics of a certain malware are analysed, it is the fact that the evolution goes on and there are other species like that threat, live and somewhere compromising the system in a manner that's not visible at all. That being the case, the APTs attack was found to live long and evade detecting in hosts and networks for more than a year (Shenwen, Yingbo, and Xiongjie, 2015).

2.2.1 Understanding Advanced Persistent Threats (APTs)

In 2006, the United States Air Force (USAF) (https://www.airforce.com) analysts introduced the term of Advanced Persistent Threats (APTs) to describe attacks on governments and commercial organizations. Though, APTs also often target the valuable information found in smaller organizations. Frequently, due to lack of security technologies and expertise, these types of network infection have remained uncovered, so APTs coders never get caught. The largest APTs groups, fifteen of the most active APTs families, is classified by FireEye (https://www.fireeye.com/current-threats/APTs-groups.html), all of which are still active across the cyber threat landscape. In practice, APTs can be based on software, hardware, social engineering or some combination of the three(Nissim *et al.*, 2015). For this, cybercriminals can coordinate their attacks among various delivery venues, including email, the Web, social media, legitimate files, etc. (Ostrowski, 2014). APTs generally do not cause damage to company networks or local machines. Instead, the goal of it is most often data theft. This research is based on software only which effectively hide the attack from traditional malware detection analysis.

APTs or multistage attack defines typically a more sophisticated and complicated threat vector that targeted a business by developing a custom exploit towards specific goals. The consequences of these incursions can be severe, and in some extreme cases cause a business to go bankrupt. On top of that, an anti-APTs solution does not exist at the time of this review (Shenwen, Yingbo, and Xiongjie, 2015). More advance sophisticated components are actively adapted by APTs writers to hide malicious software from inspection efforts. Nearly every new APTs attack includes features which are able to regulate its activities, to self-propagate, to strategically deliver other malware, and to maximize its damage while minimizing its footprint (Shenwen, Yingbo, and Xiongjie, 2015). The evasion techniques will easily escape security filters. Originally, this leads to a need of a comprehensive analysis which consists of several stages such as adding detection for known modules, collecting samples, reversing the samples, decrypting sophisticated encryption and compression schemes, understanding lateral movement, etc. To put everything together and reconstruct how the APT acted, what were its habits, what species it attacked, and

`

how these attacks were coordinated (Vert, Gonen and Brown, 2014). Simply put, this requires time and patience which is not practical and desirable. On the other hand, APT attacks are difficult to uncover as specifically, they expect to encounter a Virtual Machine (VM) environment and develop their attacks accordingly. As it explained earlier, like a biological virus this generation is designed in multi-state and multi-vector attacks (Bolton and Anderson-Cook, 2017). To combat the complex malicious programs, security community will require sharpening their skills to discern and block advanced malware-free attacks before attackers can compromise the network environment.

2.3 An Overview of Static and Dynamic Analysis

For the purpose of malware classification and detection, traditional analyses could be further divided into two main categories: static analysis and dynamic analysis. Static features are extracted without executing the file under analysis, while dynamic ones are just the opposite. Dynamic features are obtained by watching the behaviours of a program during the run-time while static features are obtained by decompiling the byte code of a suspicious program. To give an overall view of these approaches , pros and cons of each of them are presented in Table 2-2(Gandotra, Bansal and Sofat, 2016)(Rezaei and Afraz, 2016)(Ahmadi *et al.*, 2013)(Shabtai *et al.*, 2012)(Damshenas, Dehghantanha and Mahmoud, 2013)(Gandotra, Bansal and Sofat, 2016)(Shijo and Salim, 2015)(Shalaginov, 2017)(Fraley, 2016)(Mohaisen, Alrawi and Mohaisen, 2015).

Static analysis just looks at the file itself and tries to extract information about the structure and data in the file. A case in point, the source code is one of static analysis method to understand the program behaviour without actually executing it. Dissemblers like IDA-Pro is used to decompile the source code, variables, pose functions and recognize the operations, etc. This analysis can be slow and exhaustive, but the result gained is very detailed(Ding *et al.*, 2014). Moreover, it fails to detect threats with evolving capabilities such as metamorphic and polymorphic malware. Over and above that, it is incapable of defending against unknown or zero-day exploits (Thunga and Neelisetti, 2015), (Gandotra, Bansal and Sofat, 2016).

`

Table 2-2 Comparison of Static and Dynamic Analysis

Static		Dynamic	
Advantage	Disadvantage	Advantage	Disadvantage
• A straightforward analysis • The higher speed of analysis and detection thereby providing rapid classification	• Face obfuscation or in-memory mutation • Miss evasive malware and zero-day attacks • Require expert manual intervention • Not a fully automated solution	• Newly released forms of malware can be distinguished	• Verifying that a new file is malicious can be complex • Collection information requires a contained environment • Require expert manual intervention • The required period of time needed for observation is not clear hence is time exhaustive and resource consuming • Not a fully automated solution

On the other hand, a technique known as dynamic is monitored and collection information during the interaction of Operating System with a suspicious program at runtime. All requests to access specific files, processes, connections, or services will be analysed and collected. This collection can be a list of system calls, network access or memory modification (Rathnayaka and Jamdagni, 2017) (Xu et al., 2017) (Ghezelbigloo and Vafaeijahan, 2015). Depending on the methods used the behaviour of live malware is observed and learned in order to record every single step and intention at a real time. The complexity associated with this technique is to deal with well-designed metamorphic codes as they may behave differently in a control simulated environment in order to trick the analysts, resulting in the malicious verdict. Technically, the essential indicators of new threats will be cAPTsured if the malicious programs interact with real victim machine. It means to wait for potential harm to the system before it is identified as threats. Thus, the pitfalls would be losing harmless or useful files. Besides, the approaches based on this technique suffer from low false positives because they can easily be fooled once the behavioural analysis method is known (Shijo and Salim, 2015), (Afonso et al., 2015).

`

The effectiveness of the dynamic approach as compared to static one is doubtful because it is time and resource consuming, resulting hindering early detection(Islam *et al.*, 2013)(Fraley, 2016). To address the weaknesses of these, an approaches hybrid analysis is studied. This is a combination of static and dynamic analysis regardless of execution in order to enrich its analysis results using data flow tracking and instruction tracing(Shijo and Salim, 2015).

Authors of (Liu *et al.*, 2013) (Tong and Yan, 2017) (Narouei *et al.*, 2015), and (Gandotra, Bansal and Sofat, 2016) proposed the hybrid techniques which include features from both static and dynamic analysis of malware that can detect different types of malware effectively. They presented more accurate results as compared to the systems using features from static analysis or dynamic analysis alone.

(Tong and Yan, 2017) built up different pattern set of malicious and benign app based on their structure, profile, or through certain string patterns encoded in them. Although accurate, this method of analysis is expensive and slow. The above finding is consistent with the study by authors(Idrees *et al.*, 2017). However, their framework relies on the information access to vital hardware and software resources. As the authors (Tong and Yan, 2017) noted the drawback of the hybrid approach is applicable to their research approach. Due to the dynamic nature of attack profiles, the detection techniques fail to adjust to the temporal changes in real time. Specifically, for any change in the malicious pattern, the detection model will involve an updated profile with constant training. The quality of detection performance to judge new observation will wane drastically according to the model requirement. Therefore, the performance will often be exceeded resulting in a high rate of false positive alarms (Nauman, Azam, and Yao, 2016).

2.4 Machine Learning (ML)

The classification decisions or a result of prediction model depend on analysing the result of static or dynamic behavioural features (Alazab, 2015) (Milosevic, Dehghantanha and Choo, 2017).

In general, ML is associated with the Artificial Intelligence (AI) in which mimicking human (brain) learning and learning through the experience. It is frequently overlapping or somehow originated with data mining, and pattern recognition to automatically deal with known and unknown patterns. As analysing pattern is involved with feature extraction in which non-important features will be phased out. ML has been practically applied in many fields covering domains including speech recognition, text categorization, source code plagiarism, metamorphic virus detection, and many others. By outlining, ML approaches offer, the predictive quality can give malware analysts to discover the breaches and initiate an incident response.

In addition, ML is designed and oriented towards new malware acquisition. By considering the ongoing "arms race" between malware creators and malware detectors; attempts at practicing ML algorithms to cybersecurity began in the 2000s to conquer the need to be automated in handling detection potential malicious files (Scott, Fellow, and Technology, 2017). Needless to say, it is practically impossible to manually detect and analyse such a huge number of suspicious file daily.

Primarily, ML algorithms are grouped into three subgroups: supervised, unsupervised and semi-supervised. Table 1.3 presented aside-by-side comparison of these methods in agreement with (Kaur and Singh, 2014), (Ucci, Aniello and Baldoni, 2017)(Comar *et al.*, 2013)(Firdausi *et al.*, 2010)(Mohaisen, Alrawi and Mohaisen, 2015).

Several research proposals have been exhibited one or more detection techniques on the basis of robustness, efficiency, accuracy, computational overhead and noise tolerant parameter by applying ML methods (Ling, Putra and Ling, 2017)(Alam *et al.*, 2015)(Shijo and Salim, 2015). The comparison between them is indicated in Table 2-3.

Algorithm	Advantage	Disadvantage	Popular algorithms includes
Supervised	Work with known data	Require labelled data Misclassifications are all caused by very fuzzy data.	SVM Decision Trees Random Forest K-Nearest Neighbour
Unsupervised	Work with unlabelled data Work well with uncertain data	Time consuming	K-means Fuzzy C means (C-means) Gaussian means (G-means) Hieratical clustering
Semi-supervised	Worked with mixed labelled and unlabelled data		

Table 2-3Machine Learning Techniques

`

Setting a goal at the outset great greatly improves the chance of a successful ML application. When considering the use of ML within security viewpoints each of below goals would require different algorithms and different approaches:

- To have some malware samples and want to find malware families. In this way, it is possible to associate unknown samples to already known families, and by consequence provide an added-value information for further analyses(Mastjik and Varol, 2015)(Canfora *et al.*, 2015)(Ahmadi *et al.*, 2015)(Cheng, Tsai and Yang, 2013)(Bot, 2017)(Fan *et al.*, 2016)(Sharma and Sahay, 2016).
- Want to classify some network logs into malicious vs. benign to reduce the workload for human analysts(Zhao *et al.*, 2015)(Bhatt, Yano and Gustavsson, 2014).
- Want to make an early prediction on network traffic(Le, Zincir-Heywood and Heywood, 2017)(Bekerman *et al.*, 2015)(D. Liu *et al.*, 2014).

Malware analysis can be divided into two phases - malware detection and malware classification. Classification is a data mining function that assigns observations in a collection to target categories or classes. The goal of classification is to accurately predict the target class for each observation in the dataset. For instance, a classification model could be used to identify a suspicious file as malware or benign(Wang *et al.*, 2013)(Baldangombo, Jambaljav, and Horng, 2013)(Fraley, 2016)(Yewale and Singh, 2017). Commonly at the first step, an executable is detected if it is malicious, and if it contains a harmful part, then at the second step it is classified into some malware families for further research(Canfora *et al.*, 2015). However, in practice, a suspicious file may be categorized into a certain malware family first, which is considered to be an advantage in early malware detection(Ahmadi *et al.*, 2015). Rule evaluation measures can be used in discovery of high quality rules; thereby, classification can be made based on these unexpectedness or usefulness rules (Liangboonprakong and Sornil, 2013).

In an extension of approaches based on ML classification are fallen into two different evaluation; it is given in Table 2-4 (Comar *et al.*, 2013).

`

Table 2-4 Malware Evaluation

Malware Analysis	Further categorize the malicious flows into one of the pre-existing malware or new malware.	(Azab *et al.*, 2015)(Ahmadi *et al.*, 2015)(Ahmadi *et al.*, 2015)
	Used to isolate malicious flows from the non-malicious ones.	(Nissim *et al.*, 2014)(Cosovan, Benchea and Gavrilut, 2015)(Liu *et al.*, 2013)(Narouei *et al.*, 2015)

The first evaluation is based on single rules. Most existing detection frameworks are proposed based on this evaluation. They have been used to determine the stopping criteria for the rule generation and extract interesting rules for classification purpose. However, evaluation based on single rules might generate biased classification or overfitting results since it relies on user's judgements or domain specific(Ahmadi *et al.*, 2015)(Nissim *et al.*, 2014). Building classifier rule based on link analysis can fall into this category.

For the overall rule induction system performance, one also needs to consider the evaluation based on a rule set. The second evaluation is based on rule sets. It is more complicated than the other evaluation since there is more than one rule involved in making a decision. In this evolution, the relationships between rules in a set will analysis which is an essential step for this evaluation (Khazaee and Faez, 2014)(Nissim *et al.*, 2014).

2.4.1 Supervised Machine Learning

Supervised learning based approach is chosen to deal with known corpus but incomplete data. The classification accuracy depends on the features quality and the effectiveness of the classifier. Whereas clustering (unsupervised) chosen to deal with new inputs(Epishkina and Zapechnikov, 2016).

A good example of Supervised Learning is a classification for malicious file detection. Due to the diversity of malware classes, imbalanced distribution and data imperfection issues, malware classifier model are challenging. The classification algorithm would be supplied a large dataset of files as well as the labels 'malicious' or 'benign'(Nissim *et al.*, 2014). This is known as a training set; a dataset used to train the algorithm. The algorithm would produce a model which would correctly classify the training set(Nissim *et al.*, 2014). Any new, unlabelled files are then passed against the model and mapped to 'malicious' or 'benign', based on the trained model. Leaving a large fraction of samples unlabelled and hence delaying the signature distribution and abnormal behaviour in the system is accorded the greatest weakness of supervised method (Pai *et al.*, 2016). Hence, in line with the nature of the supervised algorithm, it difficult to detect new malware. Furthermore, poor performance on new and evolving malware is considered one of the limitations of this approach. Therefore, research into more robust detection mechanisms is warranted by unsupervised algorithms (Mohammadi and Hamzeh, 2017). While Vector Machine and Hidden Markov Model was largely explored (Gharacheh *et al.*, 2016)

Therefore, a security solution must be put in place to ensure that such attacks with unknown patterns will be captured. This motivated academics and researchers have been employed unsupervised learning to address these challenges and identify flows of existing and novel malwares(Comar *et al.*, 2013)(Wei, Sprague, and Warner, 2009)(Lin, Jiang and Lee, 2014)(Bot, 2017).

2.4.2 Unsupervised Machine Learning

Clustering algorithms are deployed in unsupervised learning problem. The process of organizing data points into groups whose members are similar in some way is defined as a clustering approach. Thereby, a cluster is a collection of unlabelled data which are similar between them and are dissimilar to the data belonging to other clusters. Empirically, regardless of its similarity measurement, the effectiveness of this approach is affected heavily by extracted key features. For each pair of points that share at least one cluster in the overlapping clustering results. Therefore, the only thing which potentially tricky is that a given point may appear in multiple clusters(Khazaee and Faez, 2014).

In the overall picture, some advanced malwares are able to overcome the traditional methods based on supervised machine learning. Though, detecting advanced malware is a challenge and requires and innovative learning approach. Recent research has depicted that clustering as a persuasive option for malware detection (Comar *et al.*, 2013) (Wei, Sprague, and Warner, 2009) (Bot, 2017)(Wang *et al.*, 2013)(Pandeeswari and Kumar, 2016)(Liu *et al.*, 2017). This type of unsupervised ML approach can be further divided into two categories: Hard-Clustering and Soft-Clustering (Mohammed, Mohammed, and Saraee, 2016).

Hard Clustering: In hard clustering, each data point either belongs to a cluster completely or not. For example, the malicious file is put into one group out of the 10 groups.

Soft Clustering: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the research scenario, each APT group is assigned a probability to be in either of five clusters of the families. Furthermore, this method does not require prior information to learn from the data.

As indicated, limitation of hard clustering forms ill boundaries that leads into the data space often overlapping the perimeters of the surrounding cluster. The inherent limitation of hard clustering persuaded researchers to apply soft clustering to fulfil the research objectives.

Table 2-5 Comparison of Clustering Techniques

Type	Reference	Limitation	Advantage	Sample
Hard/Crisp	(Zolotukhin and Hämäläinen, 2014)(Narra *et al.*, 2016)(Mohammadi and Hamzeh, 2017)(Sarvani, 2017)	High probability of ill-defined boundaries No capable of dealing with over-lapping No capable of handling extreme outlier points	Work well with highly structure data Provides efficient results in case of large datasets Easy to implement	k-means
Soft /Fuzzy	(Huang *et al.*, 2012)(Khazaee and Faez, 2014)(Khazaee and Sharifi Rad, 2013)	Time consuming The probability of looping in local optimum largely depends on initialization values Highly dimensionality	Work well with vagueness, uncertainty, loosely structured data Flexible and robust toward new data point Improve the detection over the time	C-means

`

The current lack of automatic and labelling of a large number of complicated malware samples led to a possible solution to automatically cluster malware sample. Additionally, efficient process this huge influx of new malware samples and accurately label them is another theme of concern. Furthermore, avoid analysing samples that have already been analysed and label new incoming samples by soft clustering rather than crisp clustering to avoid producing sharp partitioning is providing a scope for further investigation. Invariably, using a new form of a fuzzy-based clustering algorithm for grouping close malicious code together in the same cluster will repetitive analyses(Khazaee and Sharifi Rad, 2013)(Bernardi *et al.*, 2017).C-Means clustering is a well-known approach and works very efficiently for most of the cases. Fuzzy C Means clustering is based on fuzzy logic and is used when data is to be divided through fuzzy partitions. Fuzzy clustering, offers further advantages in a way that an instance may belong to more than one cluster as the membership degrees is between zero and one(Shalaginov, Grini and Franke, 2016)(Huang *et al.*, 2013)(Pandeeswari and Kumar, 2016)(Shalaginov, 2017). However, Fuzzy-C means will tend to run slower than K means, since it's actually doing more work. Each point is evaluated with each cluster, and more operations are involved in each evaluation. K-Means just needs to do a distance calculation, whereas fuzzy c means needs to do a full inverse-distance weighting(Narra *et al.*, 2016).

In hierarchical clustering, smaller clusters can be combined and larger clusters can be divided into smaller clusters. This hierarchy of clusters is also called dendrogram which displays a tree diagram.There are two types of approaches, top down and bottom up. In top down approach, larger clusters get divided in to smaller clusters, where as in bottom up approach small clusters are merged into larger ones(Sarvani, 2017).

2.5 Data Mining

Data mining is the core of knowledge discover process. Data is mined to identify associations and also to anticipate behaviour pattern. Data mining evolved from simple taking out of raw data to an analytical process from a large amount of data in order to discover knowledge, see Figure 2-1. The process of knowledge discovery includes: selecting, exploring, modifying, modelling and rating large of amounts of data. In general, the benefits of data mining come from the ability to uncover hidden patterns and relationships in data that can be used to make predictions that impact prediction or classification(Mohammed, Mohammed, and Saraee, 2016). What information is mined depend on the specific objective to achieve? It assists automation with important keywords, strong features as input for ML algorithms(Epishkina and Zapechnikov, 2016). By that it has been extensively applied as a suitable framework in many areas such from modelling to predict weather pattern, recognize imagine especially at decision-making level (Bist and Campus, 2014)(Mendel, John, and Liu, 2006).

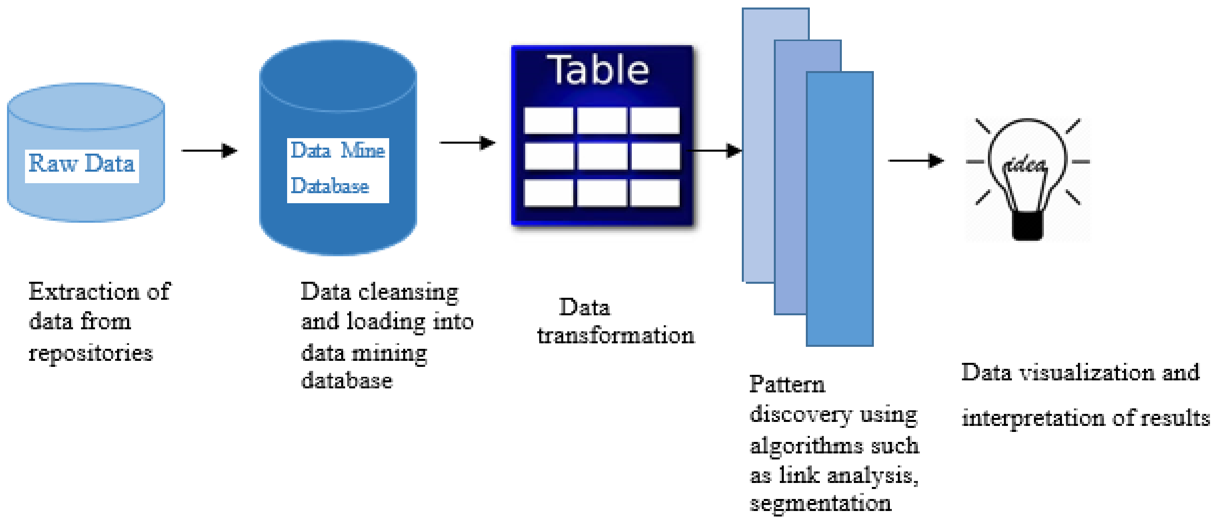

Raw Data
Data Mine Database
Table
idea
Extraction of data from repositories
Data cleansing and loading into data mining database
Data transformation
Pattern discovery using algorithms such as link analysis, segmentation
Data visualization and interpretation of results

Figure 2-1Data Mining Process

Previous works have shown that data mining is the process of studying sample files with the aim of acquiring knowledge about its pattern to spot each instant (malware) via pattern matching techniques is a promising approach (Siddequi, Wang and Lee, 2008)(Azab *et al.*, 2015)(Homayoun *et al.*, 2017)(Islamic and Minna, no date). Solutions to this problem will facilitate initial triage of new samples into existing families or as potential new families. As (Zolotukhin and Hämäläinen, 2014) have also indicated, this study is improve the efficiency of the detection process. Besides, it has been incorporated into methods of classification and clustering to empower handling levels of uncertainty in real-world complex problems such as malware detection (Souri and Hosseini, 2018). Moreover, this approach is broadly used in different types: pattern recognition, fault detection, medical diagnosis reasoning, etc.

As claimed by (Ye, 2017)(Azab *et al.*, 2015), data mining is a convincing method for malware detection. In the current research, data are a list of the opcode sequence from each malware family. Hence, the discovered knowledge will be opcode patterns, associations, or relationships among all opcode in one family. The process of pattern detection and understanding logical relationships accrue after standardization of raw dataset. The better clustering segmentation wold appeared when the data mining technique does not tolerate minor difference of code structure (Wang, 2014). In order to extract some rules to evaluate pattern matching process association rule mining is used in this study.

Schultz et al. (Schultz *et al.*, 2001) were the first to present the idea of data mining for malware detection. They use the static features such as strings, byte sequence and features from PE and used the Naïve Bayes method for classification purpose. Their results were upgraded by (Zolotukhin and Hämäläinen, 2014)who make use of n-gram and various classification methods to detect new malicious. Detection system using iterative SVM clustering to provide the best accuracy.

2.5.1 Association Rule in Data Mining

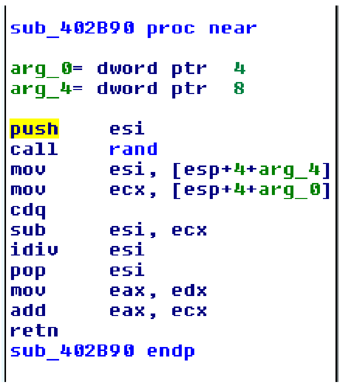
Association rule mining is the data mining process of finding the rules that may govern associations, correlations, or causal structures from data sets. It can be done in both scenarios, supervised and unsupervised learning. However, it is not a classification technique, it is used for frequent pattern and sequence pattern or a combination of both to obtain data model. Initially, association rule mining was used in unsupervised scenarios to discover interesting or novel patterns. It is originally designed to work with discrete (categorical) data(Epishkina and Zapechnikov, 2016). Link Analysis is chosen to apply among the many different techniques used for data mining such as sequential pattern mining or frequent mining. Link analysis is based on a branch of mathematics called graph theory, which represents relationships between different objects as edges in a graph. This relation-oriented methodology displays nodes and links based on their weighted values; node size and link width represent the relative size of the node and link values. Scores the nodes and edges, therefore, the edge thickness representing the strength of association and differing node sizes representing qualitative properties of the node. In general the higher the weight, the higher the contribution to the influence metric. For instance, one of the macros calculates influence centrality in addition to degree centrality(Y. Liu *et al.*, 2014).SAS® software packages provide the link analysis node and Profile Segment node inside Enterprise Miner™ 14.3. SAS® Enterprise Miner™ is a state of the art predictive analytic and data mining software package that enables organisations to analyse complex data, find useful insights and act confidentially to make fact based decisions. It contains the link analysis macros which is easy to generate and interpret.

This associative graph has been successfully adopted for various information and social networks (Thompson, 2008).This sort of analysis aims to discover patterns of activity that can be used to derive useful conclusions and business information about a subject for example malware sample. Give such insight, this research presents a novel feature construction model based on link analysis, which is effective to derived shared similarity in malware family instructions.

The research is equipped with Link Analysis to discover unknown opcodes links patterns. As the name suggests, this technique would help analyse the association rules between two or more items in large data sets. In the current scenario, Link Analysis has demonstrated the similarity of opcode fragments the similarity of two malicious based on measuring which is generated by Link Analysis macros. The formula for similarity comparison of code sequences is deduced through association rule mining analysis. Based upon the generated values ,the probability of each malware group belong to is calculated (Ban *et al.*, 2015).

2.5.2 Mining Opcode Relevance

Opcode (Operational Code) are machine language instruction that performs CPU operations on operands such as logical, arithmetic, data manipulation and program control(Zolotukhin and Hämäläinen, 2014). Part of source code is visible in Figure 2-2.Identifying similar or identical code fragment among program is valuable in some applications such as, code theft detection. As malware program contains the various sequence of opcode by usage of obfuscated techniques to hinder detection procedure(Gharacheh *et al.*, 2016). Code obfuscated methods are often employed by malicious intention to lessen the effectiveness of the detection system. This mechanism is achieved by dead code insertion, variable renaming, statement reordering, expression reshaping, break transformation or join transformation and so on. Thereby, malware analysis mechanism requires the knowledge of source code or structure of the program to determine the monopoly of functions. To find inherent regularities in the opcode, algorithms of association discovery are employed. Different type pf association mining algorithms such as frequent , sequential patterns, cyclic association, cluster analysis introduced in malware analysis by (Mooney and Roddick, 2013)(O'Kane, Sezer and McLaughlin,2014)(Zolotukhin and Hämäläinen, 2014)(Liangboonprakong and Sornil, 2013)(Mastjik and Varol, 2015)(Yewale and Singh, 2017)(Ding *et al.*, 2014).

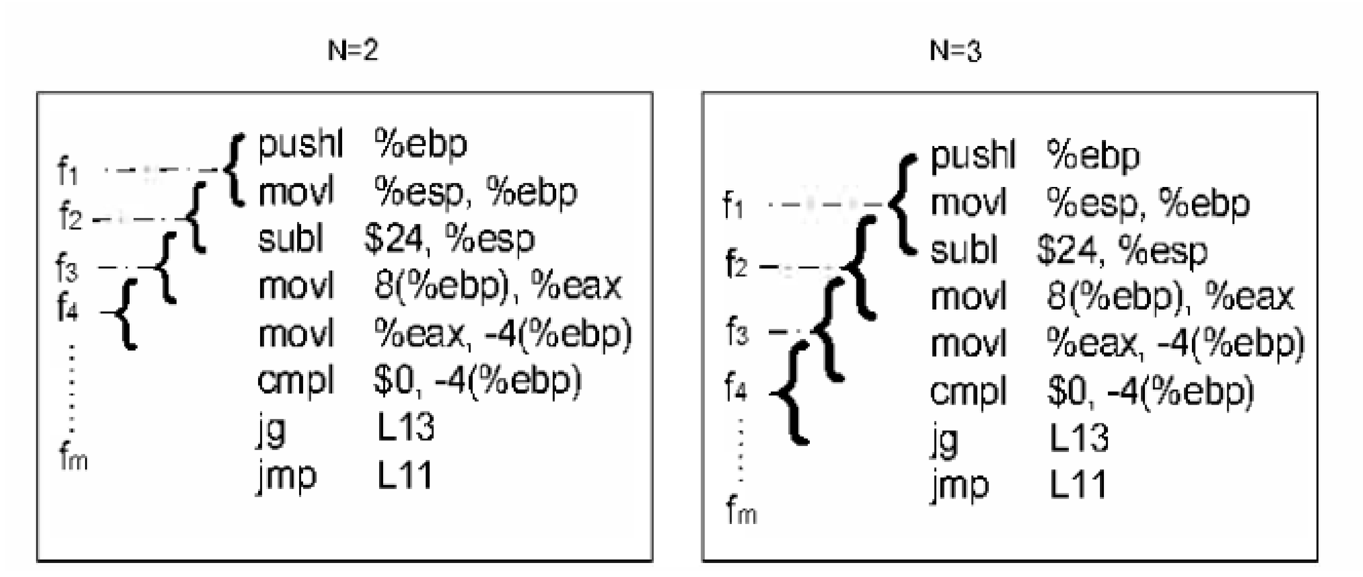


```
sub_402B90 proc near

arg_0= dword ptr  4
arg_4= dword ptr  8

push    esi
call    rand
mov     esi, [esp+4+arg_4]
mov     ecx, [esp+4+arg_0]
cdq
sub     esi, ecx
idiv    esi
pop     esi
mov     eax, edx
add     eax, ecx
retn
sub_402B90 endp
```

Figure 2-2 Assembly code example generated by IDA-pro

`

A very first framework based on N-gram and K-Nearest was presented by Abou-assaleh (2004) in the n-gram method, based upon some length sequences. The model requires substrings of a larger string with a length, for example, the string “MALWARE”, can be segmented into several 4-grams: “MALW”, “ALWA”. “LWAR”, ”WARE” and so on. Once the set is chosen, n-grams for every file extracted in that set that acts as the file signature. Figure 2-3 is illustrated N-gram 2 and 3. As authors (O’Kane, Sezer and McLaughlin, 2014) and (Liangboonprakong and Sornil, 2013) the detection system can classify any unknown instance as malware or benign software based on N-gram concept. The assignment of a new executable file was based on the most similarity of the representative profile of each malware and clean in the dataset. However, as Fraley (2016) highlighted this technique practically useless for polymorphic and metamorphic malware samples.



```
pushl   %ebp
movl    %esp, %ebp
subl    $24, %esp
movl    8(%ebp), %eax
movl    %eax, -4(%ebp)
cmpl    $0, -4(%ebp)
jg      L13
jmp     L11
```

```
pushl   %ebp
movl    %esp, %ebp
subl    $24, %esp
movl    8(%ebp), %eax
movl    %eax, -4(%ebp)
cmpl    $0, -4(%ebp)
jg      L13
jmp     L11
```

Figure 2-3 Illustration of N-gram Structure (O’Kane, Sezer and McLaughlin, 2014)

Over the past has seen the different models by observing opcode pattern to detect and eliminate new unknown malware in a prompt manner(Wang *et al.*, 2016)(Runwal, Low and Stamp, 2012)(Canfora *et al.*, 2015) (Ding *et al.*, 2014)(Santos *et al.*, 2013)(Santos *et al.*, no date)(Yewale and Singh, 2017)(Sharma and Sahay, 2016)(O’Kane, Sezer and McLaughlin, 2014)(Hu and Tan, 2017). The methodology for malware categorization by implementation concepts from text categorization is used by (Shabtai *et al.*, 2012)(Suarez-tangil *et al.,* 2014). By considering the fact that, in real time setting, patterns within the words change over time just like the malware characteristic(Nissim *et al.*, 2014). However the drawback of supervised models is training sets should be designed accordingly otherwise misclassification will appear(Nissim *et al.*, 2014)(Zolotukhin and Hämäläinen, 2014)(Gandotra, Bansal and Sofat, 2016).

For this reason, the authors (Suarez-tangil *et al.*, 2014) observed and measured the similarity between malware samples by using clustering techniques in which it succeeded in dealing with supervised learning problems.

2.6 Analysis to Detect Malware

The amount of unknown malware is growing at a staggering rate. Computer users often download malware to their computer by unknowingly visiting a malicious web-page hosting a drive-by download attack, by clicking on a malicious link included in an email, or by inserting a USB thumb drive containing malware into their computer. Once the malware gets to the organization there will be a breach in private data, disruption in operations or destruction in infrastructure by using a wide range of tactics and tools(Security and Report, 2017). The sheer number and variety of known and unknown attack strategy is part of the reason why detection cyber-attack is a difficult problem.

The method for indication of an attack and classification mainly fall into two broad categories: Signature-based, Behaviour/anomaly-based and Heuristic/specification (Deka, Sarma and Panicker, 2017)(Gharacheh *et al.*, 2016)(Bist and Campus, 2014)(Santos *et al.*, no date; Zhang *et al.*, 2011)(Kaur and Singh, 2014) (Damshenas, Dehghantanha and Mahmoud, 2013) (Pai *et al.*, 2016) (Scott, Fellow and Technology, 2017)(Louk *et al.*, 2014)(Nauman, Azam and Yao, 2016)(Nissim *et al.*, 2014)(Hassani and Zarei, 2015) (Baldangombo, Jambaljav and Horng, 2013). The advantages and disadvantage of each method are summarized in Table 2-6 and different methods belong to each category are presented in Figure 2-5.

Table 2-6Malware Detection Methods

Methods	**Pros.**	**Cons.**
Signature-based	Effective and efficient against a common type of viruses The low percentage of false alarm The high speed of detection	Fail to detect unknown malware and undocumented attacks Fail to detect crafted malware such as polymorphic and metamorphic The high amount of manpower and time

Behaviour-based	Polymorphic/Zero-day malwares can be detected. Obtaining comprehensive information about the malware Identifying what the malware does in a specific environment when files are opened Detecting an individual instance of malware targeted at a person or organization	False positive remain an issue The high amount of scanning time Require safe environment like a virtual sandbox Fail to detect curtailing malicious activities Restriction of a Cloud-based solution
Hybrid	Depends on the feature extraction techniques the accuracy of malware detection effectively is improved.	Depends on the feature extraction techniques there is time complexity of training or evaluation phase

Signature-based detectors compare signatures (using MD5 or other hashes) of files into a predefined database of known malicious files. Signatures are created by examining the internal components of an object. The object could be data which collected from static or dynamic analysis. The good and unknown file is categorized in one group and malware and its. The similarity between two objects is measured by counting the number of common blocks. If they match, the file is treated as a 'threat' (Yewale and Singh, 2017).

At this point, the challenge can be seen when it comes to the detection of a new variation of known malware families. On the other hand, many hackers have begun using polymorphic or encrypted code segments which are very difficult to create a signature for. Hence, malware authors can make signature detection redundant(Alazab, 2015). In response, observation of a particular pattern or abnormal behaviour was born as behaviour-based scanning(Ding *et al.*, 2014). For example, malware is better able to watch when it has been placed in a virtual environment such as a sandbox. Yet, detecting malware using behavioural analysis involves heavily instrumenting the operating system. Besides, it is time-consuming to observe programs as they run for suspicious or malicious behaviours in order to stop them(Mohammed, Mohammed, and Saraee, 2016). Over above that, many behaviour solutions are exclusively cloud-based which may be an issue for some organizations. Putting it all together the scheme of each approach is illustrated in Figure 2-4 based on gain knowledge of present works by (Wang *et al.*, 2013)(Wang and Wang, 2015)(Hu and Tan, 2017)(Alazab, 2015).

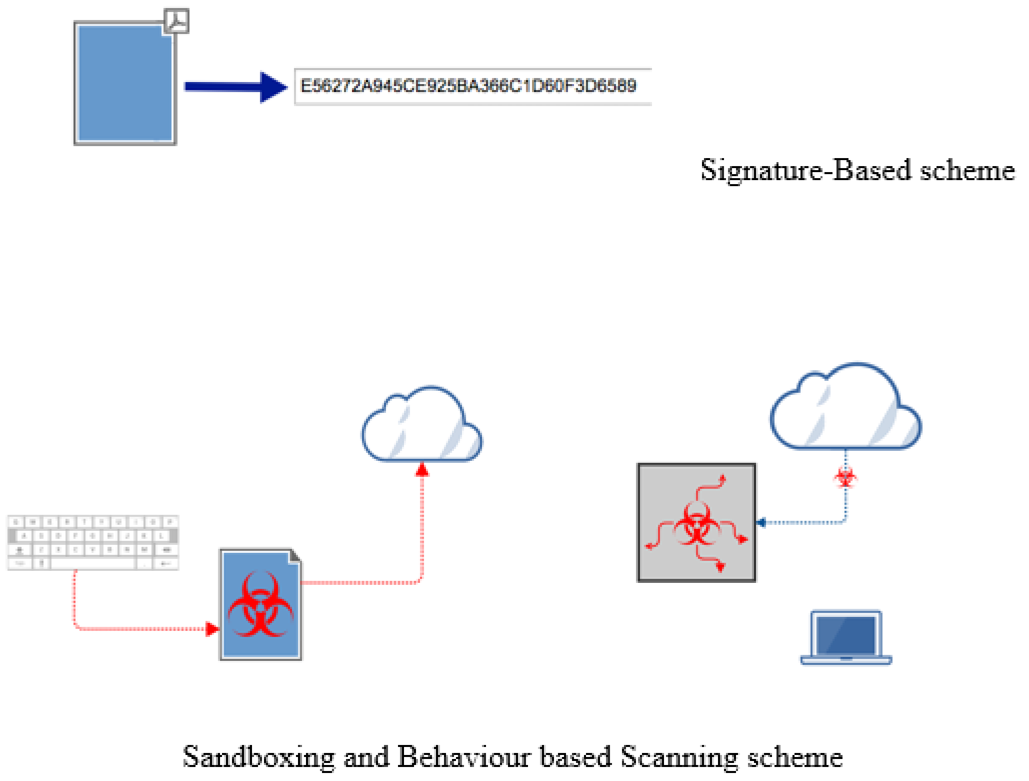

E56272A945CE925BA366C1D60F3D6589
Signature-Based scheme
Sandboxing and Behaviour based Scanning scheme

Figure 2-4 Malware Analysis Scheme

Some malware detects a specific registry key related to a virtual environment, allowing the threat to evade an automatic sandbox as well as an analyst attempting to dynamically run the suspected malware binary in a virtual machine(Mehra and Pandey, 2016). Both signature and behaviour-based malware detection are important and have advantages. These methodologies, while valid, waste resources and are imprecise because they assume that neither the malware nor its behaviour will mutate. Theoretically, behaviour and signature-based approaches hold some promise, but as pure technologies, they fail; as result, usage of these methods is debatable to detect or predict (Das *et al.*, 2016). In response, a hybrid analysis, specification-based, is introduced by utilizing both techniques. The scheme of each technique is illustrated in Figure 2-4.

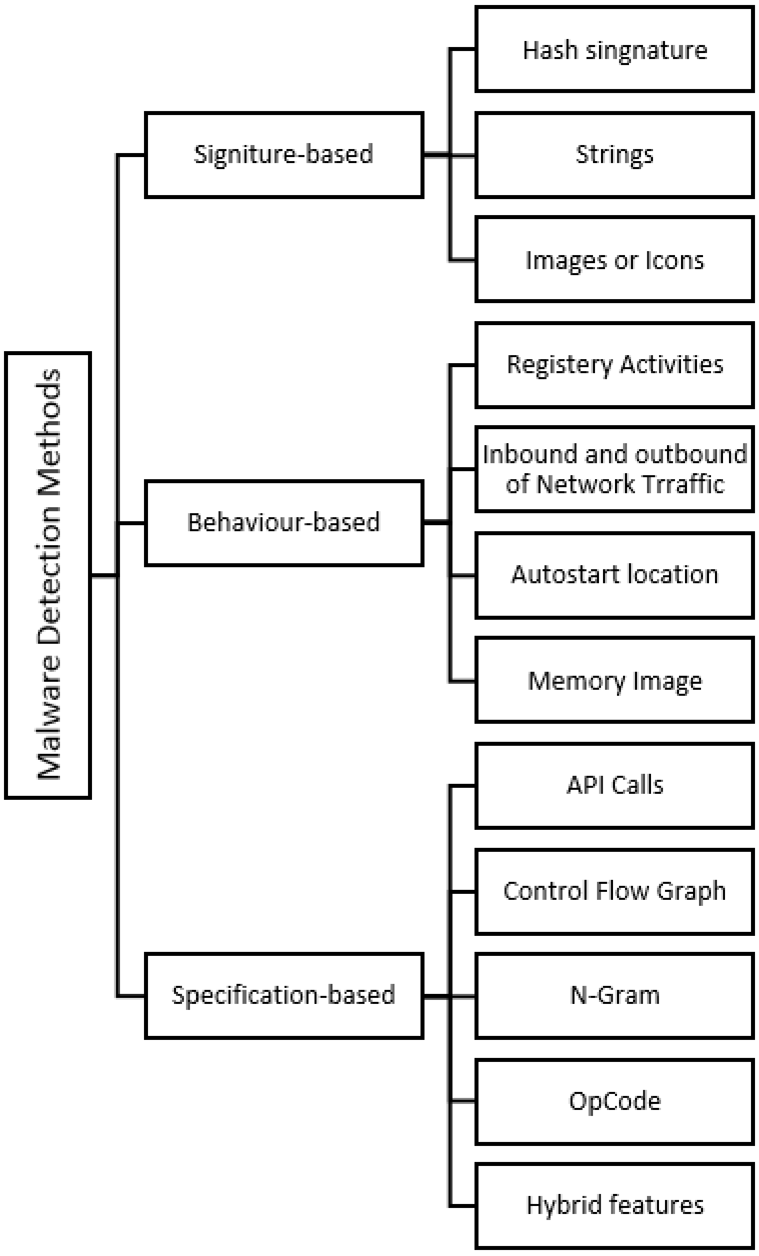


Figure 2-5 Taxonoly of Malware Detection Techniques

The authors(Yewale and Singh, 2017) in explained that features analysis conducted by analysing bytes, binary, and disassembly. This type of analysis measured certain patterns in the malware and was exemplified by(O’Kane, Sezer and McLaughlin, 2014), and(Ekhtoom *et al.*, 2016).

2.7 Techniques over Malware Detection

The ability to detect both known and unknown or zero-day threats provides a huge benefit for businesses that simply cannot afford the potential downtimes, data breaches, and remediation costs resulting from mitigated malware attacks.

In the authority of Internet Security Threat Report released by Symantec in 2016 (Security and Report, 2017), over more than 357 million new variants were observed in 2016. Thus, signature-based detection is not scalable when there are more than hundreds of new signatures every day, let alone when there are hundreds of thousands. With the daily creation of nearly one million unseen and newly establish attacks, the existing anti-malware systems which are based on signature and behavioural mechanisms cannot practically challenge this issue (Gandotra, Bansal and Sofat, 2016). In order to solve this issue, the utilization of machine learning aided has been effectively employed along with the integration of static and dynamic approaches.

The characteristic of obfuscated viruses poses a great challenge to the security professionals leading to the deployment of integration of ML techniques(Comar *et al.*, 2013)(Mohammadi and Hamzeh, 2017)(Bot, 2017)(Alazab, 2015). Malware is grouped into a malware family according to its particular functionality. Nevertheless, the attacks of same malware families are very similar in terms of file content and characteristics(Liangboonprakong and Sornil, 2013).

Suitably, features analysis key argument is that no matter what code is being executed, the end result is still the same. Subsequently, with grouping similar sophisticated malware attacks together, discover a new group of threats will accrue.

From the perspective of a growing number of unique malware samples, it is important to group similar malware variants for two main reasons. First relates to human resources: malware analysts do not have to analyse every single variant, just some representative samples(Nissim *et al.*, 2014). The second reason it is linked to the fact that grouping variants from the same family could help anti-malware companies to add generic detection for that family(Liangboonprakong and Sornil, 2013)(Canfora *et al.*, 2015)(Milosevic, Dehghantanha and Choo, 2017). Bilar (2007) for first time lesser frequent opcode could be used to identify polymorphic and metaphoric malware. He demonstrated the malicious code has a lower basic block count, implying a simpler structure. Santos *et al.* (2013) computed the cosine similarity between opcode sequence length n of 2 variant and the other variants of that specific malware family each malware and its set of variants. This model mined the importance of each opcode and evaluated the repetition of opcode group to discover obscure malware families. This approach is consistent with literature (Hu *et al.*, 2013). Code instruction in variants malware type may tend to change; however, distinct frequent sequential patterns and its association may homogeneous in a malware family(Hu *et al.*, 2013).

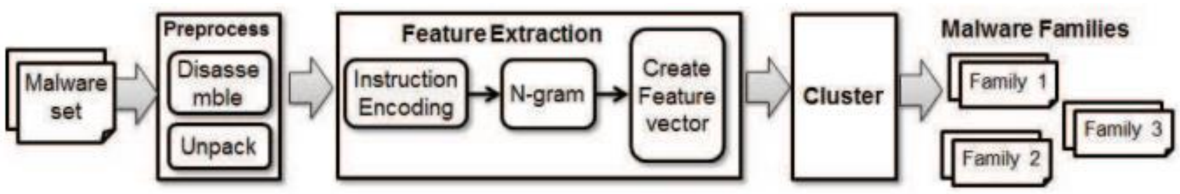


Figure 2-6 Overview of the proposed system by(Hu *et al.*, 2013)

In later papers, reduction in the effort of labelling that is required for the training phase and highlights the issue of unpacking malware by (Rathnayaka and Jamdagni, 2017) and also (Zolotukhin and Hämäläinen, 2014). The authors (Milosevic, Dehghantanha and Choo, 2017) tested several learning methods and proved the malware can be detected with a high degree of accuracy using opcode-sequence by supervised learning. Philip(O'Kane, Sezer and McLaughlin, 2014) used SVM classification to determine the ability of N-gram analysis. Their study presented dynamic N-gram analysis for N<= 4 clearly spot an opportunity for detecting obfuscated malware.

Malware analysis has been investigated extensively, however, shortcomings still exist for advanced malicious software which is specifically crafted for such target sectors: Healthcare, Defence, Aerospace, Government, Media, and Universities(Shenwen, Yingbo and Xiongjie, 2015). What follows describes these methods.

Table 2-7 Comparison of opcode in Malware Detection

File representation	**Reference**	**Techniques**	**Merits**	**Demerits**
Opcode sequence	(O'Kane, Sezer and McLaughlin, 2014)	k-clustering centroid mean	Improve detection with a classifier	Not metamorphic, Runtime performance unclear

Opcode frequency	(Yewale and Singh, 2017)(Chandran, Hrudya and Poornachandran, 2015)	SVM, Random Forst, BOOST and Decision Tree	Random Forest provided better accuracy and zero per cent false positive ratio.	Classify malware only into two categories: goodware or malware
The frequency of Opcode sequence	(Santos *et al.*, no date)	Computation the mutual information Static-based	Able to detect the most number of malware variants whilst the number of false positives is kept to 0 Able to distinguish benign executables	The longer sequence didn't use as features An experiment performed with a small malware dataset(1319 malware and 13000 benign)

(Zhang *et al.*, 2011) explored that a dynamic system relies on fuzzy inference based on hidden malware behaviour. Although the proposed technique produces a retail model of malware; change of input data characteristics require re-training of the whole model. This is a critical issue when malware detection requires a fast update of the decision rules while adding new characteristics. The developed rules should be tested, modified and further tested and modified again in order to reduce false negatives and false positives. Anyhow, false positives are unavoidable but they can be kept low(Mohammadi and Hamzeh, 2017).

In the next subsection, firstly classification of malware families discussed. Next the use of clustering and data mining argued.

2.7.1 Classification of Malware

The ML model is considered more practical which the trained model can be used to detect similar data. In other words, data can be scored against a trained model, with higher scores indicating a higher degree of similarity to the training data. However, malware functionality it remains the same while its structure is changing. The metamorphic generator presents malware files look like benign files. Therefore, apart from an indication of malicious code from benign; identification of malware families (multiple classes) is another way to allow malware analysts to uncover zero-day threats(Ling, Putra and Ling, 2017).

A common single classification method designs either by utilizing the collection of static or dynamic features of a malware. However, the current trend in the identification of malicious code is to integrate both techniques to prevent malware users to evade defences. Islam et al (Islam *et al.*, 2013) tested the first classification method which equipped with combination of static and dynamic information into a single training test. Based on their finding, Random Forest ML technique performed the best accuracy to classify the latest malicious code.

Motivated by the Hidden Markov Model (HMM) based classification system, (Thunga and Neelisetti, 2015) were able to identify the family of metaphoric malware files. The authors trained and tested multiple HMM's to measure score of a family virus-based opcode sequence similarity. The proposed technique gave an accuracy of about 90% in each virus samples. In addition, the author concluded computation based on distance to centroids in K –means provided considerable result in terms of precision and overall accuracy. In their method first, extract features from it by HMM models. Similarly, Gharacheh *et al.* (2016) tested the idea of combining different approaches to increase the efficiency in terms of time by using trained HMM in two layers based on the important sequence of the opcode. These techniques are aimed at deriving a model that captures the similarity of a given metamorphic family. Yet, the important factor of malware group similarity of viruses is not considered (Rezaei and Afraz, 2016).

Similarity and metamorphic detection Software similarity is a potentially helpful recommendation for detecting metamorphic malware. In this regard, malware creators commonly redistribute the original malware code to increase the likelihood of evading current recognition mechanisms (Ucci, Aniello and Baldoni, 2017)(Luo, Ming and Wu, 2017).

The appearance of malware is different even in one family which making it difficult to identify a common feature of each group. Moreover, it is ambiguous to what extent it is essential to retrain the classifier with new files. Thereby, a feature selection method which improves the classification of advanced malware families could be extended to unsupervised learning approach.

2.7.2 Clustering of Malware

One of the most popular approaches to data mining is clustering methods. In recent years, various research has introduced new thresholding techniques based on data mining or subsets to control the noises of data like malicious codes. Due to the complexity and high volumes of malware on daily basis usage of data mining is increased in the area of cybersecurity publication(Fraley, 2016) (Gandotra, Bansal and Sofat, 2016) (Ye, 2017) (Souri and Hosseini, 2018).

Since uncertainty is an intrinsic characteristic of a malicious program; the role of fuzzy methods will help to filter virus threats and also fight with a zero-day virus attack. Recently, the study of Neuro Fuzzy (NF) has been reported by (Shalaginov, 2017)as a mechanism for proactive malware detection. This method is covered limitations of methods in which require retraining of the whole model as authors argued in. The proposed is upgraded their previous work in a way that reduces the rate of malware misdiagnosis from VM and decreases inaccurate analysis results by malware detectors. Three of the most well-known methods for grouping observations in a cluster analysis are a single linkage, complete linkage, and average linkage. The choice of distance measures is very important, as it has a strong influence on the clustering results. For most common clustering software, the default distance measure is the Euclidean distance.

2.8 Dealing with Advanced Persistent Threats (APT)

Dealing with the exponential growth of APT attack together with evading techniques is a serious concern. Since the attackers behind the APT want both convenient and deniability. APT attack detection techniques are evolving and facing open issues. On the report of FireEye, APT attackers receive direction and support from an established nation state. Whether their mission is to steal data, disrupt operations or destroy infrastructure, these threat actors tenaciously pursue their goal using a wide range of tools and tactics. To detect malware link to still, the security team should pay close attention when their security tools detect malware linked to previous APT attacks. APT ecosystem is a group of advance malware families that work together to perform the same objective. A malware family is a collection of malware in which each sample shares a significant amount of code with all of the others. Perhaps the simplest and most typical ecosystem is a dropper and a backdoor that is used tighter. They may not share the same code structure, but they are related because one drops and installs the other. Therefore, detect APTs is a challenge and requires and hybrid approach to detect and remediate.

2.8.1 Common Techniques to Detect Advanced Persistent Threats (APT)

An APTS is typically a more sophisticated attack in which an unauthorized user gains access to a system or network and remains there for an extended period of time without being detected. The current lack of automatic and speedy labelling of a large number of unknown APTs samples seen every day leading to a low detection rate of new malware samples in the wild. As the number of clusters is ambiguous and it is not easy to mention beforehand according to have prior knowledge of the dataset. Hence K-Means clustering as a hard clustering algorithm is not suitable for this particular problem (Wang and Zhang, 2007). Conventionally, intrusion detection systems have been categorized as signature-based or anomaly based.

The authors (Shenwen, Yingbo and Xiongjie, 2015) reviewed the available literatures on APTs attack detection and introduced an architecture based on big data processing. The proposed detection system performs trace back and predicate APTs attack, as well as increase the APTs, attack warning. Until date, these issues have not been answered properly.

While opcode is considered an efficient feature for malware detection, there has been limited attempt to use opcode mining for APTs malware classification. This could be, perhaps, due to the lack of suitable research datasets and the difficulties in collecting APTs samples. In addition, using machine learning for robust malware detection in APTs appear to be another understudied topic. Hence, in the paper, we seek to contribute this gap by exploring the potential of using Opcodes as features for APTs classification with unsupervised learning. Accordingly, ML approaches have capabilities to discover and predict data breaches attributed by obscure malware.

2.9 Summary and Remarks

There are a few observation that can be made about the growing trend in advanced malware detection:

a. The traditional signature-based are purely based on signature have zero resilience against new attacks of self-modification malware variants at runtime. Hence, this type of defence approach lacks the ability to specify a new threat. That is why usage specification-based take into consideration to address this deficiency.
b. An issue with the behavioural-based methods to evaluating whether the suspicious behaviour on the system, is that the information about a sequence of an event like API calls is insufficient. Moreover, the complexity of the approach makes the system works slower. More importantly, this approach requires the involvement of security expertise to determine a program is malicious which is resource consuming and also time exhaustive.
c. Specification-based approaches to developing an ML malware detector. For this reason, detectors which are entirely based on heuristic have a very low resilience against new attacks of metamorphic malware families.
d. Malware analysis approach is a potential method for attaining a complete understanding of new malicious behaviour. The integration of both static and dynamic analysis is used conduct to high

`

performance and time-effective malware systems. Still, the main problem with this system is the high false positive and false negative rate. Nevertheless, the process of developing an automated classification model which are based on features obtained from static and dynamic analysis frequently is time to consume with high accuracy due to the large feature set.

e. Signature based methods must be along with another approach that can able to detect unknown malwares. The machine learning is a suitable approach to complement classical signature based malware detection system.
f. Malware detection is similar to other related fields, such as software plagiarism and text categorization techniques. In the analogy to text categorization, using words or sequences of words as features is analogous to using the opcode group.
g. It does not appear to be any previous work involving similarity techniques or analysis for APTs classification.
h. Data mining is a close relative. It focuses on using machine learning, pattern recognition and statistics to discover patterns in data. Besides, clustering would fall into the machine learning/pattern recognition realm.

Summarizing briefly, malicious code detection, which is considered as a sort of pattern recognition could be fairly solved by integration of both data mining and clustering algorithm. Data mining and clustering are complementary technologies. This combination method will articulate both advantages of the two, from mining relevance features, learning abilities, optimization abilities, connectionist structure to clustering. Sharing of structural information of malware types based on attributes and hidden links is consider a method to detect known and unknown malware.

Chapter Three

Methodology

3 Overview

As discussed earlier, the goal of this work is to assess the capability of using link analysis concept as rule mining with the integration of unsupervised machine learning approach in malware family classification. The hypothesis is shaped in a way that this work is given sufficient static analysis, the clustering algorithms may form well-separated malware sample to the respective family. The basis for the proposed techniques and the underlying research methodology are presented next.

3.1 Fundamental Techniques and the Proposed Models

In the section, the techniques, tool, and languages are inputted in this research is covered. Next, clustering techniques are discussed. Followed by dealing data for ML. Since each programming language has its own advantages for each specific task one language hasn't been chosen for the entire project but it always uses the one with lowest implementing costs for each given task. The research idea is driven by (Rathnayaka and Jamdagni, 2017), visible in Figure 3-1.

The proposed framework consists of three major phases (A, B, and C), namely: Feature Extraction, Feature Selection, and Selected Algorithm Assessment. To establish each phase from A to C the necessary steps are show cased in details in each phase. The idea behind the approach is structured by the proposed approach in (Hu *et al.*, 2013).Phase A is accomplished by a mixture of Python, MATLAB, and Excel scripts for feature extraction purpose. All the scripts are written from scratch. As stated, phase B practiced in SAS® Enterprise Miner™ to construct the opcodes graph to demonstrate the feature construction process. The multiple results of link analysis technique formed into a single dataset to identify factors that differentiate data segments from the population. Further exploration and profiling of the segmentation results will reveal how effective this method is at identifying segments. For this reason, Phase C is shaped to do the process of partitioning task. Data clustering is adopted to generate valuable features for future data-driven decision making. Finally, results of phase C is verified in the same phase.

The task is to develop the best mechanism for classifying malware files a given dataset into groups. Phase A and B will be outlined in details in section 3.7. Phase C will be described in 3.8.

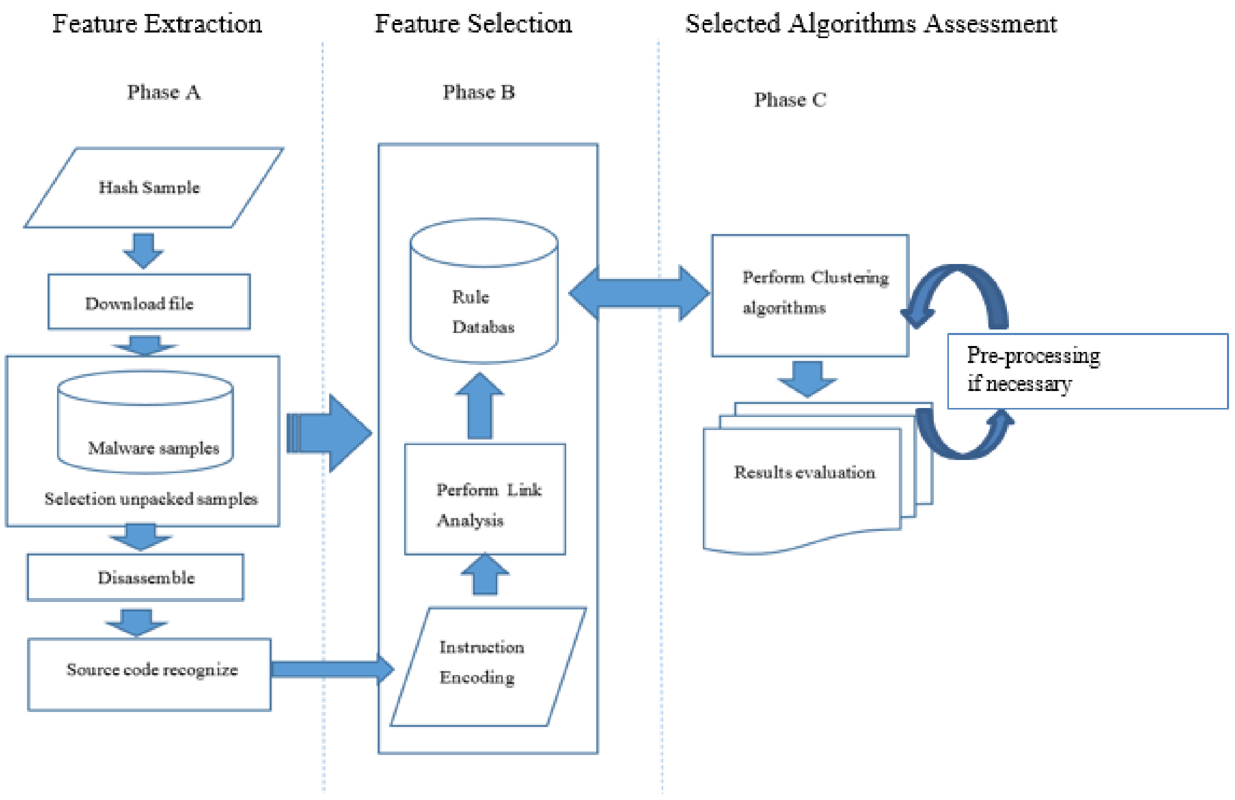


Figure 3-1Architecture of the malicious code detector

For this purpose, R-3.4.3 language used since a variety of clustering algorithms and functions are available both in statistical R packages and libraries. A collection of R Statistical packages/ libraries pack has been used which is listed in Table 3.2. As stated, the method is assessed with a standard package in which established leading clustering algorithms from statistics and ME implemented.

Name of the package	Key Usage
Mclust	For model-based clustering, classification, and density estimation based on finite normal mixture modelling. To provide functions for parameter estimation via the EM algorithm for normal mixture models with a

	variety of covariance structures, and functions for simulation from these models. It also included are functions that combine model-based hierarchical clustering, EM for mixture estimation and the Bayesian Information Criterion (BIC) in comprehensive strategies for clustering, density estimation, and discriminant analysis. Additional functionalities are available for displaying and visualizing fitted models along with clustering, classification, and density estimation results.

Table 3-1 R-packages

The classification of observations into groups requires some methods to calculate the distance or the (dis)similarity between each pair of observations. The result of this computation is known as a dissimilarity or distance matrix**.** Generally, it is recommended to standardize the variables before distance matrix computation. Standardization makes variable comparable, in the situation where they are measured in different scales. The results of link analysis in current research provide standard variables. In this research, common distance measures that best differentiate each cluster are ranked by segmentation analysis and evaluate by GMM algorithm.

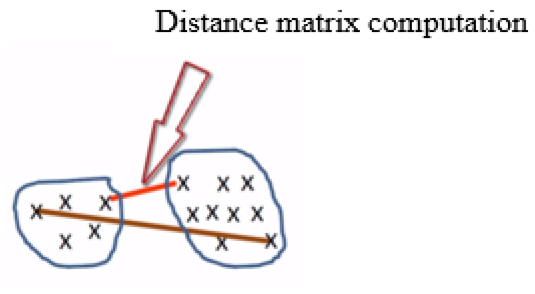


Figure 3-2 Clustering Distance

There are many methods to calculate this distance of observations. Common distance measures in unsupervised algorithms iteratively collects points/groups together until the desired number of clusters enlisted. The distribution of observations after conducting link analysis is visible in Figure 3-3.

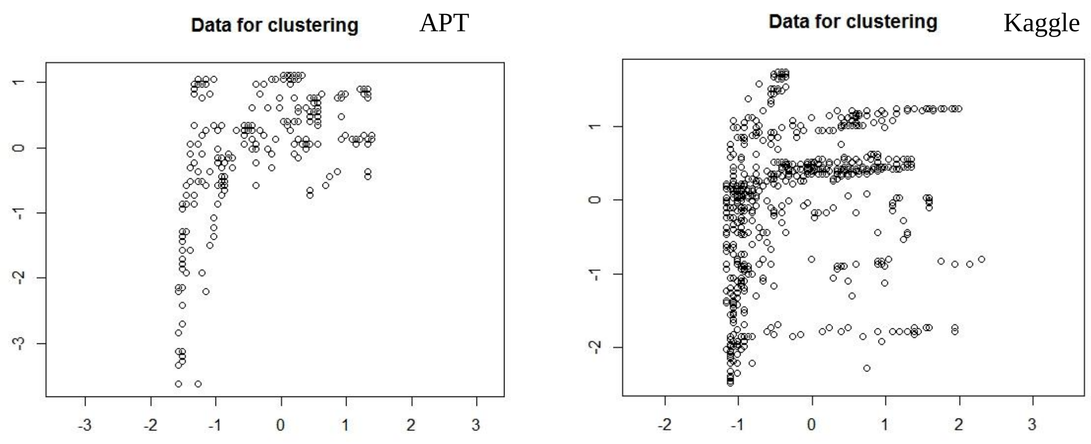


Figure 3-3Malware samples for clustering

According to the research objective we want our clusters to be as compact and separated as possible. In the following section, the justification why the Expectation Maximization clustering using GMMs implemented in Phase C provided. The most leading clustering algorithms-namely *Centroid-based clustering* (k-means, k-medoid), *Connectivity-based clustering* (hierarchical clustering), *and Distribution-based clustering* (Expectation Maximization /Gaussian mixture models) briefly explained as follow.

3.2 Expectation Maximization (EM) Clustering

The Expectation Maximization (EM) algorithm is a soft clustering technique. It used for the iterative computation of maximum likelihood (ML) estimates. The EM algorithm is commonly used for estimating the mixture model density. Moreover, it is useful in a variety of problems where the data may be viewed as incomplete data. Wherefore, EM can be viewed as clustering based on "hidden" probability distribution. Gaussian probability distribution employed to assess the strength of association between clusters and instances.

3.3 K-means and K-medoids Clustering

Key points of K-Means is summaries in Table 3-2. As discussed in chapter two, hierarchical clustering is a good tool for exploratory analysis in multivariate statistics whereas K-means clustering is a good tool for modelling. K-means clustering allows the observations to be moved from one group to another throughout the algorithm, a process that does not occur in hierarchical methods. The results of the hierarchical cluster analysis are used to set k in the K-means cluster analysis.

Table 3-2 K-means algorithm key points

K-Means Algorithm Key Notes
Takes the number of component of the population equal to the final required number of clusters
Examines each component in the population
Assigns it to one of the clusters depending on the minimum distance
Centroid’s position it recalculated every time a component is added
Unable to handle noisy and outliners
Need to specify k, the number of clusters, in advance

It is hard to assign number of clusters when the overlapping occurs in soft clustering. K-means uses Euclidean distance to the centre. Therefore, if the clusters define by non-circular shape K-means cannot discover with this type of assignment of features. K-medoid or pam is based on centroids or medoid. Both k-means and k-medoids algorithms are breaking the dataset up into k groups. That is, we determine the parameters of K probability distributions that specify the clusters. Moreover, they are both trying to minimize the distance between points of the same cluster and a particular point which is the centre of that cluster. In contrast to the k-means algorithm, the k-medoids algorithm chooses points as centres that belong to the dataset. The most common implementation of the k-medoids clustering algorithm is Partitioning Around Medoids (PAM) algorithm. PAM algorithm uses a greedy search which may not find the global optimum solution. While medoids are more robust to outliers than centroids. However, they need more computation for high dimensional data.

3.4 Hierarchical Clustering

Hierarchal is an approach for clustering one data point in more than one clusters. In hierarchical cluster analysis, it is important to consider a few different linkage methods to identify which method appropriately clusters the data type such as single, complete, average linkage or Ward. The comparison between the above algorithms is summarized in Table 3-3.

Table 3-3 Clustering comparison

Clustering name	**K-means**	**Heretical clustering**
Advantages	Assign each data point to exactly one cluster Simple Fast(usually scales linearly)	Simple Can produce a tree for visualisation
Disadvantages	Used Euclidean distance Don't play well with non-liner data Need to specify a value for the number of clusters in advance	Sensitive to noise and outlines Slow

3.5 Why EM?

When the class for each observation is unknown and estimation of them is target of clustering we can use the GMM model. GMM fitted via Expectation-Maximization (EM) algorithm has been used for model-based clustering, classification, and density estimation, dimension reduction for visualisation-means and EM in the sense that each data point is assigned to a cluster, they perform the same step. In EM, significance of statistical distribution of variables in the dataset is the measure. Therefore, every point has a probability of being associated with every cluster. For this reason, it can be viewed as "fuzzy" approach to clustering. Moreover, both K-means and hierarchical clustering work on heuristic approach to build clusters, and do not rely on a formal model. Model-based clustering assumes a data model and applies an EM algorithm to find the most likely model components and the number of clusters.

3.6 Obtaining and Dealing with Data

Before embarking upon a statistical cluster analysis, there are certain data cleaning and data preparation steps that should be countered. The process of phase A and B are outlined in this section. The following steps are customized in fulfilment of the research objectives II to VI.

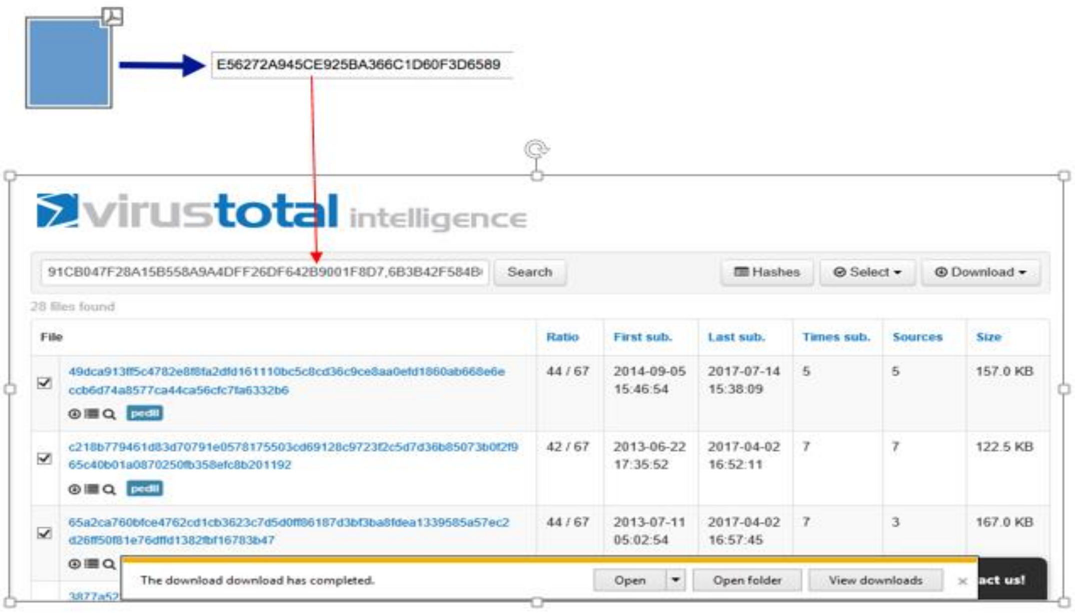
- Data Collection: the first step needs to collect the malware hashes from various resources.
- Data Preparation: the second step is to download the file and recognize unpacked malware samples.
- Feature Extraction: raw source code extracts from each sample.

- Data Cleaning: filtering irrelevant information.
- Feature Construction: the data even after cleaning are not ready for clustering as it needs to be constructed as variables. The basis variables used in the cluster detection algorithm should represent characteristics of malware samples.
- Feature Selection: variables with little relationship to the target were excluded from the analysis.

3.6.1 Data Collection

Ideally, to work on datasets which are meaningful to our research aims, it was necessary to find and download samples consisting of various advanced malicious files. If data collect from standard repositories and comes from multiple families, it could build a robust model. For the sake of learning and practicing the various algorithms, there are a few websites available which provide academic or publicly available datasets. The researcher obtained a total of 500 samples from Kaggle, VirusTotal, VirusShare, Contagio and also FireEye. The samples were also evaluated through the peer review process by Virsutotal Intelligence platform.

In phase A, collected malware hashes uploaded to www.virustotal.com to download their file. Virsutotal is a web interface which is a virus scan program integrated with several different antivirus engines into a single web interface. A large corpus of APTs downloaded in this way, see Figure 3-4.


E56272A945CE925BA366C1D60F3D6589

virustotal intelligence

91CB047F28A15B558A9A4DFF26DF642B9001F8D7,6B3B42F584B

28 files found

	File	Ratio	First sub.	Last sub.	Times sub.	Sources	Size
☑	49dca913ff5c4782e8f8fa2dfd161110bc5c8cd36c9ce8aa0efd1860ab668e6e ccb6d74a8577ca44ca56cfc7fa6332b6 pedll	44 / 67	2014-09-05 15:46:54	2017-07-14 15:38:09	5	5	157.0 KB
☑	c218b779461d83d70791e0578175503cd69128c9723f2c5d7d36b85073b0f2f9 65c40b01a0870250fb358efc8b201192 pedll	42 / 67	2013-06-22 17:35:52	2017-04-02 16:52:11	7	7	122.5 KB
☑	65a2ca760bfce4762cd1cb3623c7d5d0ff86187d3bf3ba8fdea1339585a57ec2 d26ff50f81e76dffd1382fbf16783b47	44 / 67	2013-07-11 05:02:54	2017-04-02 16:57:45	7	3	167.0 KB

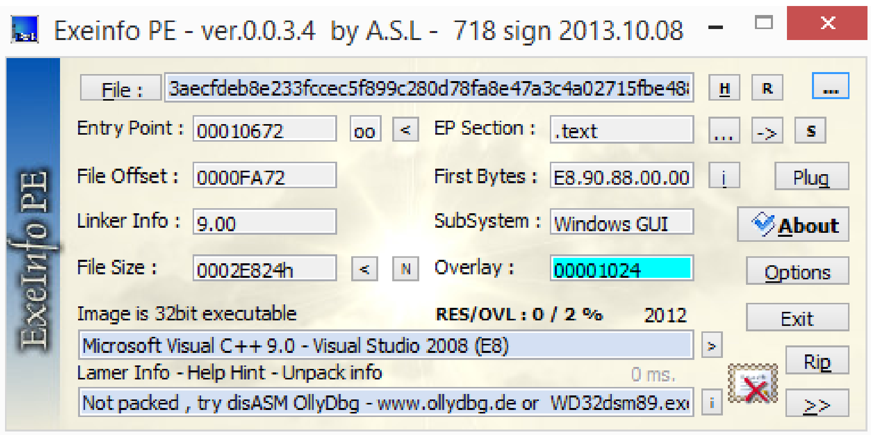
The download download has completed.

Figure 3-4Download process by Virsutotal Intelligence

An experiment has done in VMware. All malware were verified from Virsutotal and FireEye websites.

3.6.2 Data Preparation

Once datasets are collected. The next step is to find out unpacked files to have consistency in each sample datasets. This evaluation is done by the usage of Exeinfo PE, see Figure 3-5.The next challenge was static analysis.


Exeinfo PE - ver.0.0.3.4 by A.S.L - 718 sign 2013.10.08

File : 3aecfdeb8e233fccec5f899c280d78fa8e47a3c4a02715fbe48

Entry Point : 00010672 EP Section : .text

File Offset : 0000FA72 First Bytes : E8.90.88.00.00

Linker Info : 9.00 SubSystem : Windows GUI

File Size : 0002E824h Overlay : 00001024

Image is 32bit executable RES/OVL : 0 / 2 % 2012

Microsoft Visual C++ 9.0 - Visual Studio 2008 (E8)

Lamer Info - Help Hint - Unpack info 0 ms.

Not packed , try disASM OllyDbg - www.ollydbg.de or WD32dsm89.exe

Figure 3-5 Examine malware file

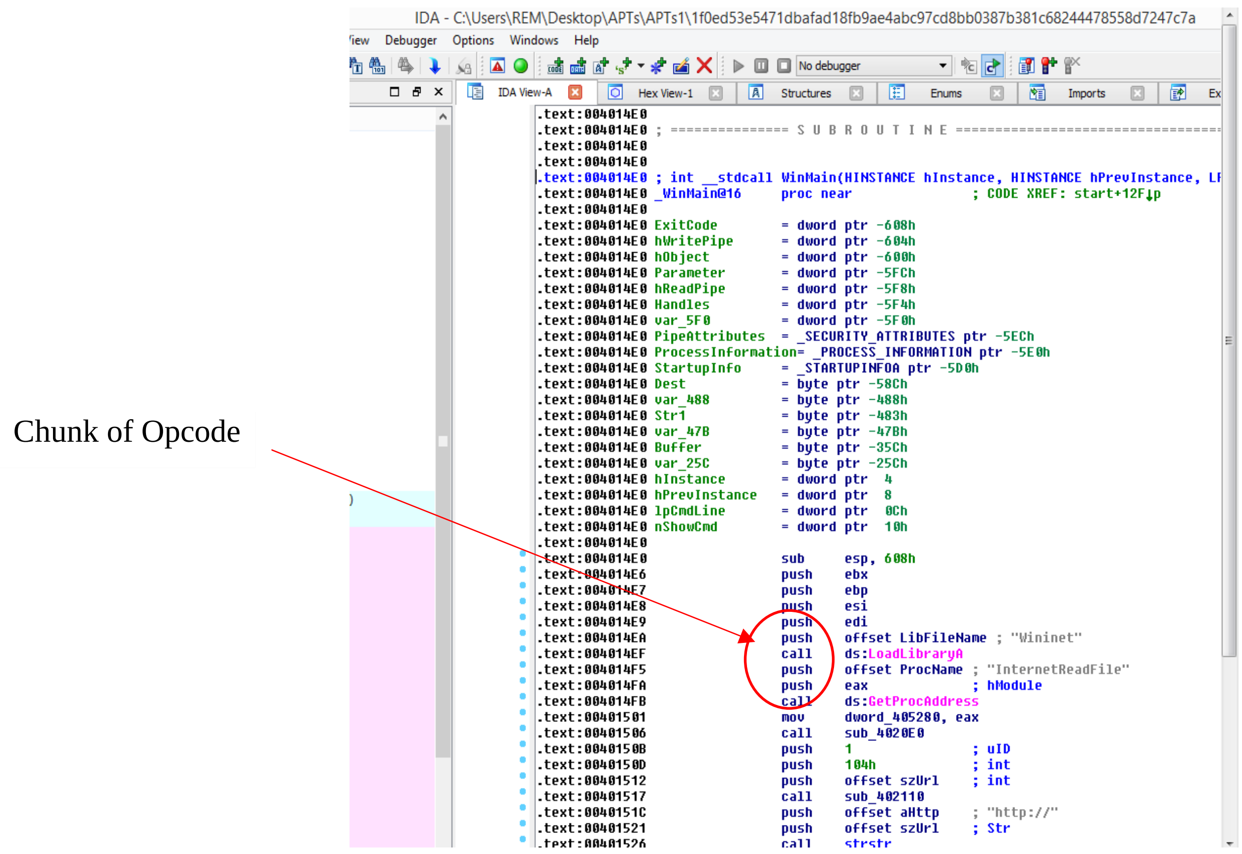
3.6.3 Feature Extraction

To capture the necessary opcode information is done by some static analysis that has been developed and ready to use, like IDA Pro 32/64. IDA Pro is included in best lists as claimed by statistical analysis launched by SAN institute. It used to disassemble a malware program into a sequence of machine instructions that are then used for feature extraction, see Figure 3-6. A portion of opcode generated by IDA Pro pointed in Figure 3-7. The gain information from this part is saved in an excel file for each individual malware sample.

`

```
.text:00401017    call sub_401098
.text:0040101C    push 8
.text:0040101E    lea ecx, [esp+24h+var_14]
.text:00401022    push offset xyz
.text:00401027    push ecx
.text:00401028    call sub_401060
.text:0040102D    add esp, 18h
.text:00401030    test eax, eax
.text:00401032    jz short loc_401045
```

Figure 3-6 Example Disassembly opcode


Chunk of Opcode
IDA - C:\Users\REM\Desktop\APTs\APTs1\1f0ed53e5471dbafad18fb9ae4abc97cd8bb0387b381c68244478558d7247c7a
View Debugger Options Windows Help
No debugger
IDA View-A
Hex View-1
Structures
Enums
Imports
Ex

```
.text:004014E0
.text:004014E0 ; =============== S U B R O U T I N E =================================
.text:004014E0
.text:004014E0
.text:004014E0 ; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LP
.text:004014E0 _WinMain@16     proc near               ; CODE XREF: start+12F↓p
.text:004014E0
.text:004014E0 ExitCode        = dword ptr -608h
.text:004014E0 hWritePipe      = dword ptr -604h
.text:004014E0 hObject         = dword ptr -600h
.text:004014E0 Parameter       = dword ptr -5FCh
.text:004014E0 hReadPipe       = dword ptr -5F8h
.text:004014E0 Handles         = dword ptr -5F4h
.text:004014E0 var_5F0         = dword ptr -5F0h
.text:004014E0 PipeAttributes  = _SECURITY_ATTRIBUTES ptr -5ECh
.text:004014E0 ProcessInformation= _PROCESS_INFORMATION ptr -5E0h
.text:004014E0 StartupInfo     = _STARTUPINFOA ptr -5D0h
.text:004014E0 Dest            = byte ptr -58Ch
.text:004014E0 var_488         = byte ptr -488h
.text:004014E0 Str1            = byte ptr -483h
.text:004014E0 var_47B         = byte ptr -47Bh
.text:004014E0 Buffer          = byte ptr -35Ch
.text:004014E0 var_25C         = byte ptr -25Ch
.text:004014E0 hInstance       = dword ptr  4
.text:004014E0 hPrevInstance   = dword ptr  8
.text:004014E0 lpCmdLine       = dword ptr  0Ch
.text:004014E0 nShowCmd        = dword ptr  10h
.text:004014E0
.text:004014E0                 sub     esp, 608h
.text:004014E6                 push    ebx
.text:004014E7                 push    ebp
.text:004014E8                 push    esi
.text:004014E9                 push    edi
.text:004014EA                 push    offset LibFileName ; "Wininet"
.text:004014EF                 call    ds:LoadLibraryA
.text:004014F5                 push    offset ProcName ; "InternetReadFile"
.text:004014FA                 push    eax             ; hModule
.text:004014FB                 call    ds:GetProcAddress
.text:00401501                 mov     dword_405280, eax
.text:00401506                 call    sub_4020E0
.text:0040150B                 push    1               ; uID
.text:0040150D                 push    104h            ; int
.text:00401512                 push    offset szUrl    ; int
.text:00401517                 call    sub_402110
.text:0040151C                 push    offset aHttp    ; "http://"
.text:00401521                 push    offset szUrl    ; Str
.text:00401526                 call    strstr
```

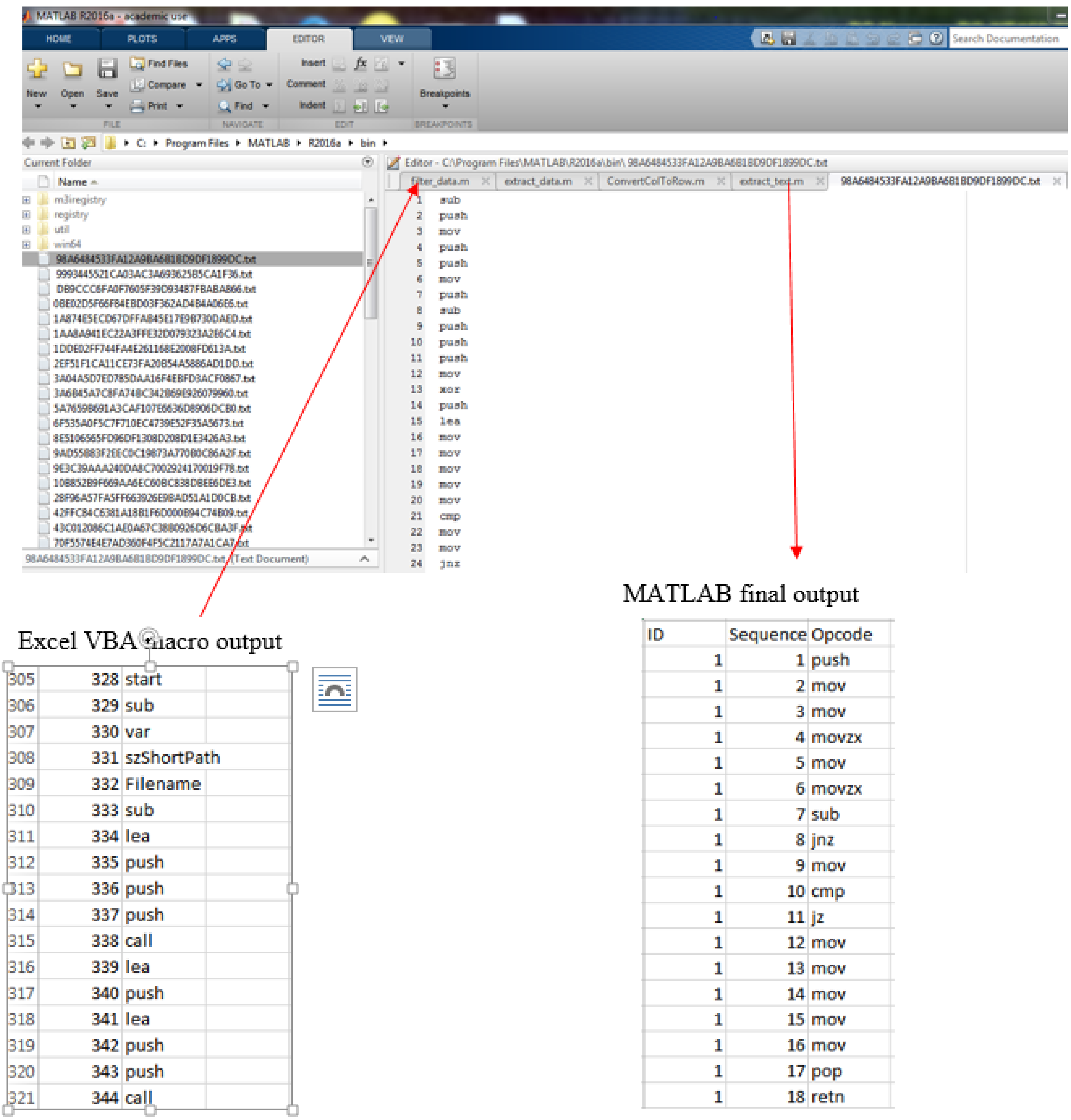
Figure 3-7Static Analysis-IDA Pro

All information collected during static analysis was compiled into a single database for each malware family.

3.6.4 Data Cleaning

The gain information from the previous stage contains opcode, instruction reference and also redundant data. Thus, all operands, labels, directives, etc. are discarded only retain the mnemonic opcode.

The filtering process has been done by MATLAB script and a Macro in Excel file to generate static opcode sequence, visible in Figure 3-8


MATLAB R2016a - academic use
HOME
PLOTS
APPS
EDITOR
VIEW
Search Documentation
New
Open
Save
Find Files
Compare
Print
Go To
Find
Insert
Comment
Indent
Breakpoints
FILE
NAVIGATE
EDIT
BREAKPOINTS
C: ▸ Program Files ▸ MATLAB ▸ R2016a ▸ bin ▸
Current Folder
Name
Editor - C:\Program Files\MATLAB\R2016a\bin\ 98A6484533FA12A9BA6B1BD9DF1899DC.txt
filter_data.m
extract_data.m
ConvertColToRow.m
extract_text.m
98A6484533FA12A9BA6B1BD9DF1899DC.txt
98A6484533FA12A9BA6B1BD9DF1899DC.txt (Text Document)

```
m3iregistry
registry
util
win64
98A6484533FA12A9BA6B1BD9DF1899DC.txt
9993445521CA03AC3A693625B5CA1F36.txt
DB9CCC6FA0F7605F39D93487FBABA866.txt
0BE02D5F66F84EBD03F362AD4B4A06E6.txt
1A874E5ECD67DFFA845E17E9B730DAED.txt
1AA8A941EC22A3FFE32D079323A2E6C4.txt
1DDE02FF744FA4E261168E2008FD613A.txt
2EF51F1CA11CE73FA20B54A5886AD1DD.txt
3A04A5D7ED785DAA16F4EBFD3ACF0867.txt
3A6B45A7C8FA748C342869E926079960.txt
5A7659B691A3CAF107E6636D8906DCB0.txt
6F535A0F5C7F710EC4739E52F35A5673.txt
8E5106565FD96DF1308D208D1E3426A3.txt
9AD55B83F2EEC0C19873A770B0C86A2F.txt
9E3C39AAA240DA8C70029241700 19F78.txt
10B852B9F669AA6EC60BC838DBEE6DE3.txt
28F96A57FA5FF663926E9BAD51A1D0CB.txt
42FFC84C6381A18B1F6D000B94C74B09.txt
43C012086C1AE0A67C3880926D6CBA3F.txt
70F5574E4E7AD360F4F5C2117A7A1CA7.txt
```

```
sub
push
mov
push
push
mov
push
sub
push
push
push
mov
xor
push
lea
mov
mov
mov
mov
mov
cmp
mov
mov
jnz
```

Excel VBA macro output

305	328	start
306	329	sub
307	330	var
308	331	szShortPath
309	332	Filename
310	333	sub
311	334	lea
312	335	push
313	336	push
314	337	push
315	338	call
316	339	lea
317	340	push
318	341	lea
319	342	push
320	343	push
321	344	call

MATLAB final output

ID	Sequence	Opcode
1	1	push
1	2	mov
1	3	mov
1	4	movzx
1	5	mov
1	6	movzx
1	7	sub
1	8	jnz
1	9	mov
1	10	cmp
1	11	jz
1	12	mov
1	13	mov
1	14	mov
1	15	mov
1	16	mov
1	17	pop
1	18	retn

Figure 3-8 Procedure of Data Cleaning

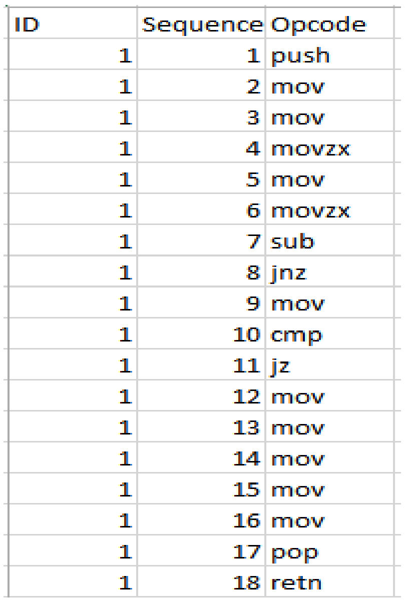
`

The obtained data from this step, it serves as a means to convert the capture opcode sequence into a raw dataset.

3.6.5 Feature Construction

Usually, there are thousands of rows (opcodes) representing individual malware type. As per research objective, it is necessary to understand and check source code relationships and links prior to running a data clustering technique. As it mentioned in section 2.5, Link Analysis tailors code instructions to opcode features. For this purpose, opcode sequence of each sample was captured and added to an individual repository for each family. Table 3-4 partically illustrates such a repository for class number one.

Table 3-4 Opcode Sequence -Gatak Family



ID	Sequence	Opcode
1	1	push
1	2	mov
1	3	mov
1	4	movzx
1	5	mov
1	6	movzx
1	7	sub
1	8	jnz
1	9	mov
1	10	cmp
1	11	jz
1	12	mov
1	13	mov
1	14	mov
1	15	mov
1	16	mov
1	17	pop
1	18	retn

To gain a better understanding of the relationship betweeen opcodes in the network, link analysis performed separately for each class, visible in Figure 3-9.

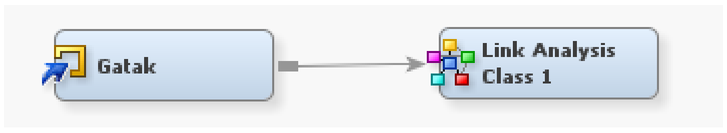


Figure 3-9Shame of Link Analysis process diagram

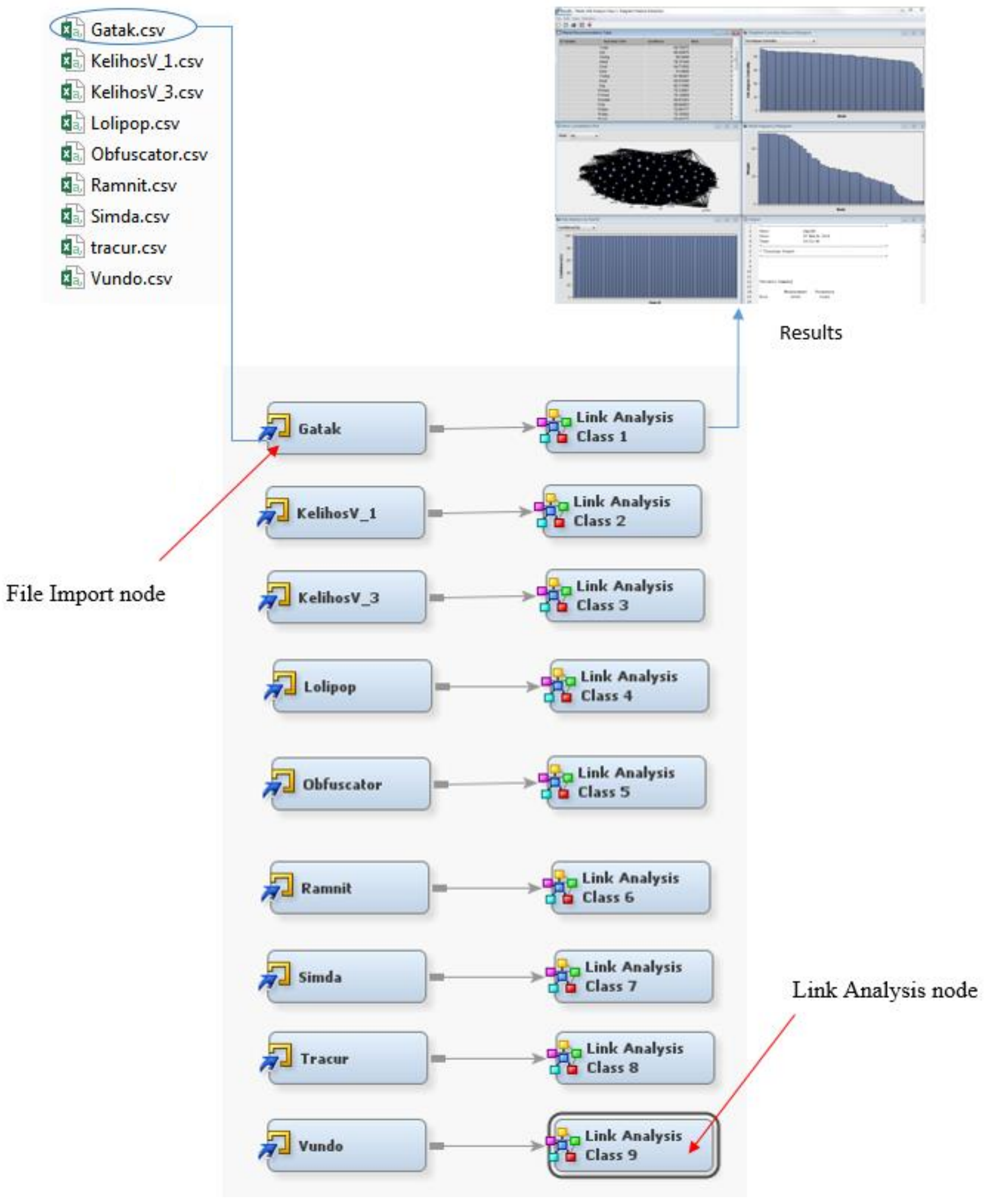
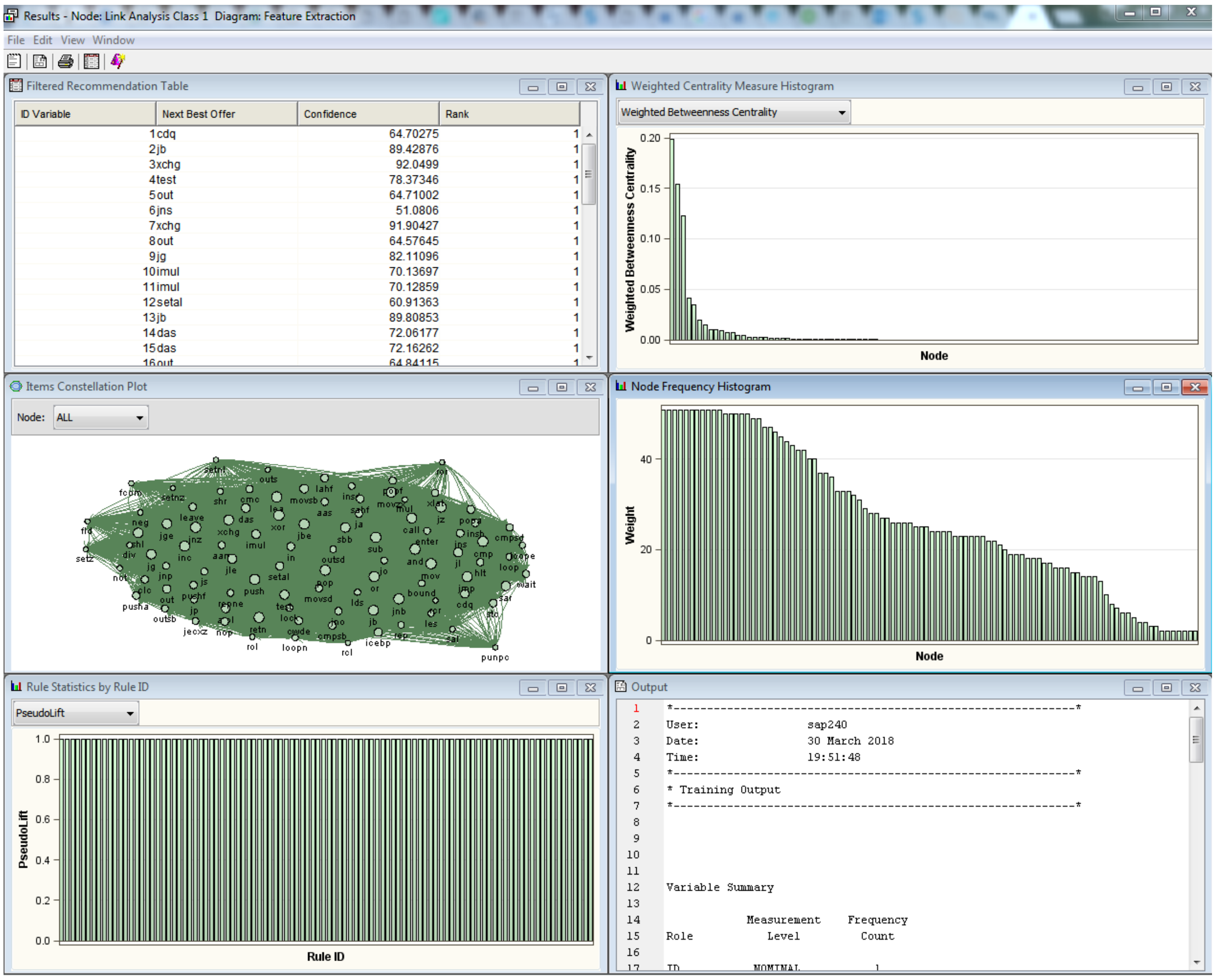


Figure 3-10 Link Analysis process diagram

After the run completes successfully, the results and generate valuable insights are available. The results view generates several outputs to allow us to explore the data, discover the associations and view them graphically in a network graph, view multiple centrality measures as well as show scoring results, see Figure 3-11.Besides, Link Analysis node builds a full rule set (a network of opcode) and then prune to the most important rules. The visualizing the relationship between all opcode as items is reported in the separated window.


Results - Node: Link Analysis Class 1 Diagram: Feature Extraction
File Edit View Window
Filtered Recommendation Table
Weighted Centrality Measure Histogram
Weighted Betweenness Centrality
Weighted Betweenness Centrality
Node
Items Constellation Plot
Node: ALL
Node Frequency Histogram
Weight
Node
Rule Statistics by Rule ID
PseudoLift
PseudoLift
Rule ID
Output

ID Variable	Next Best Offer	Confidence	Rank
1	cdq	64.70275	1
2	jb	89.42876	1
3	xchg	92.0499	1
4	test	78.37346	1
5	out	64.71002	1
6	jns	51.0806	1
7	xchg	91.90427	1
8	out	64.57645	1
9	jg	82.11096	1
10	imul	70.13697	1
11	imul	70.12859	1
12	setal	60.91363	1
13	jb	89.80853	1
14	das	72.06177	1
15	das	72.16262	1
16	out	64.84115	1

```
*------------------------------------------------------------*
User:                   sap240
Date:                   30 March 2018
Time:                   19:51:48
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*




Variable Summary

             Measurement    Frequency
Role            Level         Count

ID             NOMINAL          1
```

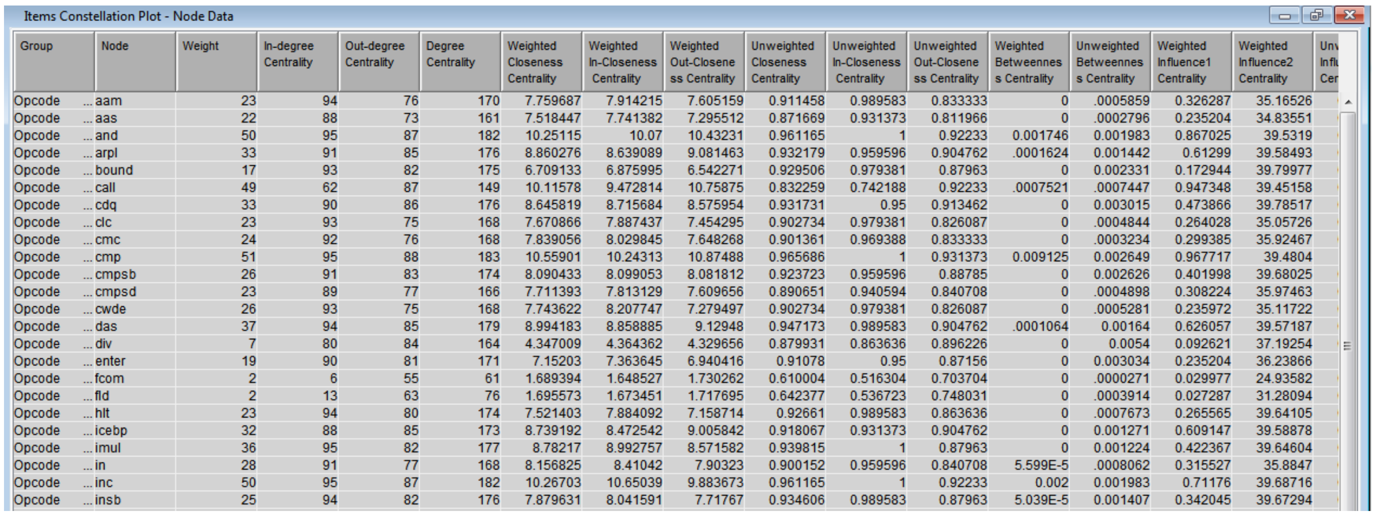
Figure 3-11Sample Link Analysis output

The reports revealed the associations, relationships, structures, and patterns among opcode sequence. Based on the association rules and centrality measure, a bunch of continuous variables produced. These variables consider as opcode features which mentioned below:

opcode, weight, centr_degree_in, centr_degree_out, centr_degree, centr_close_wt, centr_close_in_wt, centr_close_out_wt, centr_close_unwt, centr_close_in_unwt, centr_close_out_unwt, centr_between_wt,

`

centr_between_unwt,centr_influence1_wt,centr_influence2_wt,centr_influence1_unwt,centr_influence2_unwt, centr_cluster



Items Constellation Plot - Node Data

Group	Node	Weight	In-degree Centrality	Out-degree Centrality	Degree Centrality	Weighted Closeness Centrality	Weighted In-Closeness Centrality	Weighted Out-Closeness Centrality	Unweighted Closeness Centrality	Unweighted In-Closeness Centrality	Unweighted Out-Closeness Centrality	Weighted Betweenness Centrality	Unweighted Betweenness Centrality	Weighted Influence1 Centrality	Weighted Influence2 Centrality	Unv Influ Cer
Opcode	... aam	23	94	76	170	7.759687	7.914215	7.605159	0.911458	0.989583	0.833333	0	.0005859	0.326287	35.16526	
Opcode	... aas	22	88	73	161	7.518447	7.741382	7.295512	0.871669	0.931373	0.811966	0	.0002796	0.235204	34.83551	
Opcode	... and	50	95	87	182	10.25115	10.07	10.43231	0.961165	1	0.92233	0.001746	0.001983	0.867025	39.5319	
Opcode	... arpl	33	91	85	176	8.860276	8.639089	9.081463	0.932179	0.959596	0.904762	.0001624	0.001442	0.61299	39.58493	
Opcode	... bound	17	93	82	175	6.709133	6.875995	6.542271	0.929506	0.979381	0.87963	0	0.002331	0.172944	39.79977	
Opcode	... call	49	62	87	149	10.11578	9.472814	10.75875	0.832259	0.742188	0.92233	.0007521	.0007447	0.947348	39.45158	
Opcode	... cdq	33	90	86	176	8.645819	8.715684	8.575954	0.931731	0.95	0.913462	0	0.003015	0.473866	39.78517	
Opcode	... clc	23	93	75	168	7.670866	7.887437	7.454295	0.902734	0.979381	0.826087	0	.0004844	0.264028	35.05726	
Opcode	... cmc	24	92	76	168	7.839056	8.029845	7.648268	0.901361	0.969388	0.833333	0	.0003234	0.299385	35.92467	
Opcode	... cmp	51	95	88	183	10.55901	10.24313	10.87488	0.965686	1	0.931373	0.009125	0.002649	0.967717	39.4804	
Opcode	... cmpsb	26	91	83	174	8.090433	8.099053	8.081812	0.923723	0.959596	0.88785	0	0.002626	0.401998	39.68025	
Opcode	... cmpsd	23	89	77	166	7.711393	7.813129	7.609656	0.890651	0.940594	0.840708	0	.0004898	0.308224	35.97463	
Opcode	... cwde	26	93	75	168	7.743622	8.207747	7.279497	0.902734	0.979381	0.826087	0	.0005281	0.235972	35.11722	
Opcode	... das	37	94	85	179	8.994183	8.858885	9.12948	0.947173	0.989583	0.904762	.0001064	0.00164	0.626057	39.57187	
Opcode	... div	7	80	84	164	4.347009	4.364362	4.329656	0.879931	0.863636	0.896226	0	0.0054	0.092621	37.19254	
Opcode	... enter	19	90	81	171	7.15203	7.363645	6.940416	0.91078	0.95	0.87156	0	0.003034	0.235204	36.23866	
Opcode	... fcom	2	6	55	61	1.689394	1.648527	1.730262	0.610004	0.516304	0.703704	0	.0000271	0.029977	24.93582	
Opcode	... fld	2	13	63	76	1.695573	1.673451	1.717695	0.642377	0.536723	0.748031	0	.0003914	0.027287	31.28094	
Opcode	... hlt	23	94	80	174	7.521403	7.884092	7.158714	0.92661	0.989583	0.863636	0	.0007673	0.265565	39.64105	
Opcode	... icebp	32	88	85	173	8.739192	8.472542	9.005842	0.918067	0.931373	0.904762	0	0.001271	0.609147	39.58878	
Opcode	... imul	36	95	82	177	8.78217	8.992757	8.571582	0.939815	1	0.87963	0	0.001224	0.422367	39.64604	
Opcode	... in	28	91	77	168	8.156825	8.41042	7.90323	0.900152	0.959596	0.840708	5.599E-5	.0008062	0.315527	35.8847	
Opcode	... inc	50	95	87	182	10.26703	10.65039	9.883673	0.961165	1	0.92233	0.002	0.001983	0.71176	39.68716	
Opcode	... insb	25	94	82	176	7.879631	8.041591	7.71767	0.934606	0.989583	0.87963	5.039E-5	0.001407	0.342045	39.67294	

Figure 3-12Items Constellation Plot-Gatak sample Group

In the new dataset, there is an apparent good mix of variables which explain the difference between the groups. Therefore, the primary dataset consists of *N* observations and 17 variables, see Table 3-5

Table 3-5 Featuers names

Variable/Feature name
weight, centr_degree_in
centr_degree_out
centr_degree
centr_close_wt
centr_close_in_wt
centr_close_out_wt
centr_close_unwt
centr_close_in_unwt
centr_close_out_unwt
centr_between_wt, centr_between_unwt
centr_influence1_wt
centr_influence2_wt
centr_influence1_unwt,
centr_influence2_unwt,
centr_cluster

The result measured the redundancy between variables which means that it can be beneficial as it can identify similarities among these observations. These attributes can be meaningful to segment and classify observations which are justified in section 5.1.

`

3.6.6 Feature Selection

Prior to developing predictive models, it is recommended to exclude variables which are unnecessary or irrelevant to the stated dimensional reduction objective. Feature selection methods are normally used to reduce the number of features considered in a classification task by weeding out meaningless features. Thereupon, data segmentation analysis as a scalable feature selection method used for mining meaningful opcode features, refer to section 5.2.

`

Chapter Four

Design and Implementation of Research Case Studies

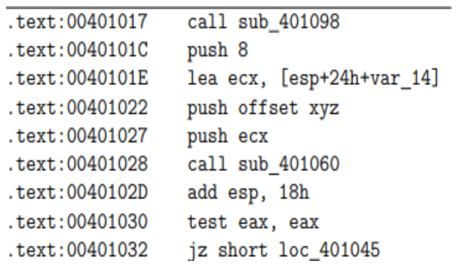
4 Overview

This chapter explains the steps have been taken to complete the Link Analysis process. A sample of 500 malware used to visualize and illustrate which equally divided into two case studies. Two malware datasets which consist of different malware types of known family used. The two case studies are outlined in details in sections 4.2 and 4.3. In both case studies, a similar process employed. A separate experiment is conducted for each of the malware datasets listed below:

- Classifying Kaggle into families (Ramnit, Lollipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator, and Gatak)
- Classifying APTs into groups(APTs1,APTs28,APTs29,APTs30,APTs33)

4.1 Opcode Mining

Opcode structure extracted by utilizing static analysis via IDA Pro 32/64. Figure 4-1 exhibits part of source code. Disassemble samples were compiled into a single database for each malware family. The gain information contains *opcode*, instruction reference and also redundant data. Thus, all operands, labels, directives, etc. are discarded only retain the mnemonic opcodes. The filtering process has been done by MATLAB script and a Macro in Excel file to generate static opcode sequence.



```
.text:00401017    call sub_401098
.text:0040101C    push 8
.text:0040101E    lea ecx, [esp+24h+var_14]
.text:00401022    push offset xyz
.text:00401027    push ecx
.text:00401028    call sub_401060
.text:0040102D    add esp, 18h
.text:00401030    test eax, eax
.text:00401032    jz short loc_401045
```

Figure 4-1Assembly code example generated by IDA-pro

As a case in point, the static opcode sequence corresponding to the example in Figure 4-1 is:

call, push, lea, push, push, call, add, test, jz

The obtained data from this step serves as a means to convert the capture opcode dataset into a raw dataset. Thus, all sequences of opcodes of each family inserted as observation into a CSV file. This way it will be easier to turn it into vectors since data mining algorithms work by performing computation on vectors.

4.2 Kaggle Case Study

A free set provided by Microsoft for a competition is used. Kaggle datasets are widely used for research and studies on malware. This dataset constructed a collection of nine families which contains about 10,000 labelled data instances in total. The size dataset is enormous, therefore a smaller subset randomly is chosen for faster evaluation of the proposed approach

4.2.1 Dataset Characteristics and Pre-processing

This dataset constructed a collection of nine malware families. A brief description of each class is provided below according to the Microsoft threat intelligence archive: https://www.microsoft.com/en-us/wdsi/research/antimalware-security-research-papers.

- **Gatak (class 1)** is a spyware usually hidden in key generators (a piece of software generating product activation key, usually created by hackers) or as an update to the regular application.
- **Kelihos_ver1 (class 2)** is a Trojan and a spam bot. It usually sends messages containing links leading to a virus installer. Some versions of this malware also capture sensitive data about the user, therefore it can be classified as spyware as well.
- **Kelihos_ver3 (class 3)** is just another version of Kelihos_ver1described above. Its functionality is similar, it is just a different implementation.
- **Lollipop (class 4)** is an adware which shows ads in the browsers. In addition, it often monitors the activities of the user via search engines. It is usually distributed as a browser add-in with some third-party software. It affects all main browsers: Fire-fox, Microsoft Internet Explorer and Google Chrome.
- **Obfuscator (class 5)** is, as the name suggests, a tool for hiding other malware from being discovered by antivirus and antispyware software. The hidden malware can be any-thing. This makes this category very hard to detect.

- **Ramnit (class 6)** is a spyware trying to steal financial, banking and social accounts from the PC. Moreover, it disables all Microsoft Windows security features (antivirus, firewall, and User account control).
- **Simda (class 7)** is a Trojan and a spyware. It steals your passwords and sends them to virus creator.
- **Tracur (class 8)** is a Trojan adware redirecting your searches to advertisement pages. It can download other malware as well. Some implementations are even creating back door for a hacker, thereupon they can take over a system.
- **Vundo (class 9)** is an adware and a virus. Additionally, it shows unrequested pop-up ads and downloads other malware and installs it on the system too.

Once opcode excels sheets of each family is generated, it would be ready to feed into SAS.It can be seen from Table 4-1 that the number of rows (opcodes) that belong to each class is different.

Table 4-1Kaggle raw dataset

Family Name	Number of Opcode Sequence
Gatak	136465
KelihosV-1	38797
KelihosV-3	43533
Lolipop	895597
Obfuscator	343722
Ramint	1000000
Smida	67836
Tracur	184399
Vundo	56445

4.2.2 Experimental Set up and Reports

In what follows, the focus will be less on the details of the mathematical algorithms used and more on the visual interpretation of the outcomes. In the figure that appears in the next page the processing completes.

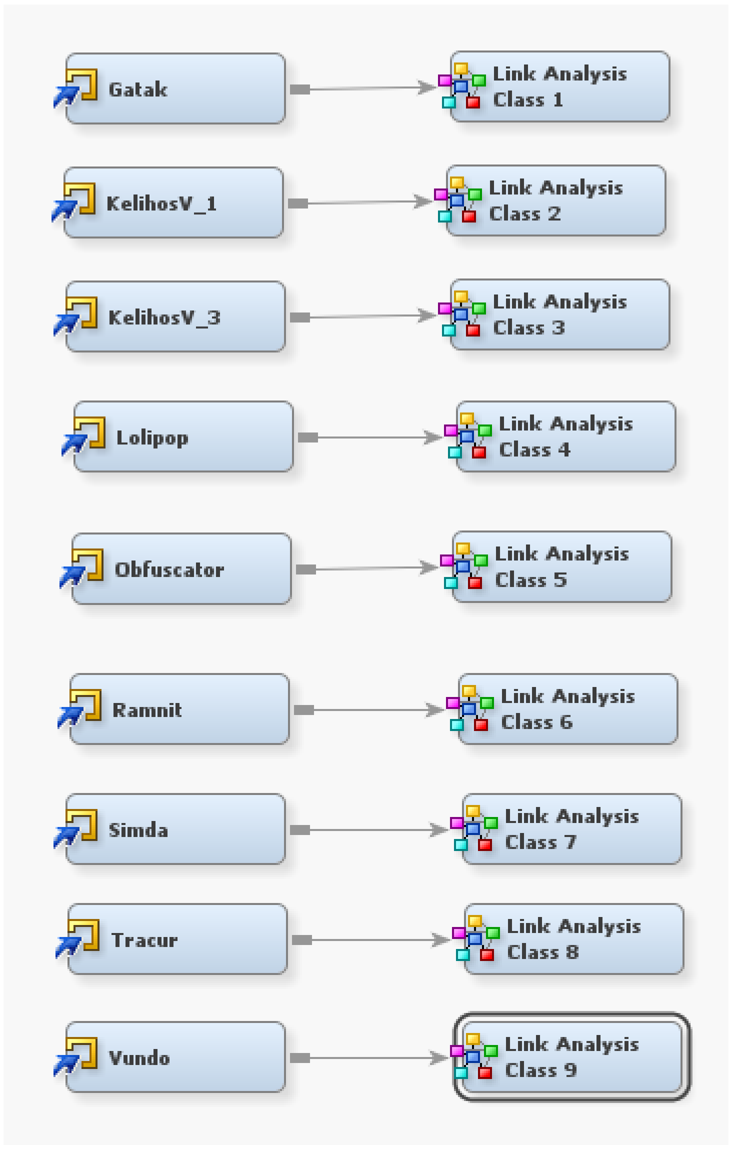


Figure 4-1 Link Analysis process diagram

One of the powerful advantages of link analysis is to convert association and/or sequence rules into network graphs. Each link chart designs according to centrality measures computation. Clustering algorithms could be developed by using these centrality measures-namely *degree*, *betweenness*, *closeness*, and *eighnvector* (Liu *et al.*, 2014). Centrality measure qualifies how important a node is in the association community. It provides information about which variables are meaningful. These variables will be passed as inputs to the variable clustering node in order to select interesting variables associate with each group. These variables formed as raw dataset in MS Excel format and then it uploaded into variable clustering node. Potentially useful features will be pulled out which enhance segmentation of clustering. Part of these variables is shown in Table 4-2.

`

As the Figure 4-2 illustrated link analysis produced interesting differentiation among the groups. Interesting means different forms of link maps which present relationships and connections in opcodes.

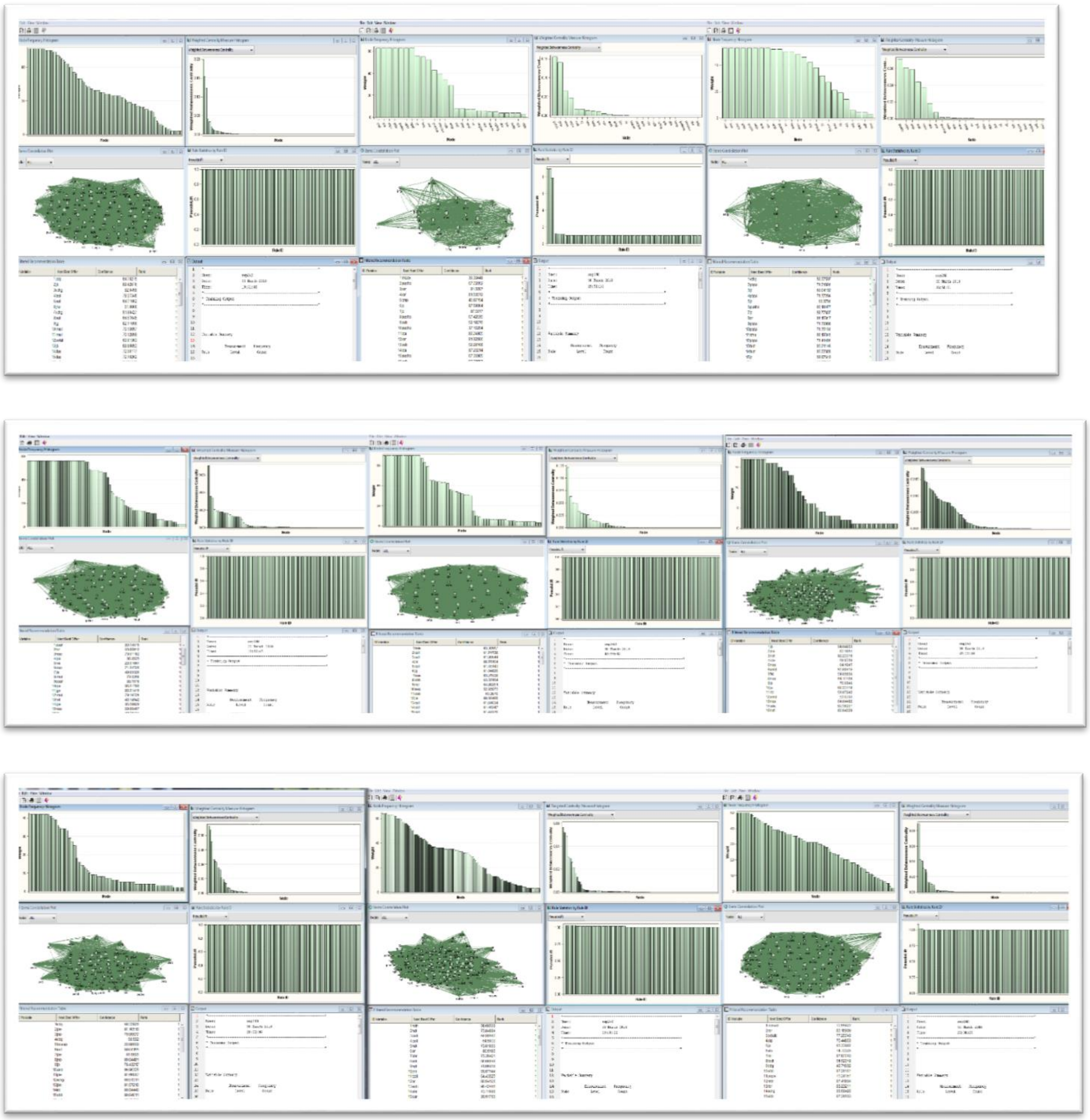
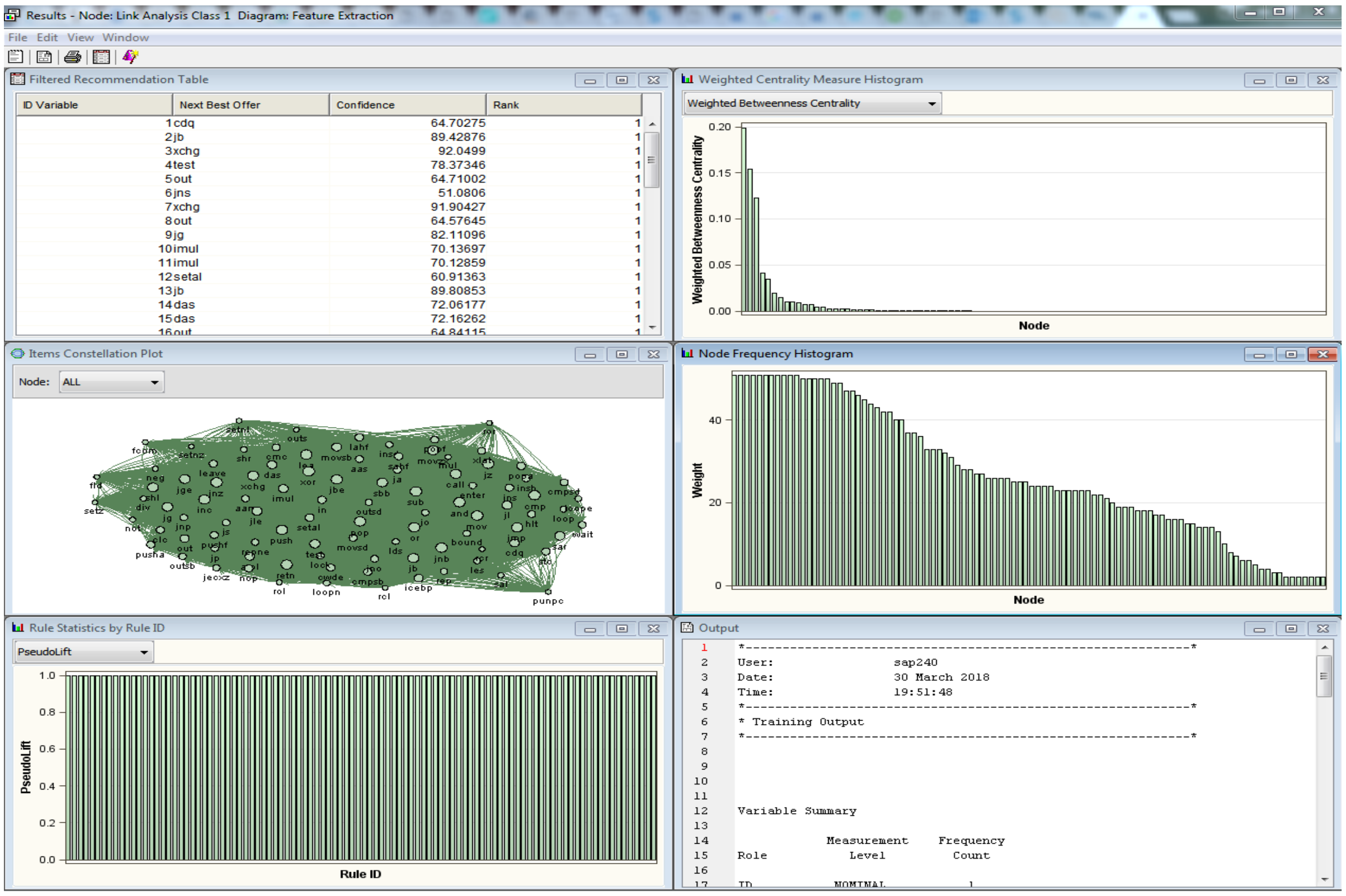


Figure 4-2Link Analysis Windows Output -Kaggle Dataset

Gatak windows output is provided in Figure 4-3.


Results - Node: Link Analysis Class 1 Diagram: Feature Extraction
File Edit View Window
Filtered Recommendation Table
Weighted Centrality Measure Histogram
Weighted Betweenness Centrality
Node
Items Constellation Plot
Node: ALL
Node Frequency Histogram
Weight
Node
Rule Statistics by Rule ID
PseudoLift
Rule ID
Output

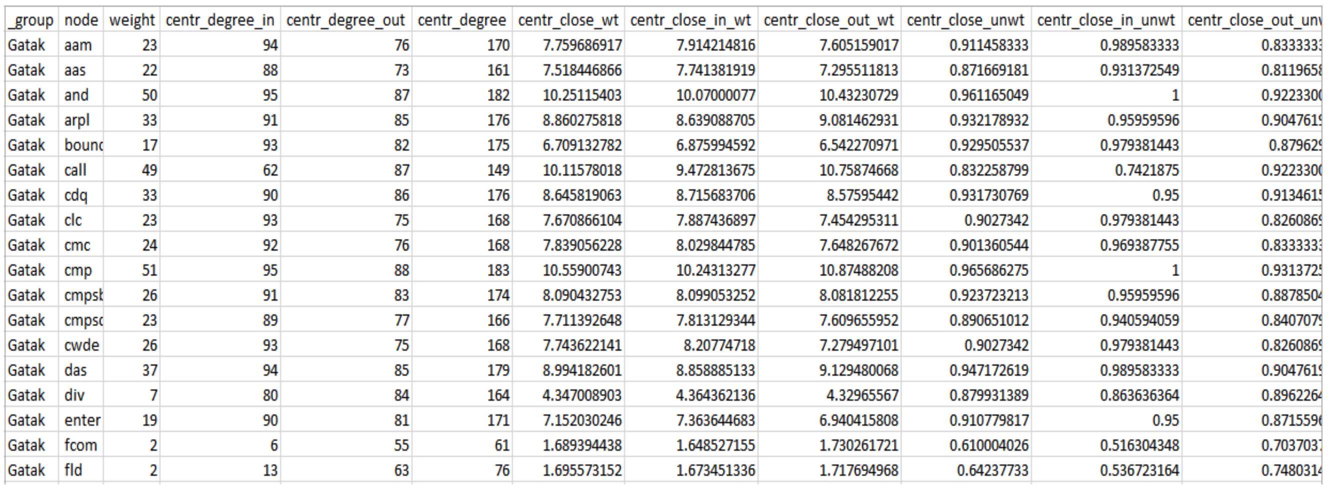
ID Variable	Next Best Offer	Confidence	Rank
1	cdq	64.70275	1
2	jb	89.42876	1
3	xchg	92.0499	1
4	test	78.37346	1
5	out	64.71002	1
6	jns	51.0806	1
7	xchg	91.90427	1
8	out	64.57645	1
9	jg	82.11096	1
10	imul	70.13697	1
11	imul	70.12859	1
12	setal	60.91363	1
13	jb	89.80853	1
14	das	72.06177	1
15	das	72.16262	1
16	out	64.84115	1

```
*------------------------------------------------------------*
User:                    sap240
Date:                    30 March 2018
Time:                    19:51:48
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary

              Measurement    Frequency
Role             Level         Count

ID              NOMINAL          1
```

Figure 4-3Gatak windows output

Extracted calculation of centrality measure can be accessible by reports, see Table 4-2.

Table 4-2 Variables in the Kaggle dataset



_group	node	weight	centr_degree_in	centr_degree_out	centr_degree	centr_close_wt	centr_close_in_wt	centr_close_out_wt	centr_close_unwt	centr_close_in_unwt	centr_close_out_unw
Gatak	aam	23	94	76	170	7.759686917	7.914214816	7.605159017	0.911458333	0.989583333	0.833333
Gatak	aas	22	88	73	161	7.518446866	7.741381919	7.295511813	0.871669181	0.931372549	0.811965
Gatak	and	50	95	87	182	10.25115403	10.07000077	10.43230729	0.961165049	1	0.92233
Gatak	arpl	33	91	85	176	8.860275818	8.639088705	9.081462931	0.932178932	0.95959596	0.904761
Gatak	bounc	17	93	82	175	6.709132782	6.875994592	6.542270971	0.929505537	0.979381443	0.87962
Gatak	call	49	62	87	149	10.11578018	9.472813675	10.75874668	0.832258799	0.7421875	0.92233
Gatak	cdq	33	90	86	176	8.645819063	8.715683706	8.57595442	0.931730769	0.95	0.913461
Gatak	clc	23	93	75	168	7.670866104	7.887436897	7.454295311	0.9027342	0.979381443	0.826086
Gatak	cmc	24	92	76	168	7.839056228	8.029844785	7.648267672	0.901360544	0.969387755	0.833333
Gatak	cmp	51	95	88	183	10.55900743	10.24313277	10.87488208	0.965686275	1	0.931372
Gatak	cmpsl	26	91	83	174	8.090432753	8.099053252	8.081812255	0.923723213	0.95959596	0.887850
Gatak	cmpsc	23	89	77	166	7.711392648	7.813129344	7.609655952	0.890651012	0.940594059	0.840707
Gatak	cwde	26	93	75	168	7.743622141	8.20774718	7.279497101	0.9027342	0.979381443	0.826086
Gatak	das	37	94	85	179	8.994182601	8.858885133	9.129480068	0.947172619	0.989583333	0.904761
Gatak	div	7	80	84	164	4.347008903	4.364362136	4.32965567	0.879931389	0.863636364	0.896226
Gatak	enter	19	90	81	171	7.152030246	7.363644683	6.940415808	0.910779817	0.95	0.871559
Gatak	fcom	2	6	55	61	1.689394438	1.648527155	1.730261721	0.610004026	0.516304348	0.703703
Gatak	fld	2	13	63	76	1.695573152	1.673451336	1.717694968	0.64237733	0.536723164	0.748031

The result of each class analysed carefully and structured meaningful features for next step which is data segmentation analysis.

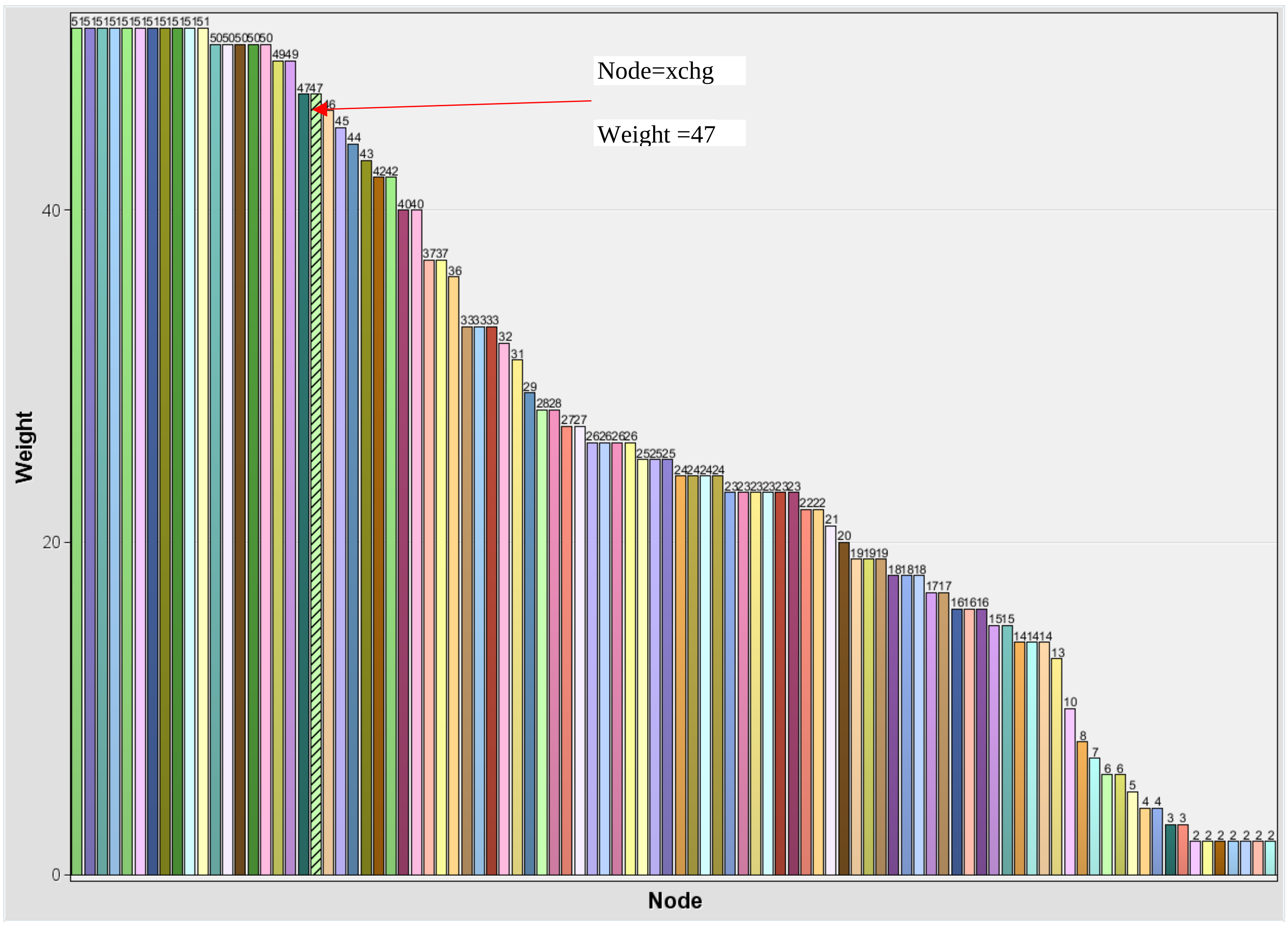


Figure 4-4 Gatak distribution Opcode weight

The node weight of an opcode is the frequency of it. Figure 4-4 and Figure 4-5 simply illustrate which opcode is the most popular in two different families.

To give an example, ,*cmp*,*jnb*,*jnz*,*jz*,*lea*,*mov*,*or*,*pop*,*push*,*sub*,and *xor* are the most popular opcode with weight of 15 in Gatak family. On contrary, *fcom*,*fld*,*punpc*,*ror*,*setnl*,*setnz* are unpopular with lowest weight of 2.

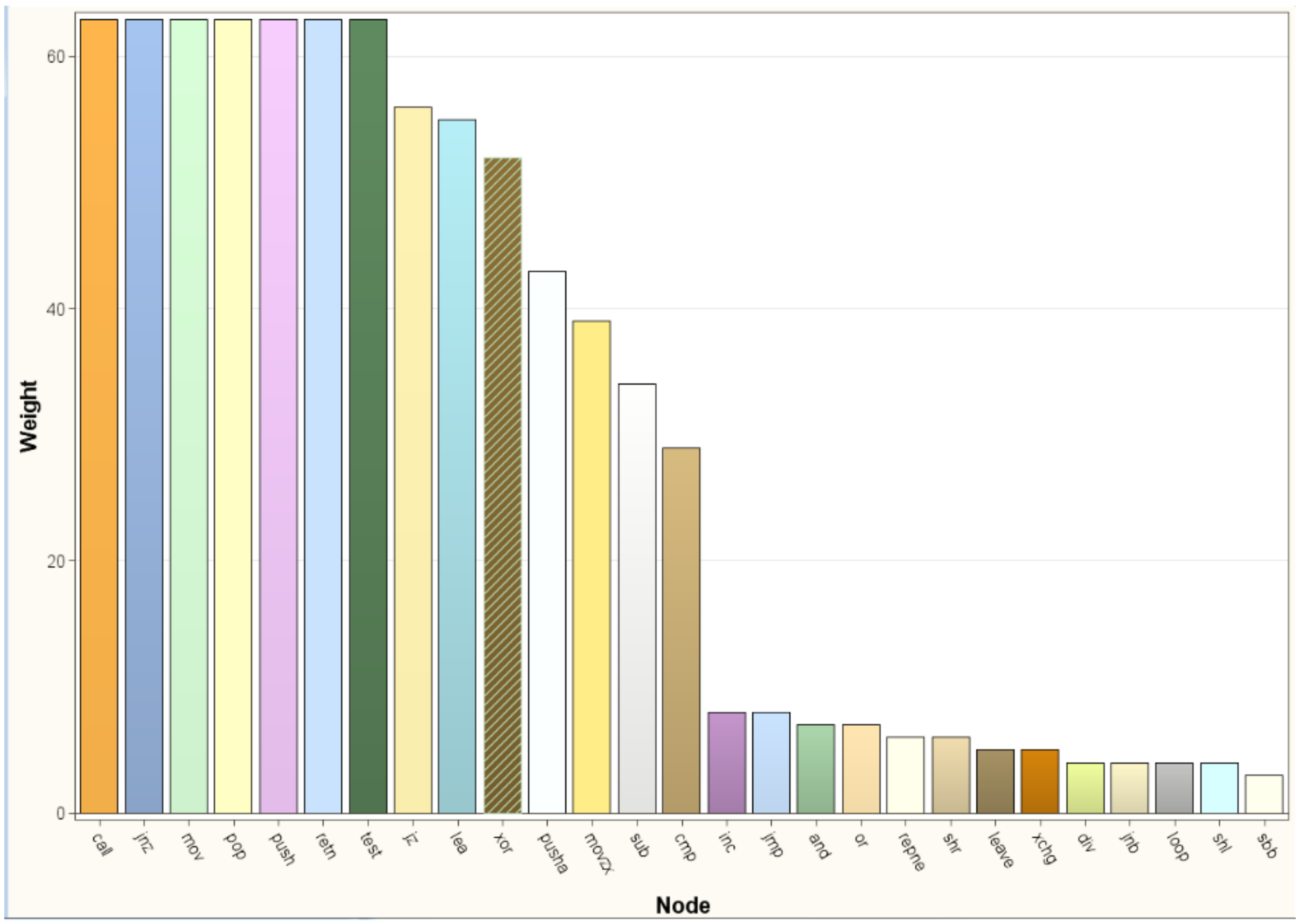


Figure 4-5 KelihosV-1 distribution Opcode weight

From the thickness of the link, the most frequent opcode can be seen. Link Analysis node provides a very powerful visualization of Opcode-malware relations, visible in Figure 4-6 and Figure 4-7.

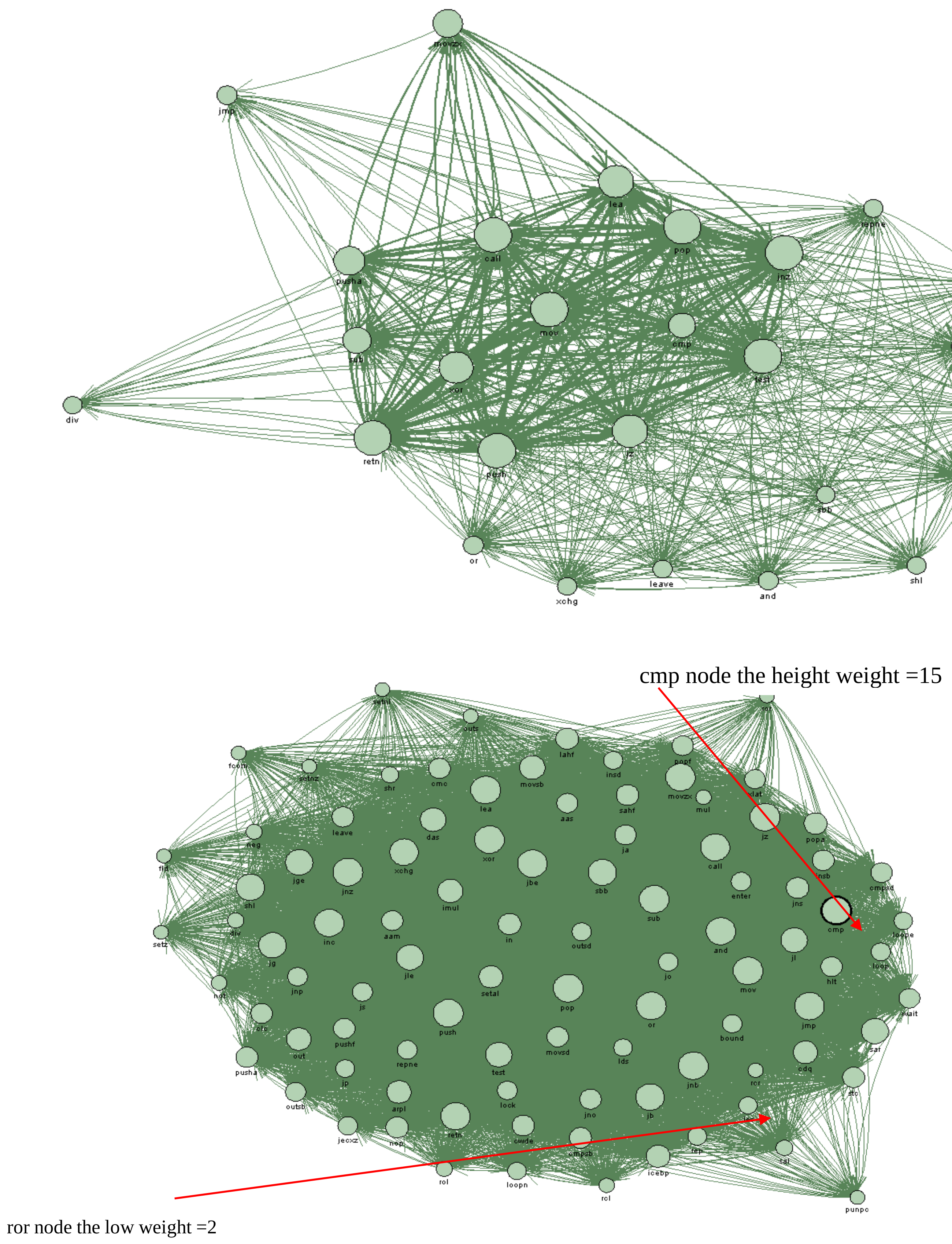


Figure 4-6 Lolipop Opcode Network

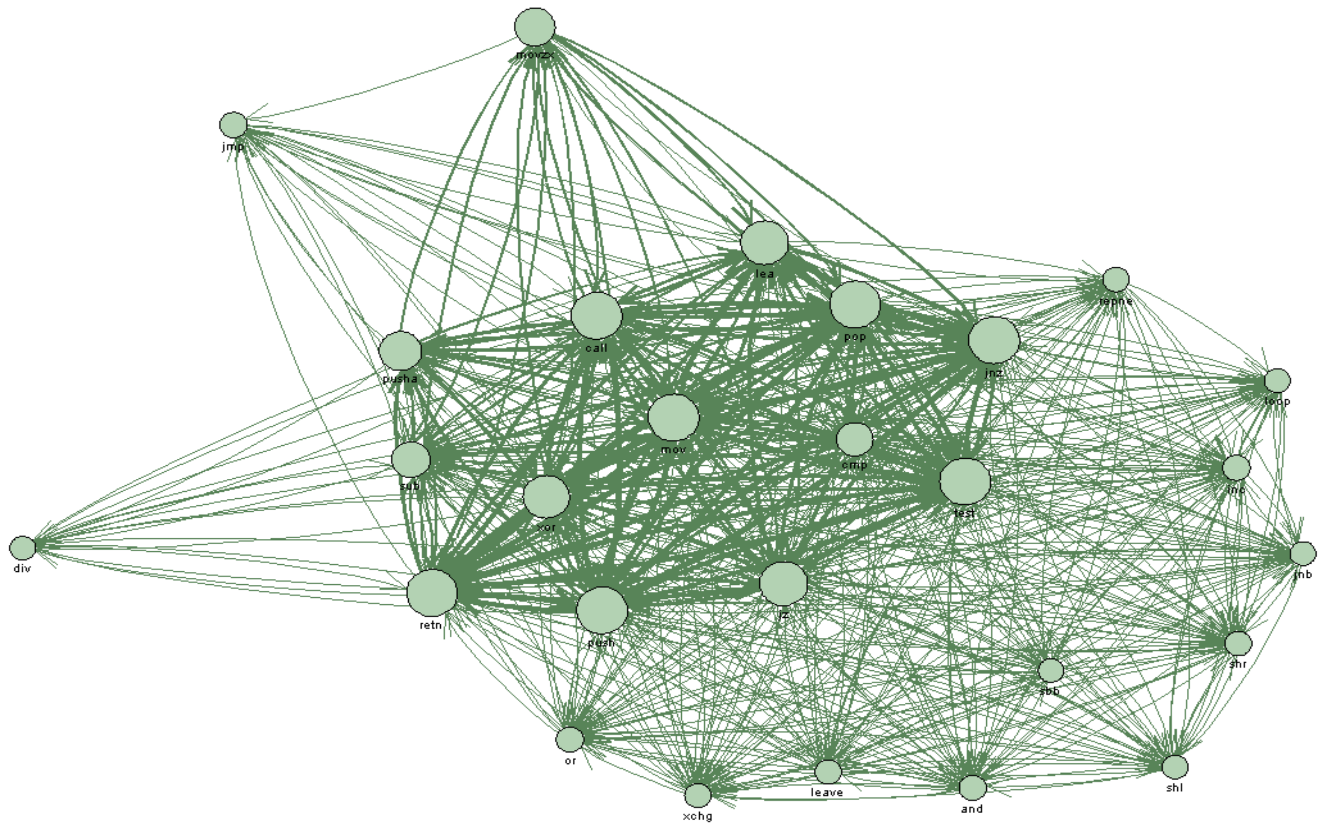


Figure 4-7 KelihosV-1 Opcode Network

The plot gives association graphs among opcode nodes. For more specific data, if a single node or a pattern of node to node linkages is “interesting” the investigator can select one or more nodes for further reporting. Figure visualized a single opcode ‘cmp’ with its neighbours. The size of nodes represent the support of opcode and the thickness of the link represents association strength between opcode.

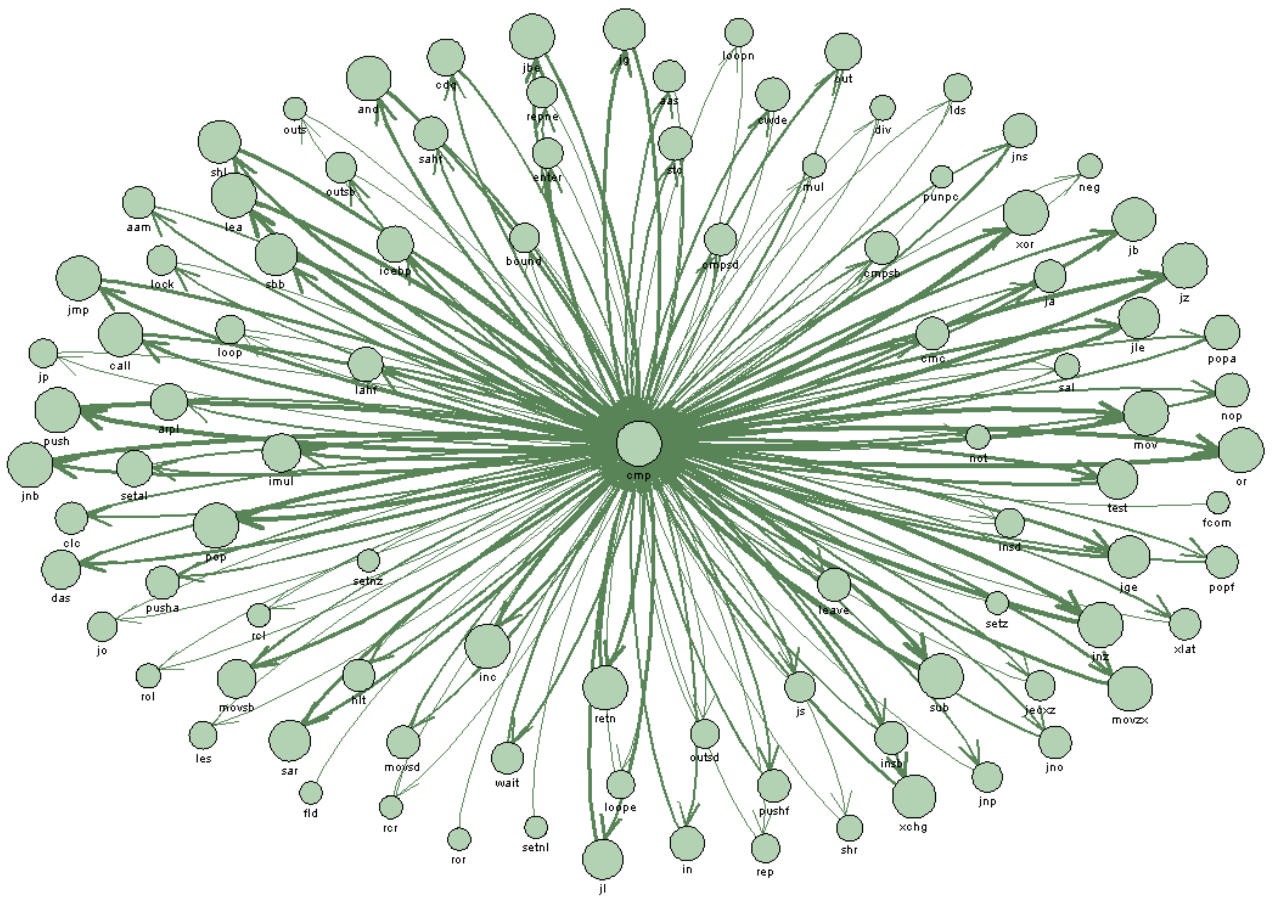
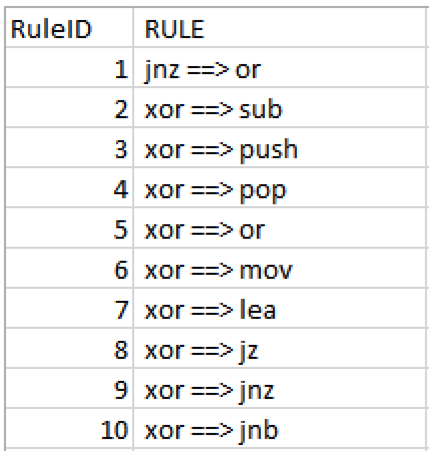


Figure 4-8 Links among selection, Constellation ‘cmp’

Graph based visualization sketches the weight of the nodes and links which represent features vectors as it showed in Table 4-2.Two malicious codes are judged to be similar to each other according to the association support and minimum confidence. Furthermore, sequence discovery presented based on order opcode orders. The Rules Table displays information about each rule that was generated – the Support, Confidence and Lift, as well as the number of occurrences and the items in each rule. Which rules have the highest Support is matter in terms of knowledge discovery process. Figure illustrated the association rule for Gatak family, and part of rules yields the following Table 4-3.

Table 4-3 Sample rules from Gatak family analysis



RuleID	RULE
1	jnz ==> or
2	xor ==> sub
3	xor ==> push
4	xor ==> pop
5	xor ==> or
6	xor ==> mov
7	xor ==> lea
8	xor ==> jz
9	xor ==> jnz
10	xor ==> jnb

Since these values are different from one family to another one and such can be expressed as follows, see Figure 4-9.

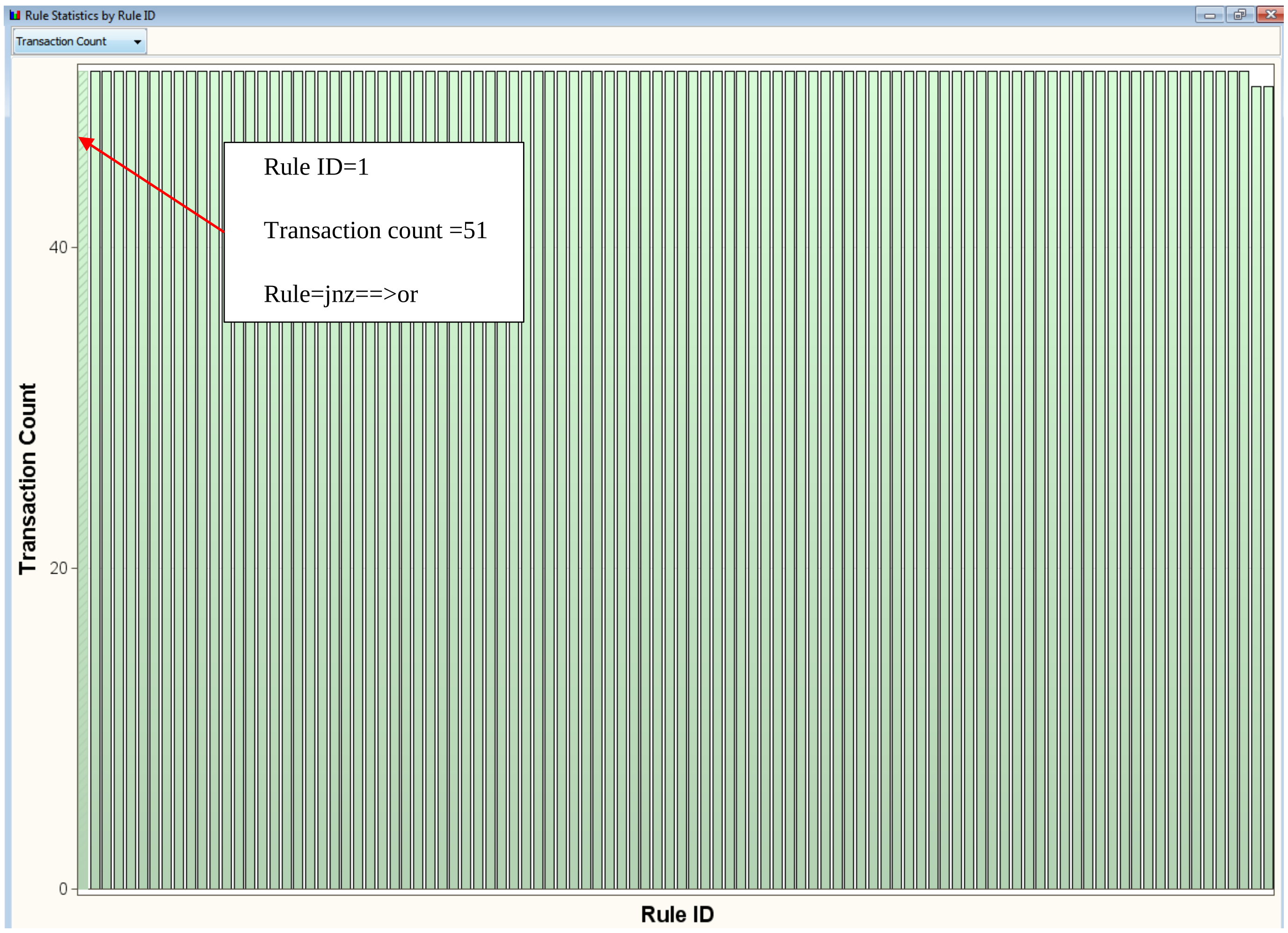


Figure 4-9Rule statistics-Gatak family

Sample exploratory plot of opcode ‘cmp’ is visible in the following figure.

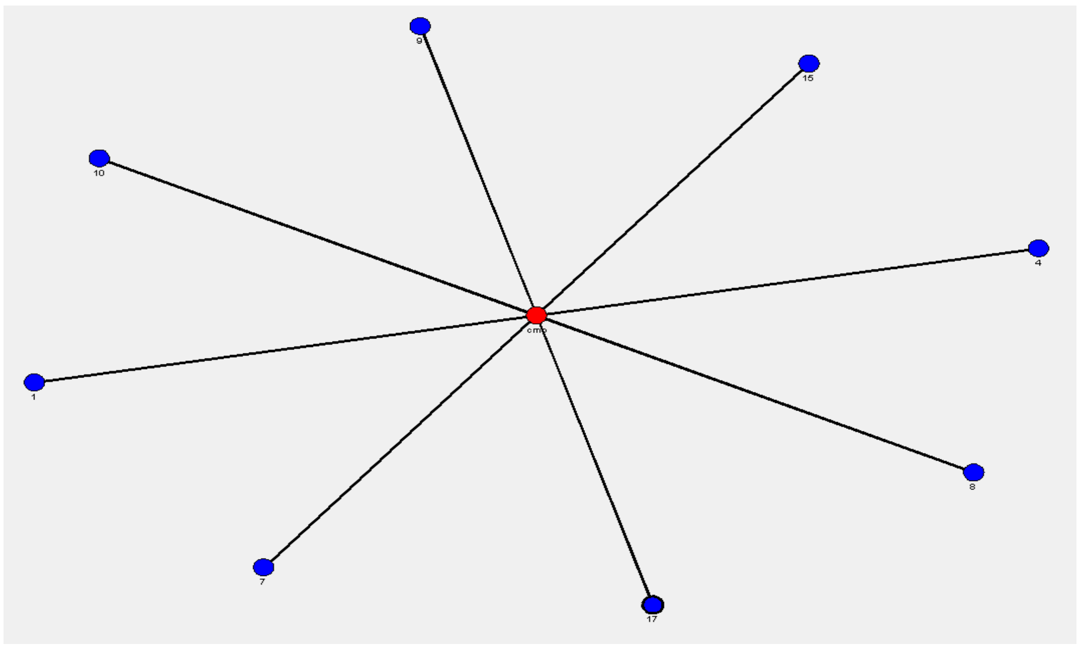


Figure 4-10 Opcode constellation plot for ‘cmp’

A few outliers in the data spotted in each plot, example provided in Figure 4-11.

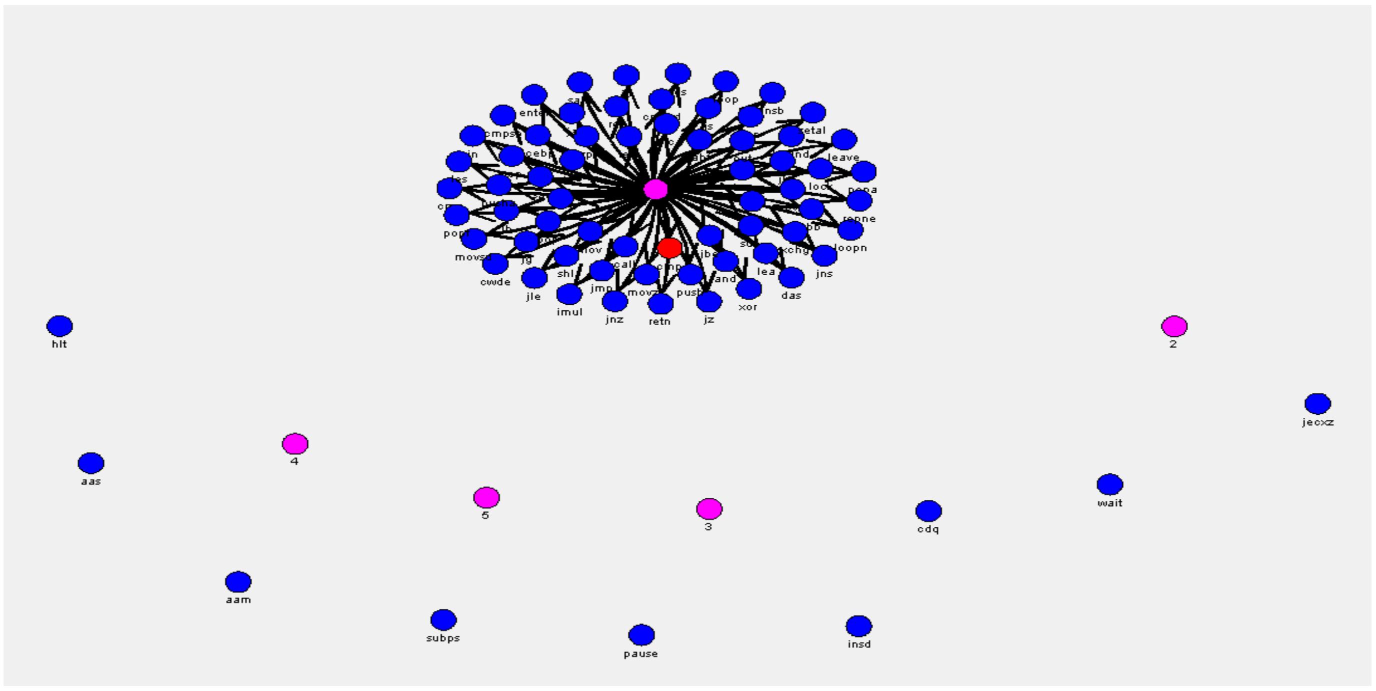
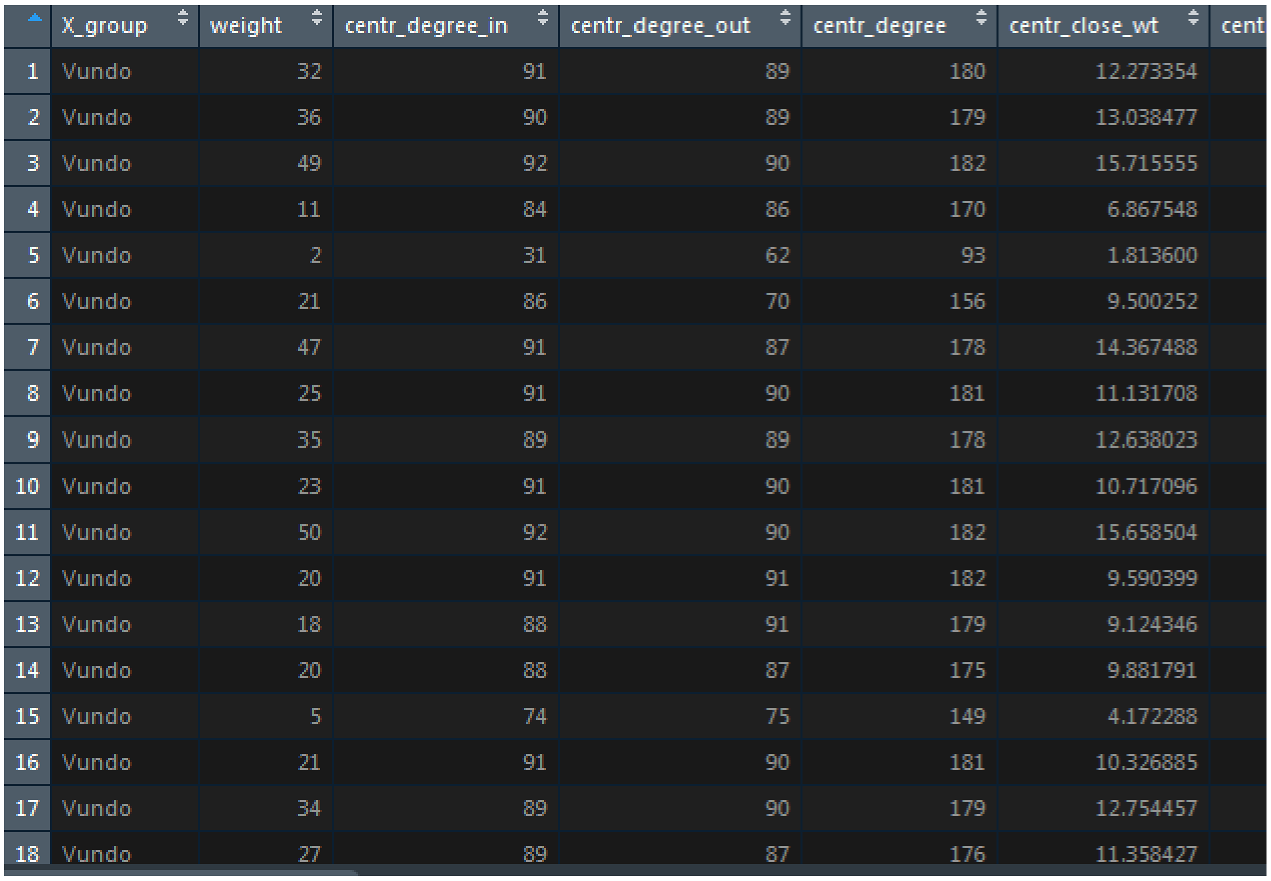


Figure 4-11 Relationship between cmp and other opcode based on rules

Finally, extracted weight centrality measures values is built Kaggle primary datasets.



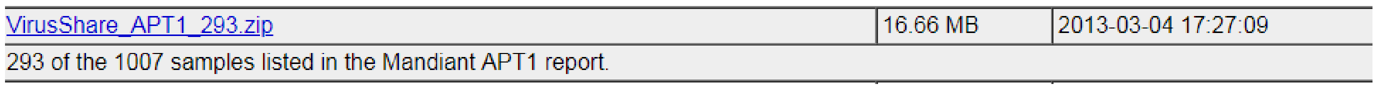
	X_group	weight	centr_degree_in	centr_degree_out	centr_degree	centr_close_wt	cent
1	Vundo	32	91	89	180	12.273354	
2	Vundo	36	90	89	179	13.038477	
3	Vundo	49	92	90	182	15.715555	
4	Vundo	11	84	86	170	6.867548	
5	Vundo	2	31	62	93	1.813600	
6	Vundo	21	86	70	156	9.500252	
7	Vundo	47	91	87	178	14.367488	
8	Vundo	25	91	90	181	11.131708	
9	Vundo	35	89	89	178	12.638023	
10	Vundo	23	91	90	181	10.717096	
11	Vundo	50	92	90	182	15.658504	
12	Vundo	20	91	91	182	9.590399	
13	Vundo	18	88	91	179	9.124346	
14	Vundo	20	88	87	175	9.881791	
15	Vundo	5	74	75	149	4.172288	
16	Vundo	21	91	90	181	10.326885	
17	Vundo	34	89	90	179	12.754457	
18	Vundo	27	89	87	176	11.358427	

Figure 4-12Kaggle Primarily dataset in RStudio

Additional reports are available in appendix A .These reports used interactively, nodes and kinks to provide more detailed information.

4.3 APTs Case Study

For current experiments, the primarily focus is on five dominant APTs groups, namely, group number 1, 28, 29, 30, and 33. A collection of malware samples from each group is 50 files. This is first collection of APTs samples which is available now for other researcher. Hashes of APTs group number 1 retrieved from VirusShare https://virusshare.com/torrents.4n6, see Figure 4-13, other hashes were collected from FireEye cyber threat intelligence reports. FireEye is an American company regularly publishes cyber threat intelligence reports that include hashes of APT groups https://www.fireeye.com/current-threats/threat-intelligence-reports.html.All corpus of APTs downloaded from VirustotalIntelligence. This is first time APTs samples were collected from different groups for research purpose. The dataset is available on GitHub.



VirusShare_APT1_293.zip	16.66 MB	2013-03-04 17:27:09
293 of the 1007 samples listed in the Mandiant APT1 report.		

Figure 4-13APT1 hash dataset

4.3.1 Dataset Characteristics and Pre-processing

Standard APTs groups according to FireEye is used in the research.A number of observations associated with each family presents in Table 4-4.

Table 4-4.APTs raw dataset

Group Name	Number of Opcode Observation
APTs1	244098
APTs28	1964498
APTs29	529931
APTs30	391935
APTs33	1000000

4.3.2 Experimental set up and Reports

The experiment setup for is the same as in the previous section. Link Analysis process diagram is shown below:

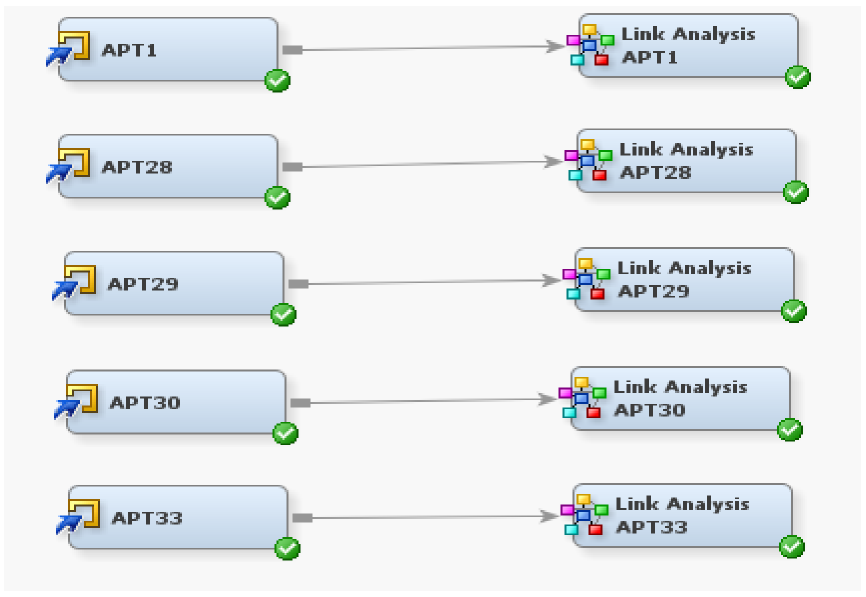
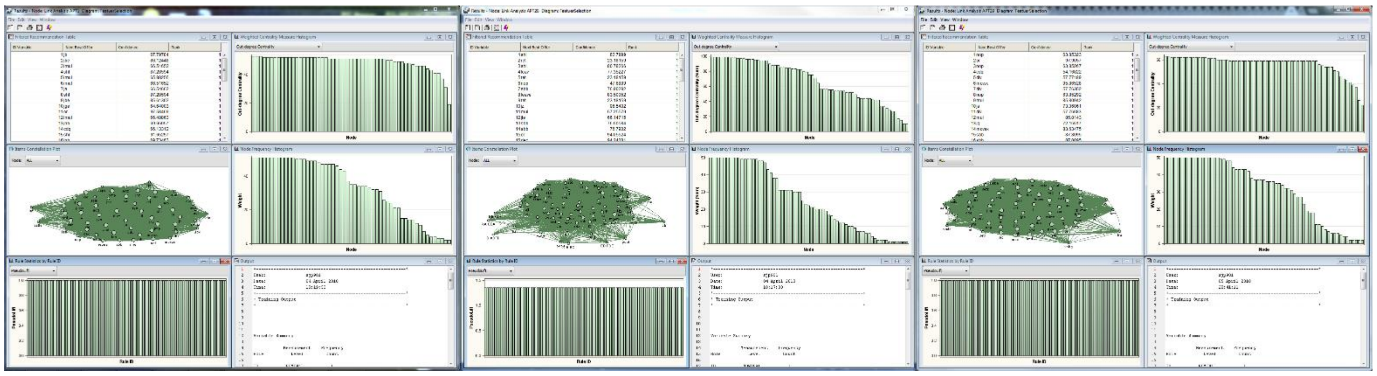
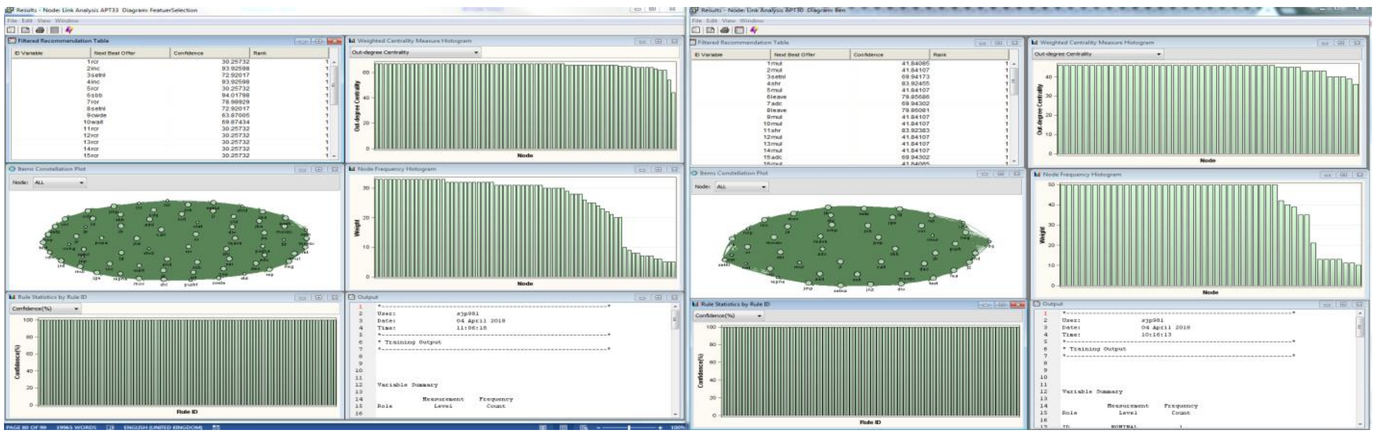


Figure 4-14APTs Link Analysis process diagram-APTs

After running the link analysis node, a variety of standard graphs can be displayed. The result of this model is illustrated below:





Results - Node: Link Analysis APT33 Diagram: FeatuerSelection

Filtered Recommendation Table

ID Variable	Next Best Offer	Confidence	Rank
1	rcr	30.25732	1
2	inc	93.92598	1
3	setnl	72.92017	1
4	inc	93.92598	1
5	rcr	30.25732	1
6	sbb	84.01798	1
7	ror	78.98929	1
8	setnl	72.92017	1
9	cwde	63.87005	1
10	wait	69.87434	1
11	rcr	30.25732	1
12	rcr	30.25732	1
13	rcr	30.25732	1
14	rcr	30.25732	1
15	rcr	30.25732	1

Output

```
*------------------------------------------------------------*
User:                  zjp981
Date:                  04 April 2018
Time:                  11:06:18
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary

               Measurement    Frequency
Role              Level         Count

```

Results - Node: Link Analysis APT30 Diagram: Ben

Filtered Recommendation Table

ID Variable	Next Best Offer	Confidence	Rank
1	mul	41.84085	1
2	mul	41.84107	1
3	setnl	69.94173	1
4	shr	83.92455	1
5	mul	41.84107	1
6	leave	79.85686	1
7	adc	69.94302	1
8	leave	79.86081	1
9	mul	41.84107	1
10	mul	41.84107	1
11	shr	83.92383	1
12	mul	41.84107	1
13	mul	41.84107	1
14	mul	41.84107	1
15	adc	69.94302	1
16	mul	41.84085	1

Output

```
*------------------------------------------------------------*
User:                  zjp981
Date:                  04 April 2018
Time:                  10:16:13
*------------------------------------------------------------*
* Training Output
*------------------------------------------------------------*



Variable Summary

               Measurement    Frequency
Role              Level         Count

ID              NOMINAL           1
```

Figure 4-15Link Analysis windows output- APTs dataset

The delivered weighted links and nodes from each APT family is presented in following pages (from Figure 4-16 to Figure 4-20.)

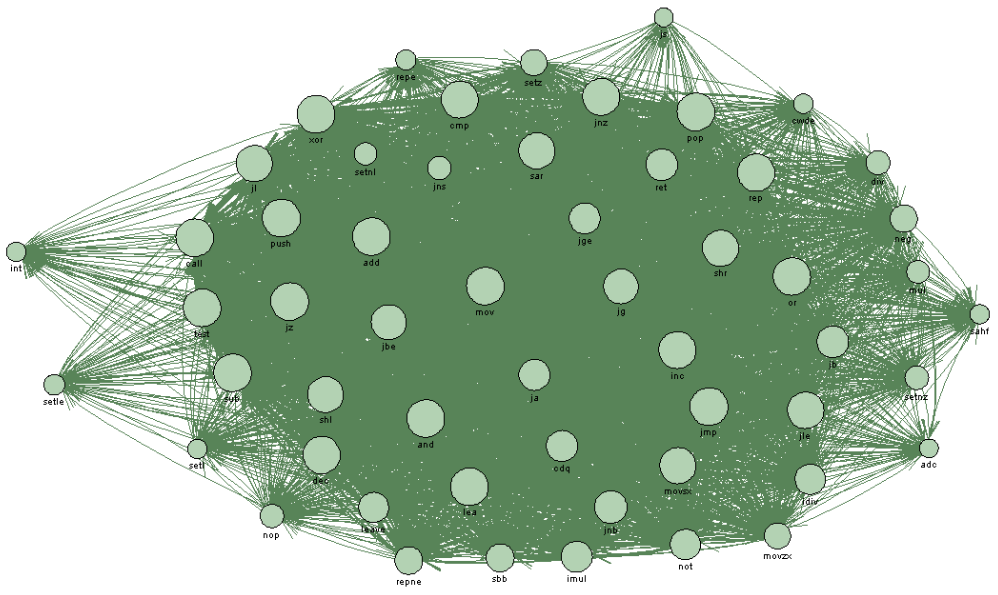


Figure 4-16 opcode Network -APT1

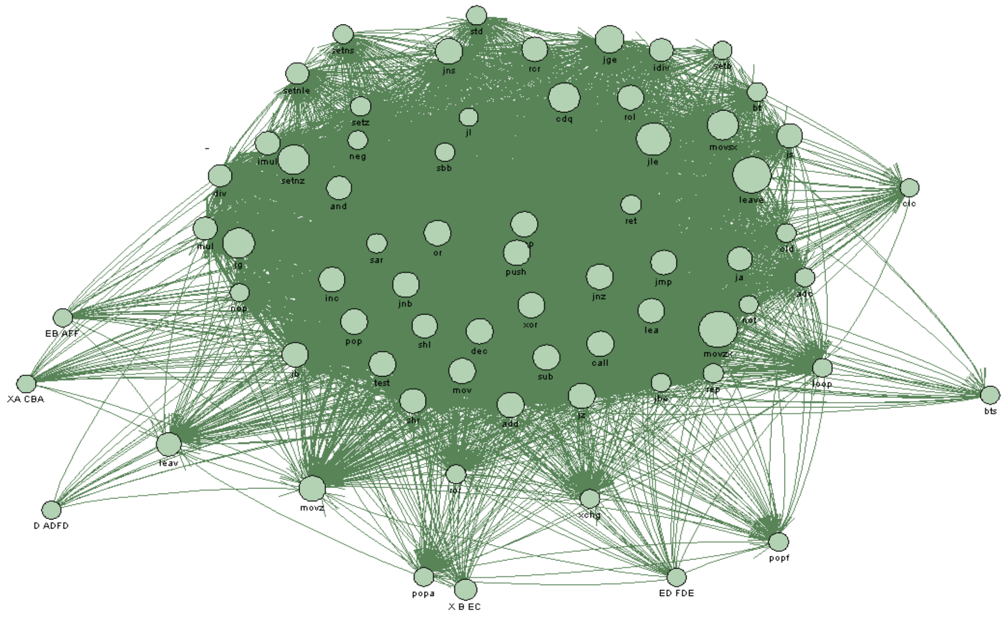


Figure 4-17 Opcode Network-APT28

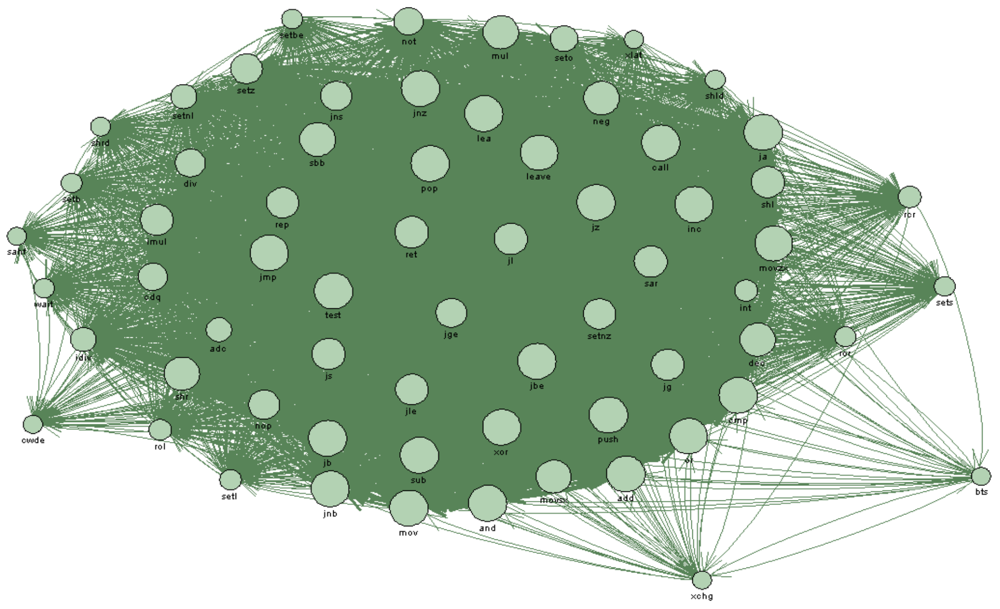


Figure 4-18Opcode Network-APT29

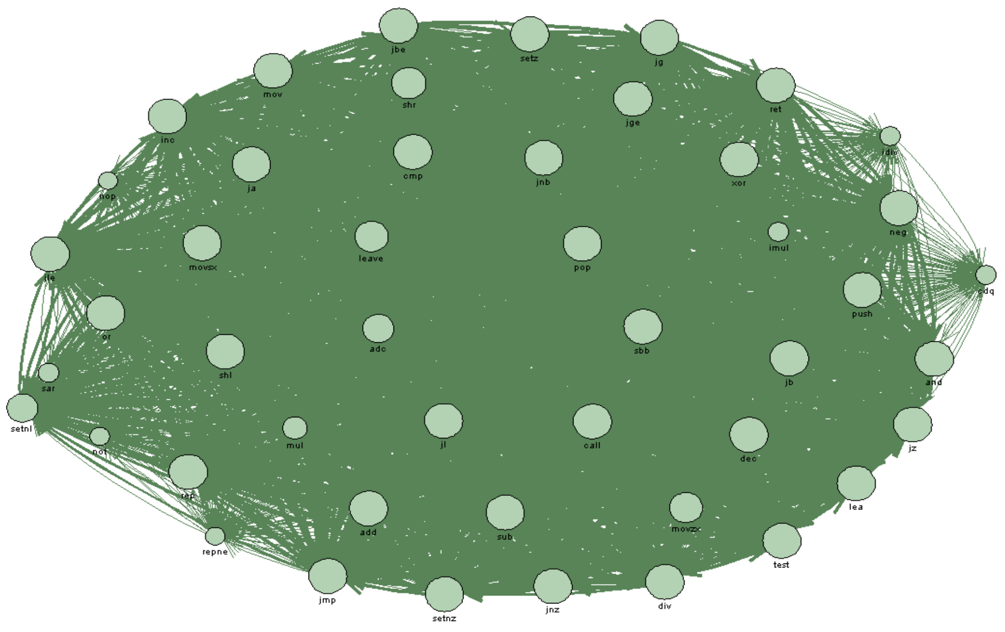


Figure 4-19 Opcode Network-APT30

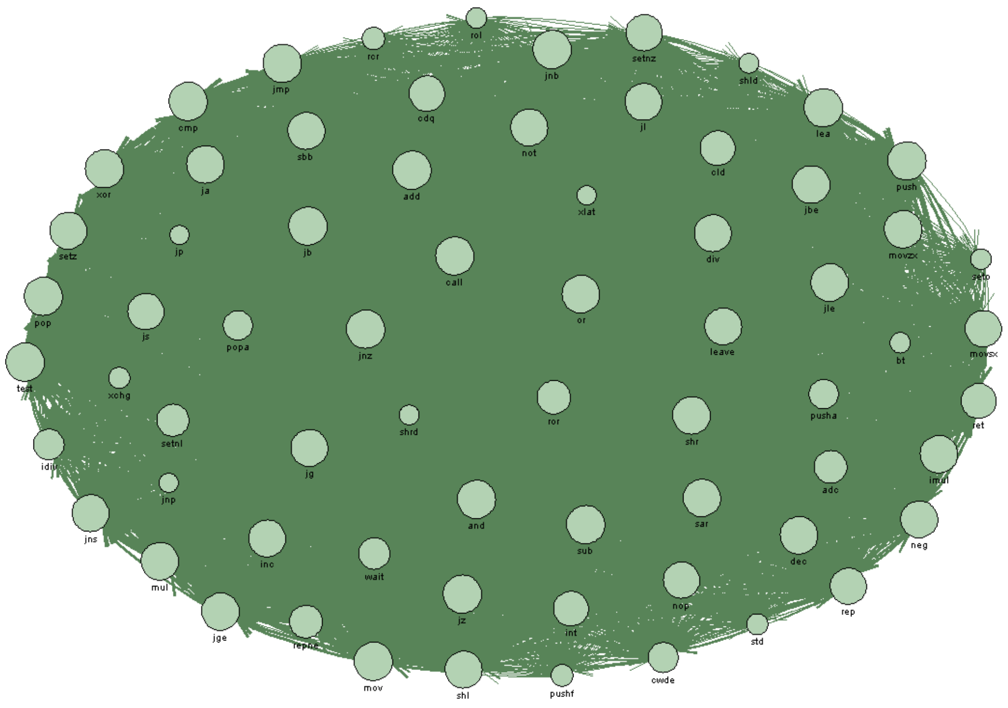


Figure 4-20Opcode Network-APT33

Once all the above processing has been completed, the primary dataset was uploaded into SAS for feature engineering purpose. Additional reports are available in appendix B. These reports used interactively, nodes and kinks to provide more detailed information.

4.4 Summary

With prepared primarily dataset of each case study we intended to identify whether constructed features can be segmented meaningfully in the view of malware classification. This objective successfully achieved by usage of variable clustering and segmentation algorithms in SAS.

Chapter Five

Models Assessment for Feature Engineering

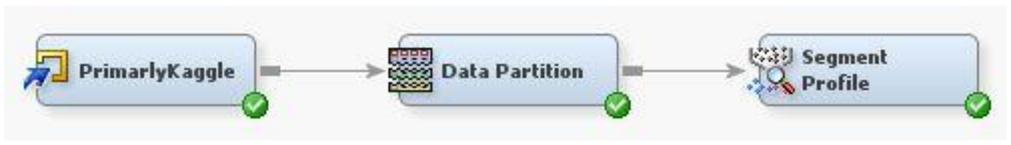
5 Overview

In this section, the performance of feature construction and selection models are investigated. To illustrate the usefulness of these models in feature engineering, we selected segmentation analysis and Gaussian Mixer Model. The performances are evaluated based which will be discussed further.

5.1 Validity of the Feature Construction Model

Ultimately, link analysis, like any data mining analysis, needs to prove its value as a feature construction method. To examine the efficiency and effectiveness of this method for malware classification segmentation analysis are used. Segmentation strategy across all dimensions execute and analyse the extracted feature relationships. In the context of business data mining, segmentation analysis has been applied for marketing or customer contact strategy(Y. Liu *et al.*, 2014) .The unlabelled data is partitioned correctly by this prediction method. Suppose a dataset consists of n malicious samples, the partitioning method construct k partition and k < n. It means that it will classify the malicious sample into k group, in a way that each group contains at least one sample. Beside, each malware must belong to exactly one group. The following diagrams created to represent clusters or segments, see Figure 5-1.

A common strategy prior to developing clusters or predictive models is to reduce the number of variables to a more manageable uncorrelated, non-redundant set. Finding interesting regularities in large malware datasets is can be possible by using decision Tree methodology. The segmentation analysis attempts to evaluate the overall value or importance of the variable over the fitted tree. For instance, if variable A has a variable importance higher than variable B then variable A can be said to have a larger impact on the clustering model. Keeping track of where each observation was placed and how each cluster is expanding can be achievable by variable clustering and segmentation techniques. The aim is to segregate groups with similar traits and assign them into clusters.



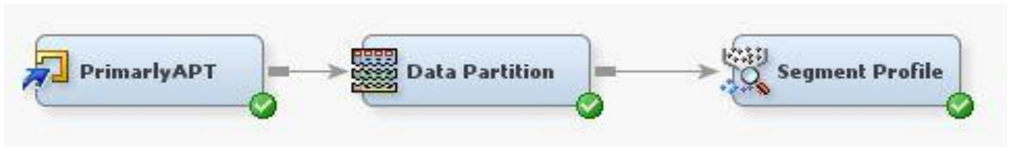
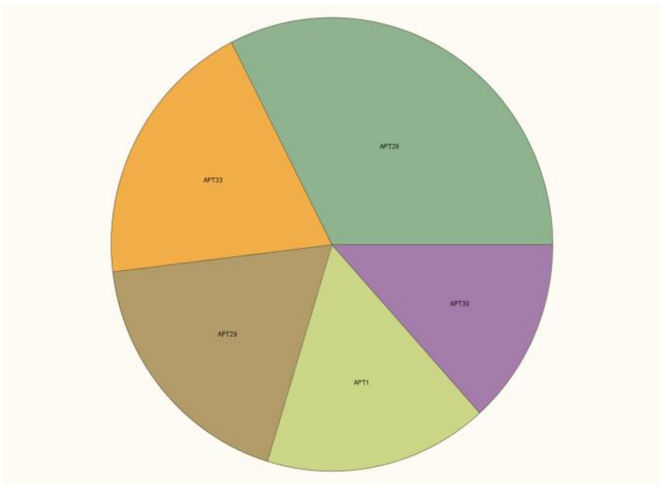


Figure 5-1Segmentation process diagram for both case studies

Segment Profile node scans clustered data and pulls out the factors that differentiate data segments (clusters/groups) from the population reflects the finding of the author(Thompson, 2008). As specified to the relationships between variables and observations in a dataset; objects will be grouped. The segmentation node selected nine clusters in Kaggle primary data and five in APTs primary data as seen in the segment size pie chart of Figure 5-2.



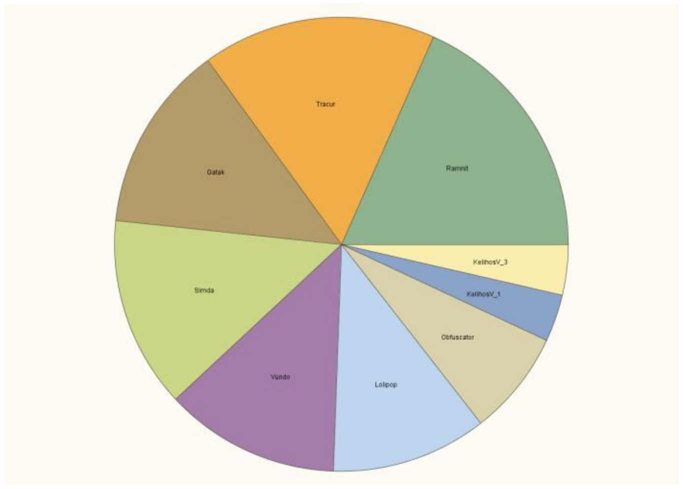
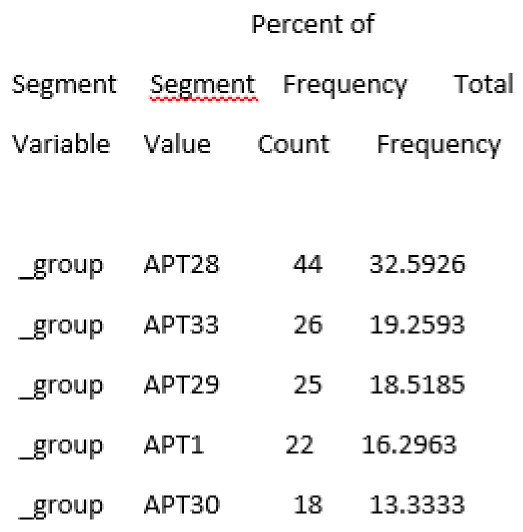


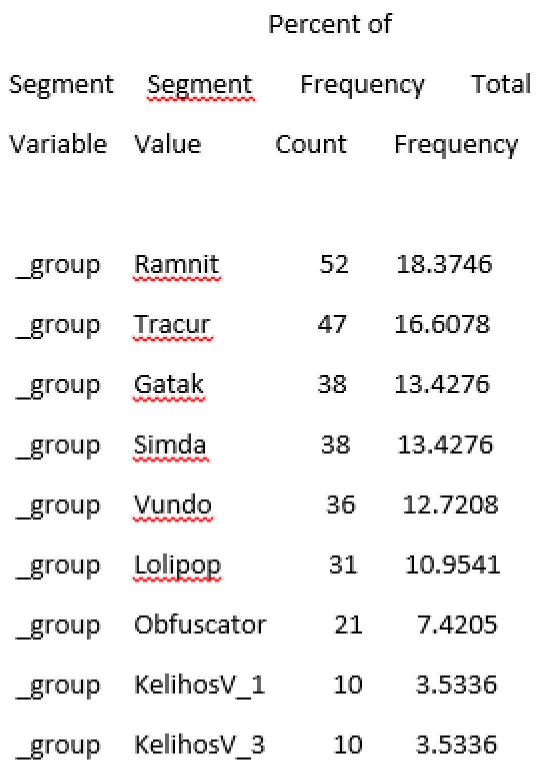
Figure 5-2 Segment Size: Cluster window generated by Segment Profile Node

Referring to Figure 4-2 and also Figure 4-15 a significant difference between families can be seen. A group of partitioning clustering algorithms is used to partition a data set of n objects with 17 features into 9 groups for Kaggle, 5 groups for APTs. The results indicate the ‘correct’ number of clusters correctly in both case studies. The figure shows the distribution of Kaggle observation based on first two weight centrality measurement features. Once an observation has been included in a cluster, it cannot be reassigned. The pie char can be scaled by count or by percentage of the segment population.

Summary of segment:



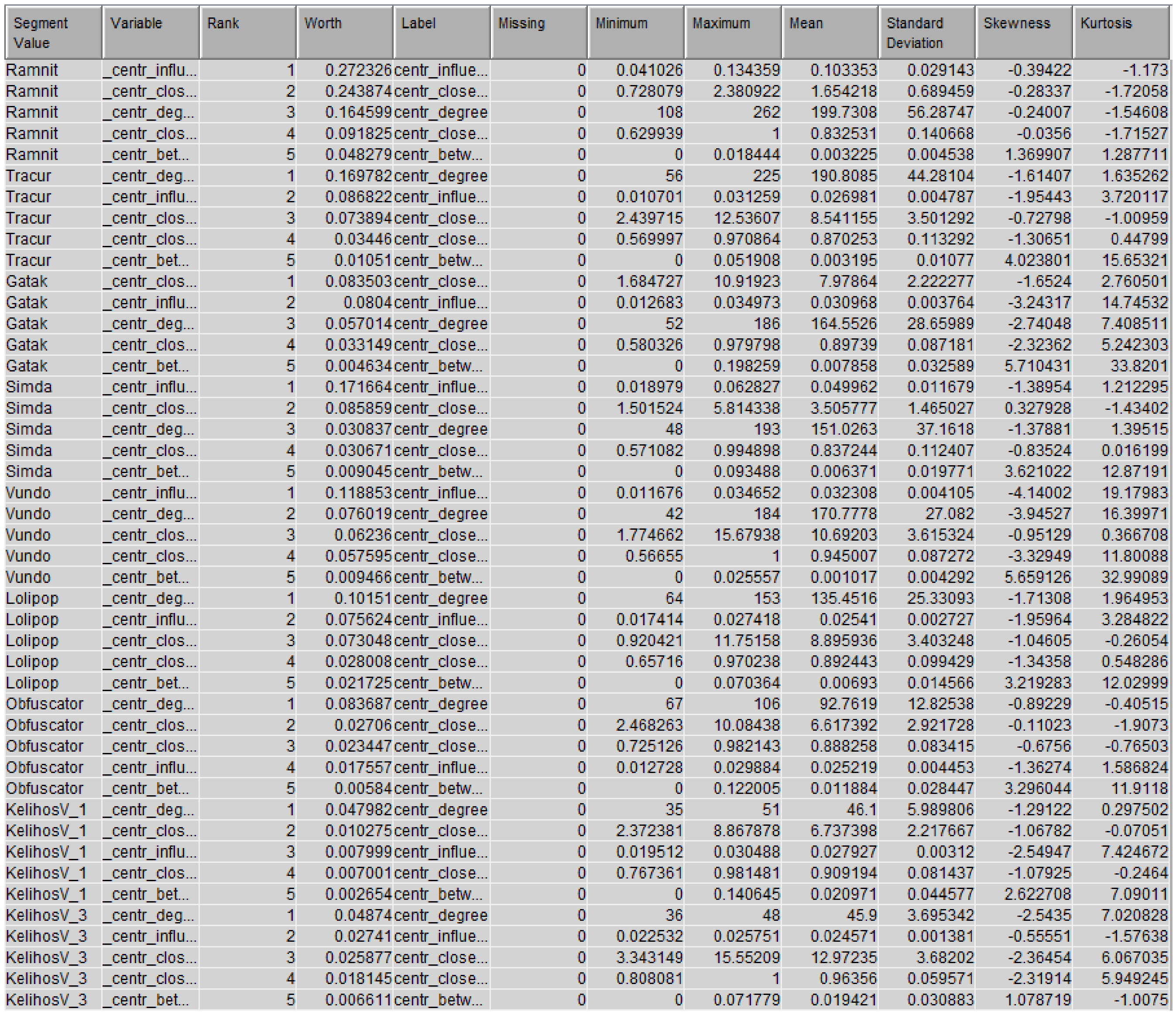
Segment Variable	Segment Value	Frequency Count	Percent of Total Frequency
_group	APT28	44	32.5926
_group	APT33	26	19.2593
_group	APT29	25	18.5185
_group	APT1	22	16.2963
_group	APT30	18	13.3333



Segment Variable	Segment Value	Frequency Count	Percent of Total Frequency
_group	Ramnit	52	18.3746
_group	Tracur	47	16.6078
_group	Gatak	38	13.4276
_group	Simda	38	13.4276
_group	Vundo	36	12.7208
_group	Lolipop	31	10.9541
_group	Obfuscator	21	7.4205
_group	KelihosV_1	10	3.5336
_group	KelihosV_3	10	3.5336

Segmentation analysis in SAS is based on factor analysis. It can perfume two methods to determine which variables can be employed to differentiate among segments. The first method works on binning the input variable to identify their maximum log-worth which is known as ‘variable worth’. Another one creates a Decision Tree to predict the segments from the inputs and use the Tree methodology of assessing ‘variable importance’. The first step is to form groups of attributes that express some sort of common theme. The number of factors is determined using a combination of statistics and knowledge of the category. Once the number of factors has been calculated, each respondent receives a score for each of the factors. Respondents are then assigned to the factor that has the highest score. Results of segmentation analysis create a subset of “important” features and find “meaningfully” related features among malware families. Partial results of factor segmentation classification are shown in Table 5-1.

Table 5-1 Kaggle cluster metrics



Segment Value	Variable	Rank	Worth	Label	Missing	Minimum	Maximum	Mean	Standard Deviation	Skewness	Kurtosis
Ramnit	_centr_influ...	1	0.272326	centr_influe...	0	0.041026	0.134359	0.103353	0.029143	-0.39422	-1.173
Ramnit	_centr_clos...	2	0.243874	centr_close...	0	0.728079	2.380922	1.654218	0.689459	-0.28337	-1.72058
Ramnit	_centr_deg...	3	0.164599	centr_degree	0	108	262	199.7308	56.28747	-0.24007	-1.54608
Ramnit	_centr_clos...	4	0.091825	centr_close...	0	0.629939	1	0.832531	0.140668	-0.0356	-1.71527
Ramnit	_centr_bet...	5	0.048279	centr_betw...	0	0	0.018444	0.003225	0.004538	1.369907	1.287711
Tracur	_centr_deg...	1	0.169782	centr_degree	0	56	225	190.8085	44.28104	-1.61407	1.635262
Tracur	_centr_influ...	2	0.086822	centr_influe...	0	0.010701	0.031259	0.026981	0.004787	-1.95443	3.720117
Tracur	_centr_clos...	3	0.073894	centr_close...	0	2.439715	12.53607	8.541155	3.501292	-0.72798	-1.00959
Tracur	_centr_clos...	4	0.03446	centr_close...	0	0.569997	0.970864	0.870253	0.113292	-1.30651	0.44799
Tracur	_centr_bet...	5	0.01051	centr_betw...	0	0	0.051908	0.003195	0.01077	4.023801	15.65321
Gatak	_centr_clos...	1	0.083503	centr_close...	0	1.684727	10.91923	7.97864	2.222277	-1.6524	2.760501
Gatak	_centr_influ...	2	0.0804	centr_influe...	0	0.012683	0.034973	0.030968	0.003764	-3.24317	14.74532
Gatak	_centr_deg...	3	0.057014	centr_degree	0	52	186	164.5526	28.65989	-2.74048	7.408511
Gatak	_centr_clos...	4	0.033149	centr_close...	0	0.580326	0.979798	0.89739	0.087181	-2.32362	5.242303
Gatak	_centr_bet...	5	0.004634	centr_betw...	0	0	0.198259	0.007858	0.032589	5.710431	33.8201
Simda	_centr_influ...	1	0.171664	centr_influe...	0	0.018979	0.062827	0.049962	0.011679	-1.38954	1.212295
Simda	_centr_clos...	2	0.085859	centr_close...	0	1.501524	5.814338	3.505777	1.465027	0.327928	-1.43402
Simda	_centr_deg...	3	0.030837	centr_degree	0	48	193	151.0263	37.1618	-1.37881	1.39515
Simda	_centr_clos...	4	0.030671	centr_close...	0	0.571082	0.994898	0.837244	0.112407	-0.83524	0.016199
Simda	_centr_bet...	5	0.009045	centr_betw...	0	0	0.093488	0.006371	0.019771	3.621022	12.87191
Vundo	_centr_influ...	1	0.118853	centr_influe...	0	0.011676	0.034652	0.032308	0.004105	-4.14002	19.17983
Vundo	_centr_deg...	2	0.076019	centr_degree	0	42	184	170.7778	27.082	-3.94527	16.39971
Vundo	_centr_clos...	3	0.06236	centr_close...	0	1.774662	15.67938	10.69203	3.615324	-0.95129	0.366708
Vundo	_centr_clos...	4	0.057595	centr_close...	0	0.56655	1	0.945007	0.087272	-3.32949	11.80088
Vundo	_centr_bet...	5	0.009466	centr_betw...	0	0	0.025557	0.001017	0.004292	5.659126	32.99089
Lolipop	_centr_deg...	1	0.10151	centr_degree	0	64	153	135.4516	25.33093	-1.71308	1.964953
Lolipop	_centr_influ...	2	0.075624	centr_influe...	0	0.017414	0.027418	0.02541	0.002727	-1.95964	3.284822
Lolipop	_centr_clos...	3	0.073048	centr_close...	0	0.920421	11.75158	8.895936	3.403248	-1.04605	-0.26054
Lolipop	_centr_clos...	4	0.028008	centr_close...	0	0.65716	0.970238	0.892443	0.099429	-1.34358	0.548286
Lolipop	_centr_bet...	5	0.021725	centr_betw...	0	0	0.070364	0.00693	0.014566	3.219283	12.02999
Obfuscator	_centr_deg...	1	0.083687	centr_degree	0	67	106	92.7619	12.82538	-0.89229	-0.40515
Obfuscator	_centr_clos...	2	0.02706	centr_close...	0	2.468263	10.08438	6.617392	2.921728	-0.11023	-1.9073
Obfuscator	_centr_clos...	3	0.023447	centr_close...	0	0.725126	0.982143	0.888258	0.083415	-0.6756	-0.76503
Obfuscator	_centr_influ...	4	0.017557	centr_influe...	0	0.012728	0.029884	0.025219	0.004453	-1.36274	1.586824
Obfuscator	_centr_bet...	5	0.00584	centr_betw...	0	0	0.122005	0.011884	0.028447	3.296044	11.9118
KelihosV_1	_centr_deg...	1	0.047982	centr_degree	0	35	51	46.1	5.989806	-1.29122	0.297502
KelihosV_1	_centr_clos...	2	0.010275	centr_close...	0	2.372381	8.867878	6.737398	2.217667	-1.06782	-0.07051
KelihosV_1	_centr_influ...	3	0.007999	centr_influe...	0	0.019512	0.030488	0.027927	0.00312	-2.54947	7.424672
KelihosV_1	_centr_clos...	4	0.007001	centr_close...	0	0.767361	0.981481	0.909194	0.081437	-1.07925	-0.2464
KelihosV_1	_centr_bet...	5	0.002654	centr_betw...	0	0	0.140645	0.020971	0.044577	2.622708	7.09011
KelihosV_3	_centr_deg...	1	0.04874	centr_degree	0	36	48	45.9	3.695342	-2.5435	7.020828
KelihosV_3	_centr_influ...	2	0.02741	centr_influe...	0	0.022532	0.025751	0.024571	0.001381	-0.55551	-1.57638
KelihosV_3	_centr_clos...	3	0.025877	centr_close...	0	3.343149	15.55209	12.97235	3.68202	-2.36454	6.067035
KelihosV_3	_centr_clos...	4	0.018145	centr_close...	0	0.808081	1	0.96356	0.059571	-2.31914	5.949245
KelihosV_3	_centr_bet...	5	0.006611	centr_betw...	0	0	0.071779	0.019421	0.030883	1.078719	-1.0075

The segmentation process used to select the best subset of opcodes sequence. To deal with high dimensionality, only variables with the highest score used in the secondary dataset. Segment Profile results with aligned reporting variables exhibit in Variables (segment vs overall) are displayed horizontally in descending order of importance for each segment/cluster. The distribution charts are displayed for the most significant variables in each segment, which can be different for each. The columns are well ordered from left to right according to their ability to discriminate that segment from the population. The green shaded region represents the within-segment distribution. The orange outline represents the population distribution.

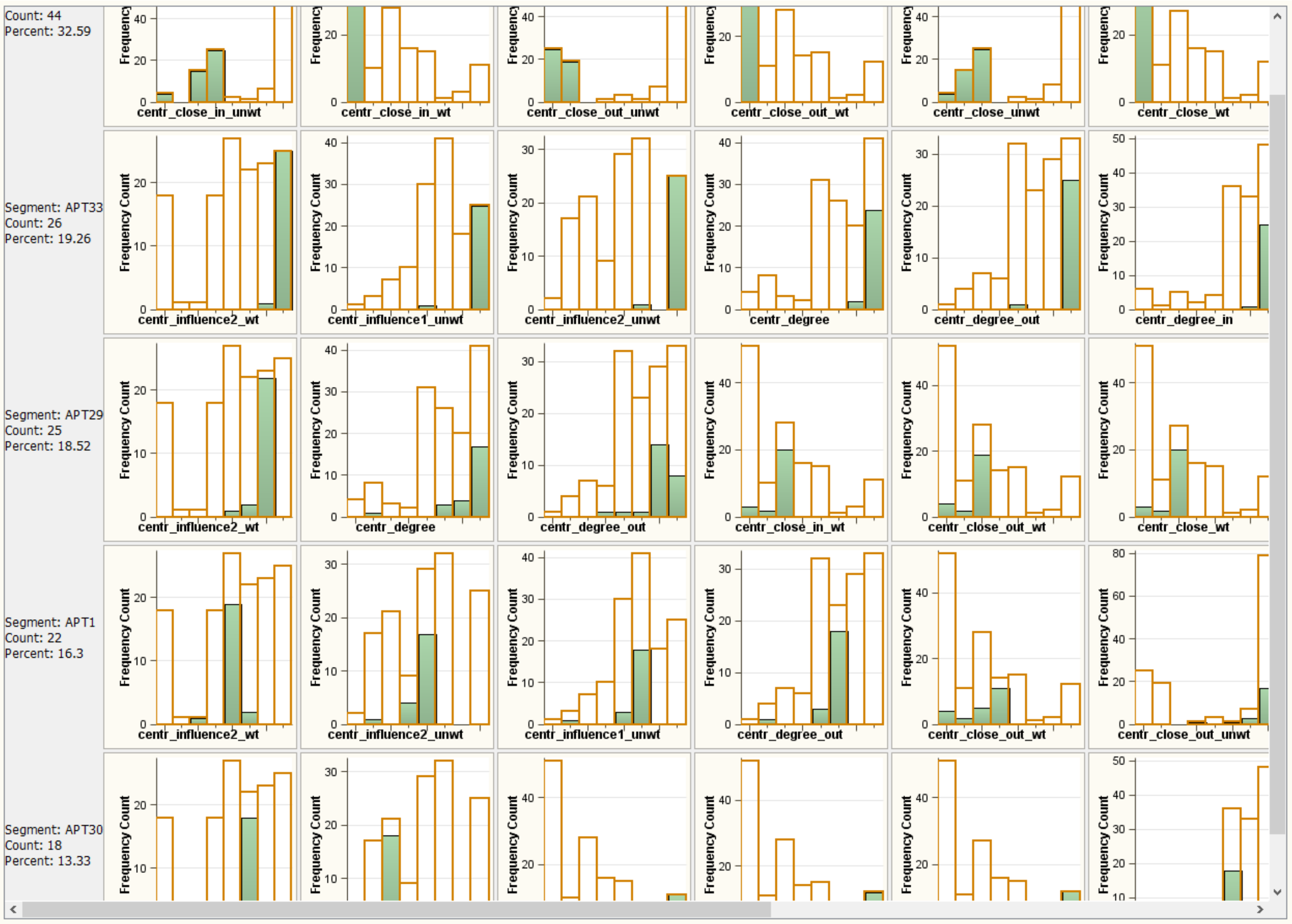
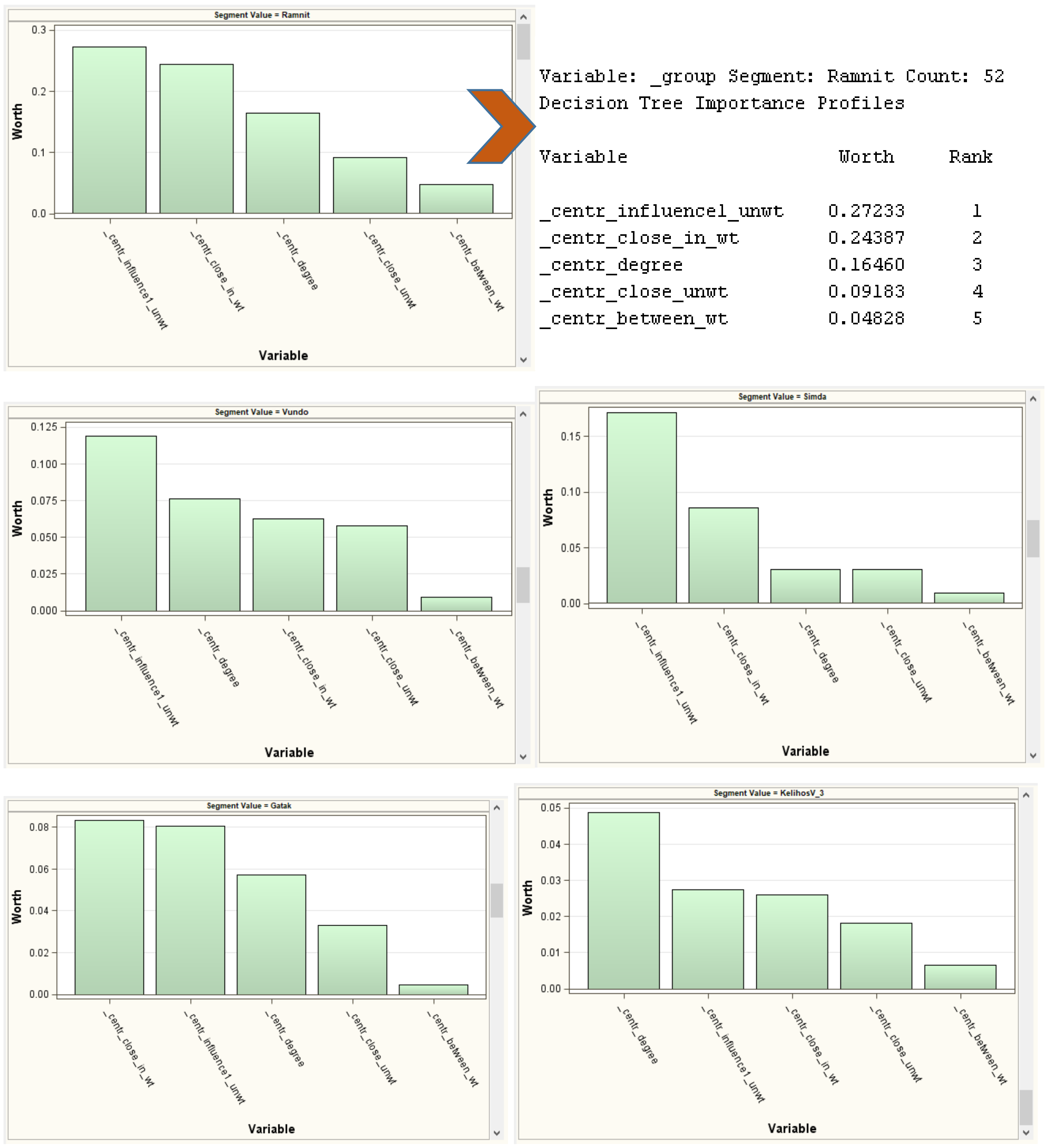


Figure 5-3 Segment Profile Distribution

As can be seen from the plat the following five variables have been chosen: *_center_influencel_unwt*, *_center_close_in_wt*, *_center_dgree*,*_center_close_unwt*,*_center_between_wt*. It is the above bar graph that highest worth variable is *_center_influencel_unwt*, and the lowest is *_center_between_wt*. The present finding reflects in the following pages (Figure 5-4 and Figure 5-5)



```
Variable: _group Segment: Ramnit Count: 52
Decision Tree Importance Profiles

Variable                   Worth     Rank

_centr_influence1_unwt    0.27233      1
_centr_close_in_wt        0.24387      2
_centr_degree             0.16460      3
_centr_close_unwt         0.09183      4
_centr_between_wt         0.04828      5
```

Figure 5-4Segment Profile Outputs -Kaggle Segment

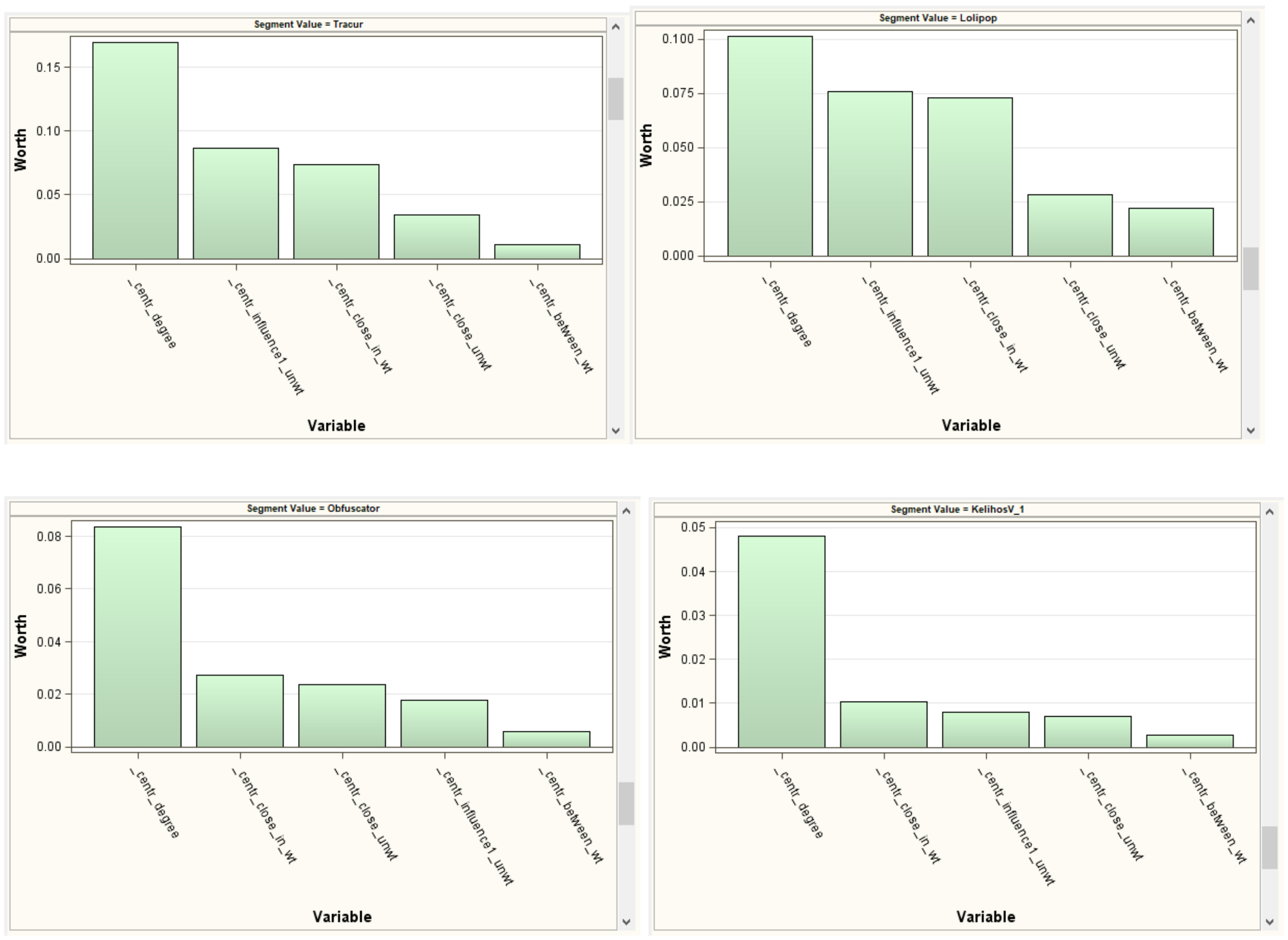


Figure 5-5 Segment Profile Outputs - Kaggle Segment

The output shows the decision tree results of importance for the variable by each segment. The variables which have been grouped together show obvious similarity relationships. The similarity relationships provide a very good idea of which variables are dominant in the Ramnit segment. The variable importance in the Segment Profile node is based on the relationship to the Segment variable. When the depth of the decision tree used to differentiate among variables is 1, the ‘variable worth’ is used to rank the variables. If not, ‘variable importance’ is used the rank the variables. A comparison of the outputs is furnished the variables sub set that can sufficiently help to build cluster and predictive model. These variables can be segmented various type of malware that has similar opcode structure in each family. From here, the important variables are pulled out which separate each individual malware family by their worth variables. The following shows worth variables chart of APTs dataset (from Figure 5-6 to Figure 5-10).

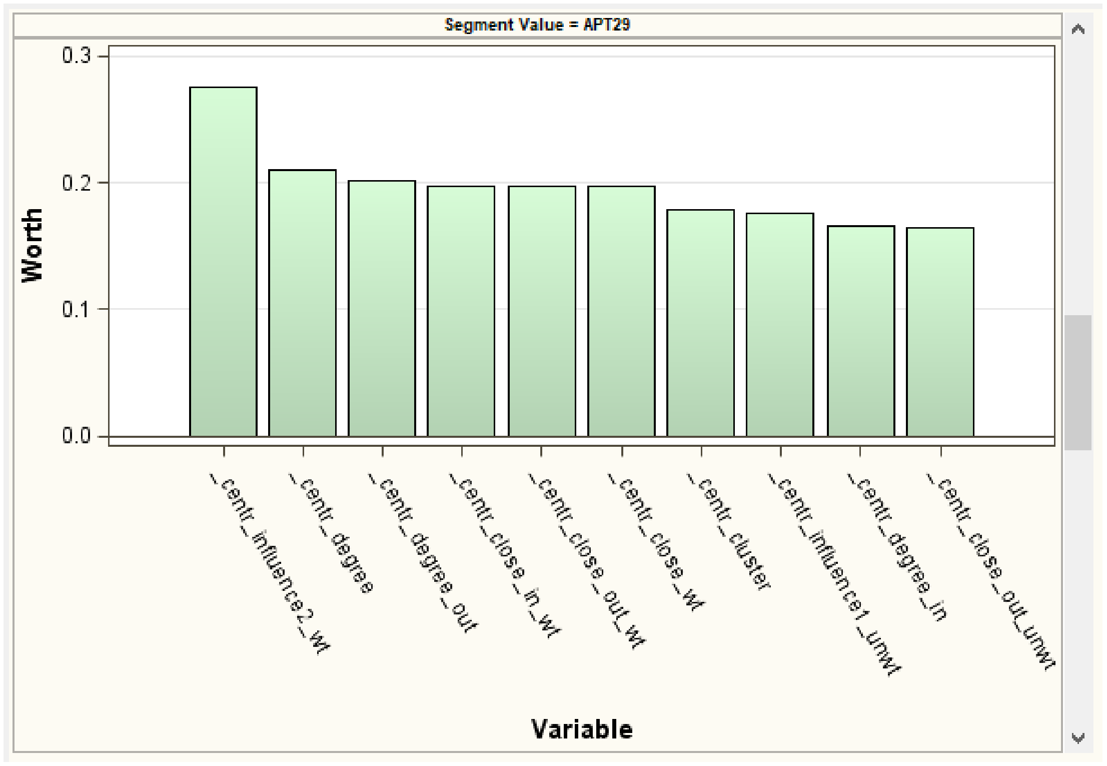


Figure 5-6 Segment Value-APT29

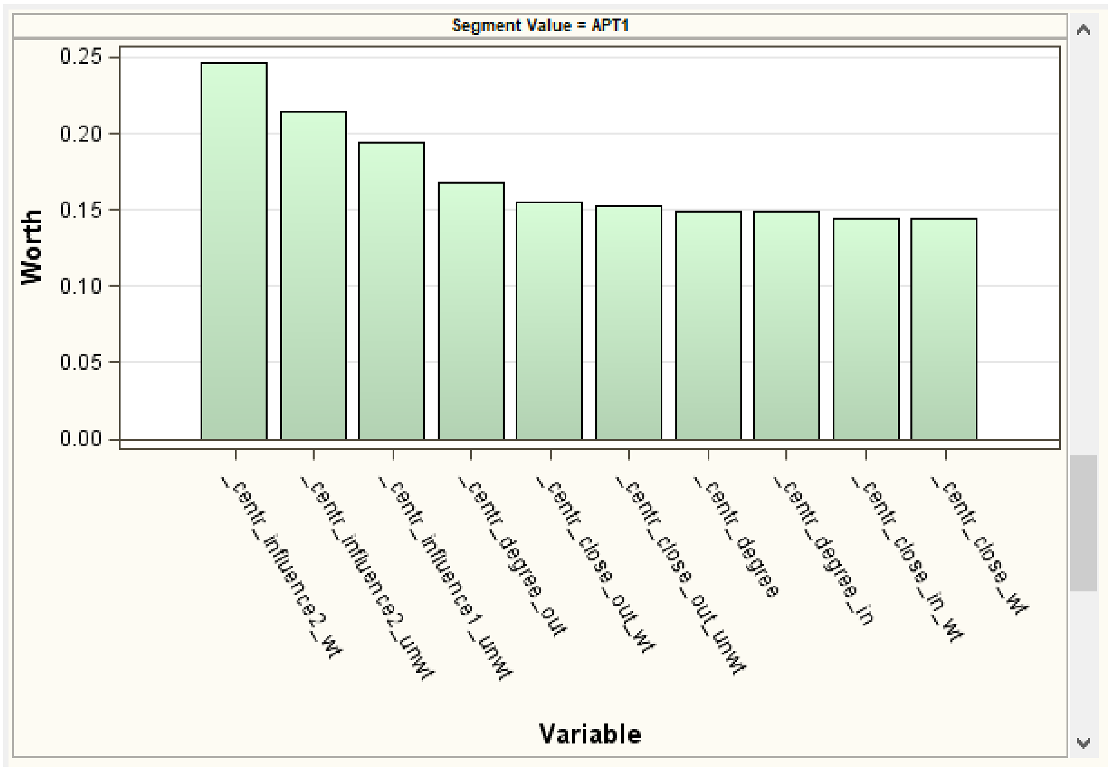


Figure 5-7 Segment Value-APT1

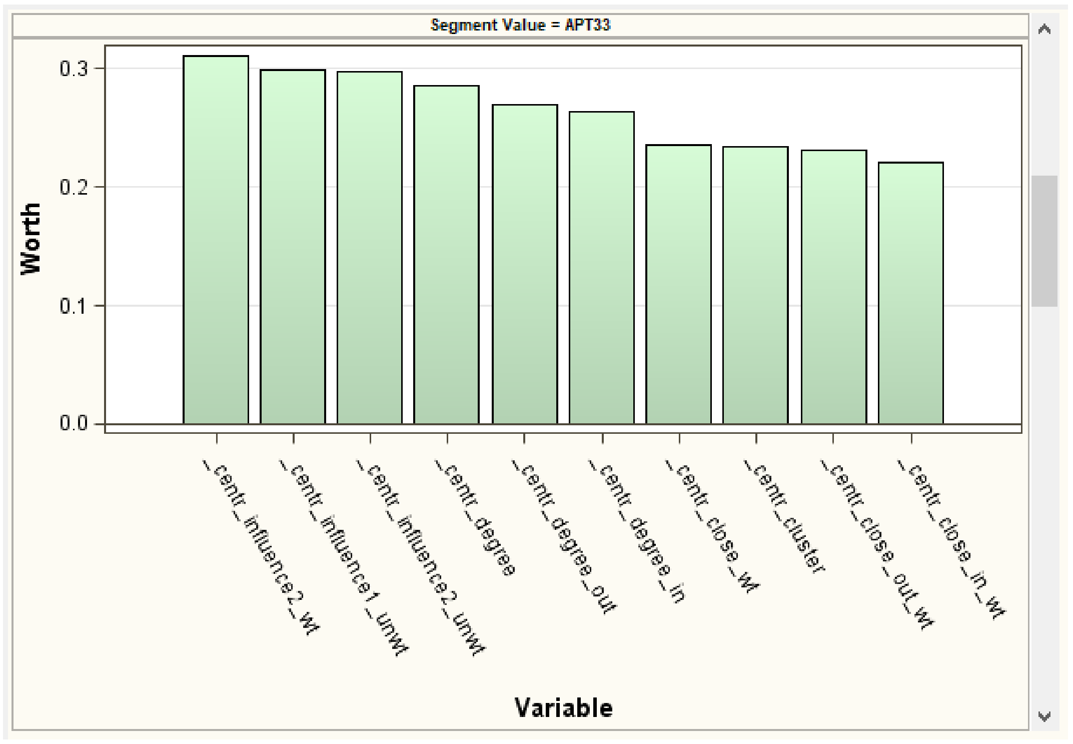


Figure 5-8 Segment Value-APT33

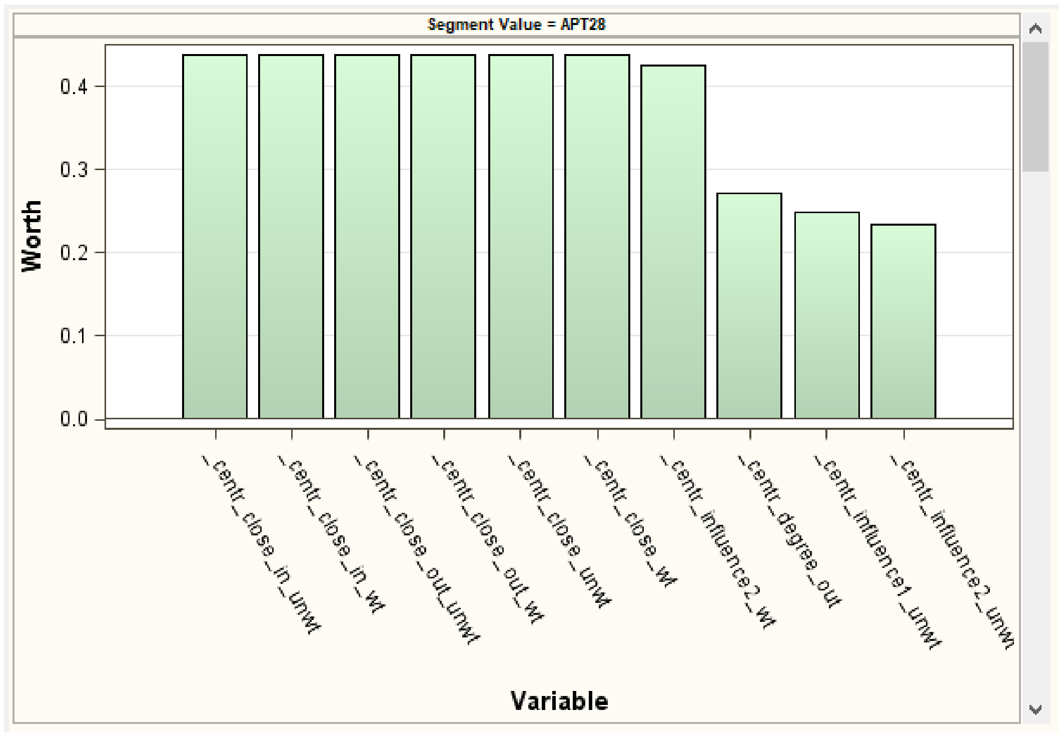


Figure 5-9 Segment Value-APT28

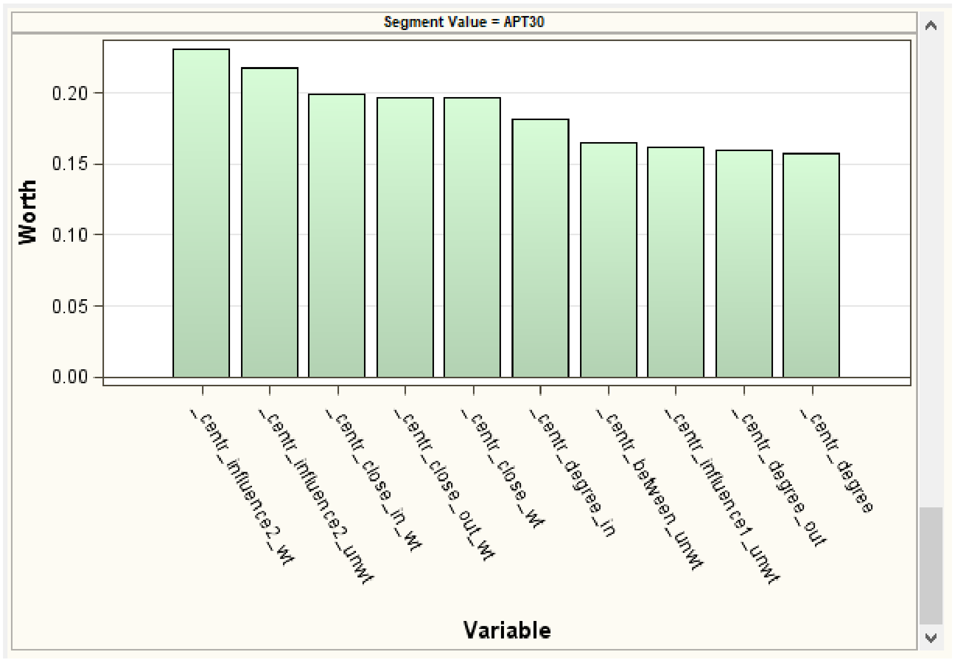


Figure 5-10Segment Value-APT30

As can be seen from the plat the following ten variables have been chosen: *_center_influencel2_wt ,_center_influencel2_unwt, _center_close_in_wt, _center_dgree,_center_close_wt,_center_between_unwt. _centr_close_wt ,_centr_between_unwt , center_influencel1_unwt, _centr_degree_out , _centr_degree_in* will constructed the final APT dataset.

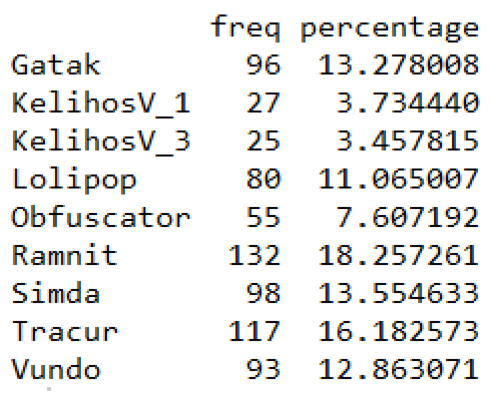
Summarization Kaggle class distribution generated by data segmentation node is presented in Table 5-2.

Table 5-2 Class Distribution in Secondary Kaggle Dataset

Segment Value	Frequency Count	Percent of Total Frequency
Ramnit	52	18.3746
Tracur	47	16.6078
Gatak	38	13.4276
Simda	38	13.4276
Vundo	36	12.7208
Lolipop	31	10.9541
Obfuscator	21	7.4205
KelihosV_1	10	3.5336
KelihosV_3	10	3.5336

There is slight difference between the class distributions in actual vs secondary dataset, see Table 5-3.

Table 5-3 Class Distribution in Actual Kaggle Dataset

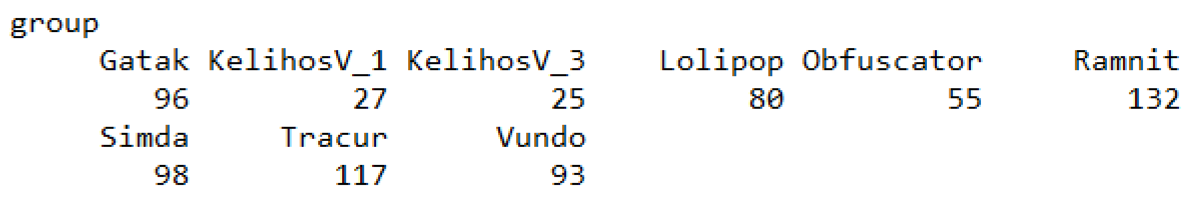


```
             freq percentage
Gatak          96  13.278008
KelihosV_1     27   3.734440
KelihosV_3     25   3.457815
Lolipop        80  11.065007
Obfuscator     55   7.607192
Ramnit        132  18.257261
Simda          98  13.554633
Tracur        117  16.182573
Vundo          93  12.863071
```

Following this procedure, a target dataset for testing classification performance is constructed in an Excel file for each dataset.

5.2 Validity of the Feature Selection Model

Validation of feature selection performance it can be a difficult task. There are multiple methods to understand the goodness of the proposed model for clustering task. To test how well the segmentation model partitioned malware observation, Gaussian Mixture Model (GMM) is employed. It utilized to determine that selected features inherent to malware families are interesting features. The mixture models not only can estimate the density function but also can provide a probabilistic clustering of the observed data i (i = 1, ..., n) into k clusters. Gaussian model is developed on the Bayesian theory (Tree theory) which is based on counting the relative occurrences of observations. K-fold cross-validation is measure too. The result of it shows the discriminant analysis based on Gaussian finite mixture modelling. Gaussian finite mixture model fitted by EM algorithm is run. In the following steps we shall show how GMM operations split the final set of distribution. SAS will not implement model-based clustering algorithms. With R, Mclust package loaded and the final Kaggle dataset is imported in R as shown in Figure 5-11.



```
group
     Gatak KelihosV_1 KelihosV_3    Lolipop Obfuscator      Ramnit
        96         27         25         80         55         132
     Simda      Tracur      Vundo
        98         117         93
```

Figure 5-11 Number of Observation of Kaggle dataset

The model-based clustering performed on primary and final Kaggle dataset, visible in Figure 5-12 and Figure 5-13.

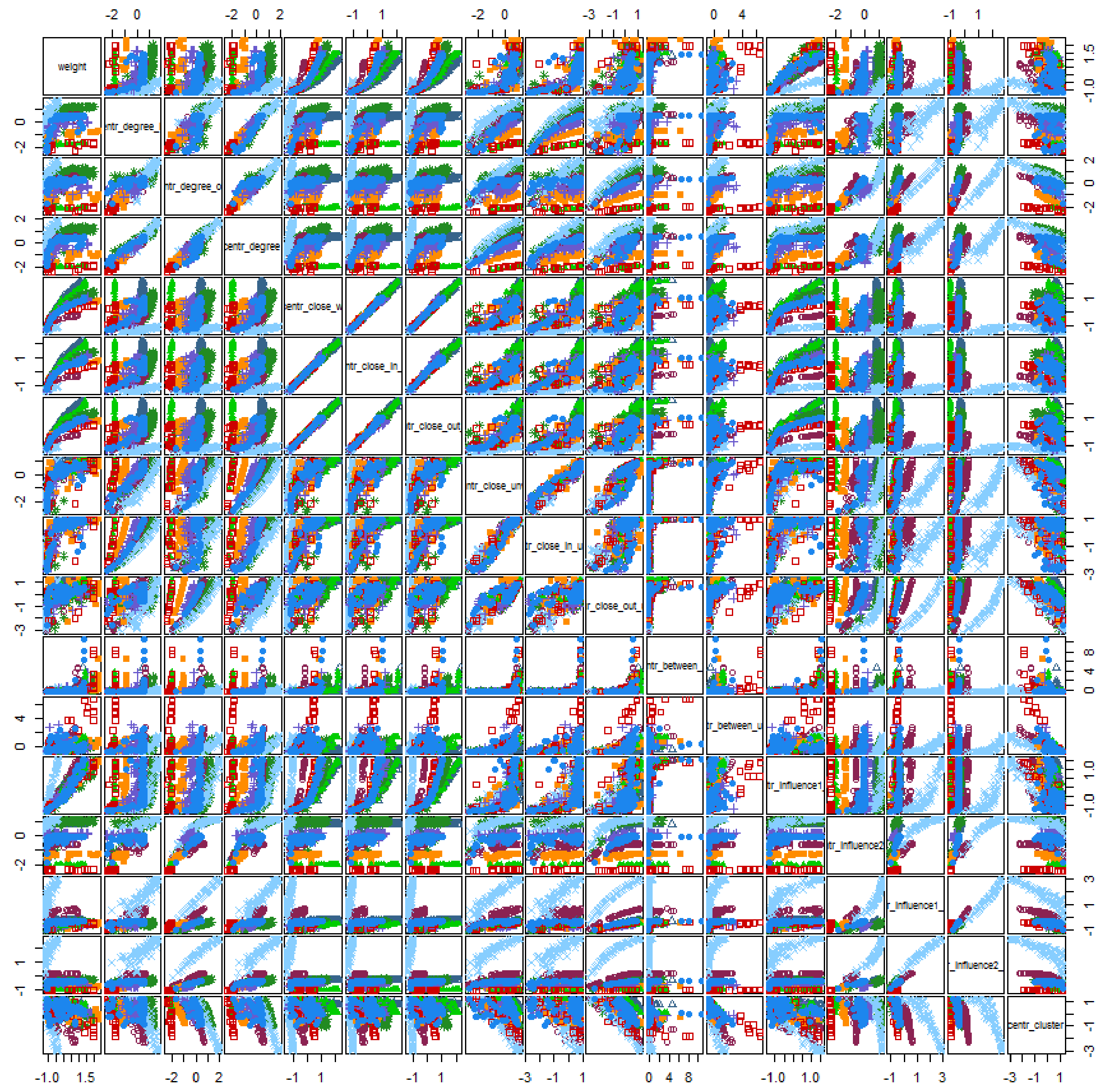


Figure 5-12 Classification Model on Primary Kaggle Dataset

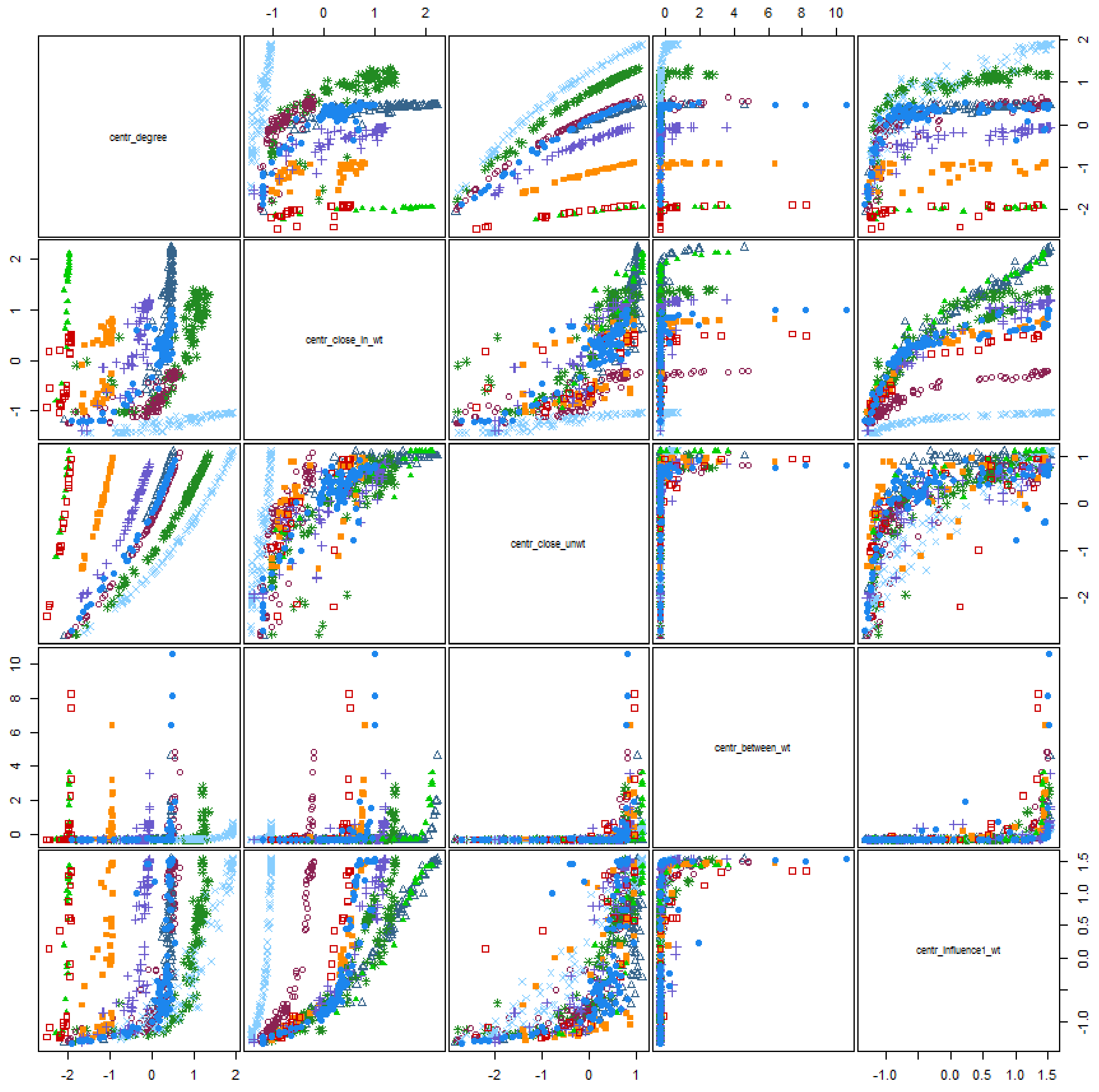


Figure 5-13 Classification model on final Kaggle dataset

To select which log-likelihoods should be used, the function Bayesian Information Criterion (BIC) is run. According to BIC from Mclust (R package) for the 6 available model parameterizations and up to 9 clusters for the Kaggle dataset. Different symbols and line types encode different model parameterizations. The 'best' model is taken to be the one with the highest BIC among the fitted models. The resulting plot is shown in Figure 5-14.

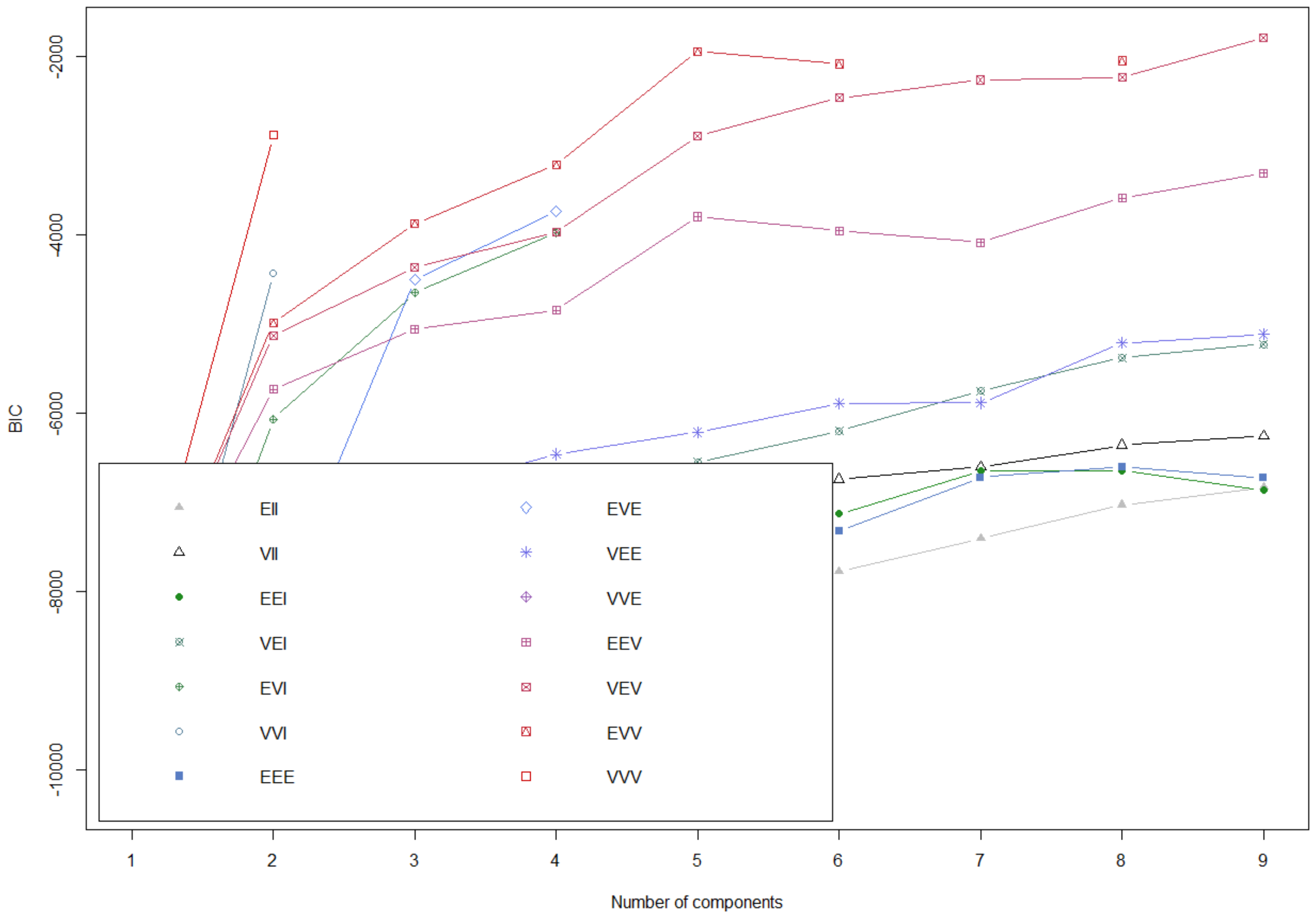
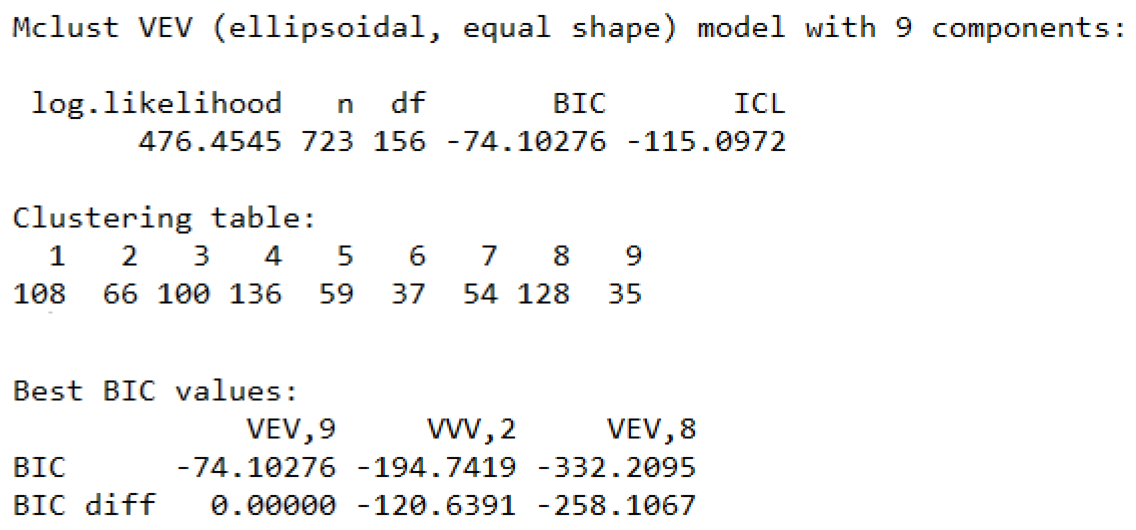


Figure 5-14 BIC values and plot for models fitted to Kaggle data

A summary of the selected model is obtained as:



```
Mclust VEV (ellipsoidal, equal shape) model with 9 components:

 log.likelihood   n  df       BIC       ICL
       476.4545 723 156 -74.10276 -115.0972

Clustering table:
  1   2   3   4   5   6   7   8   9 
108  66 100 136  59  37  54 128  35 

Best BIC values:
             VEV,9      VVV,2      VEV,8
BIC      -74.10276 -194.7419 -332.2095
BIC diff   0.00000 -120.6391 -258.1067
```

Each component is the mixture is what we call a cluster. Mclust uses an identifier for each possible parametrization of the covariance matrix that has three letters: E for “equal”, V for “variable” and I for “coordinate axes. The summary is showing the top models are VEV and VVV. Estimation model complexity by the BIC determined VEV is the best model, visible Figure 5-15.

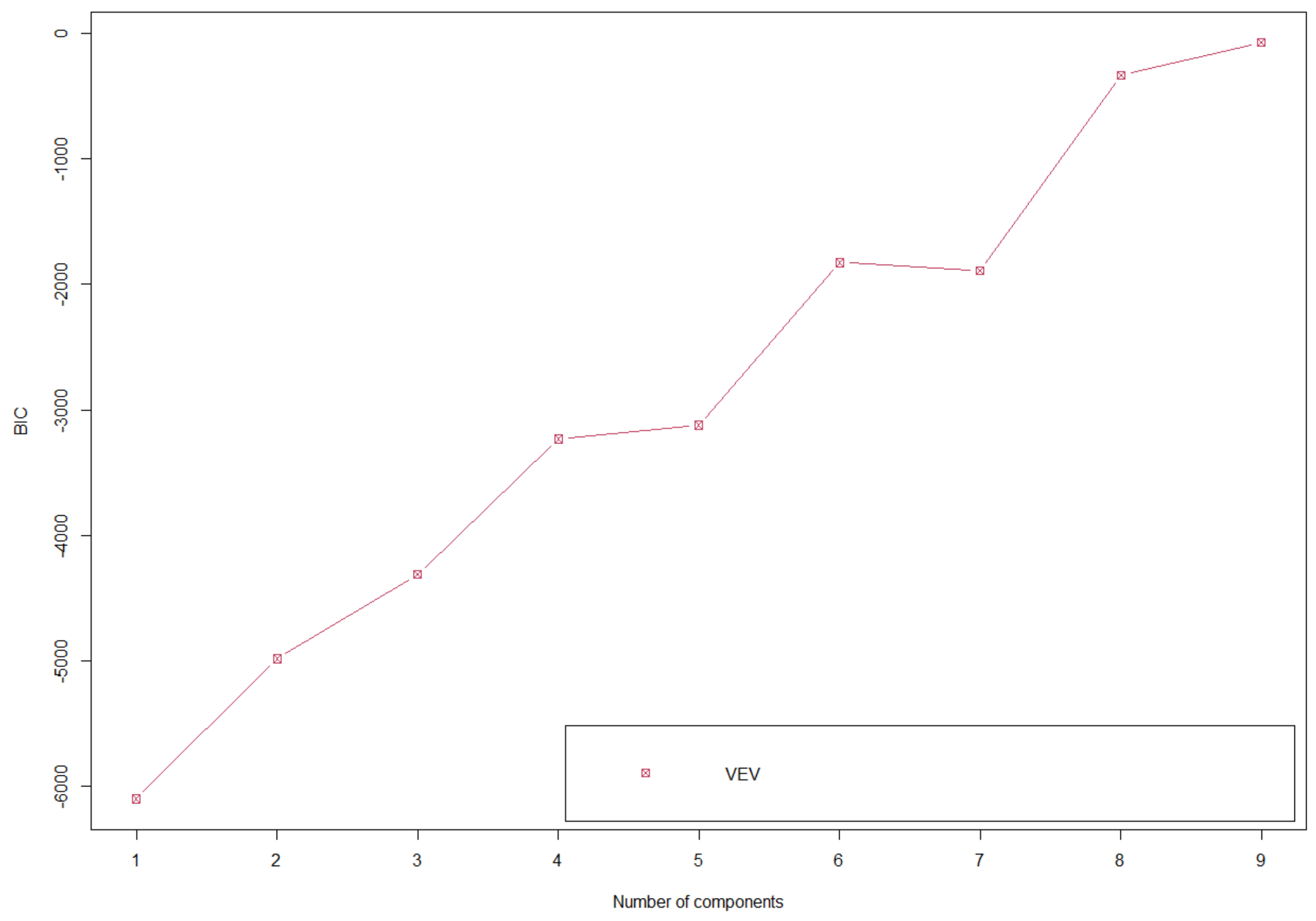


Figure 5-15 Extracted VEV from BIC

BIC fit models with all K of interest and choose the one with largest BIC. Gaussian Mixture Model (GMM) classifier may not be properly separated as the dataset too small. However, the outcome of clustering method validate true number of Kaggle dataset. Cluster may be separated but surrounded, see Figure 5-16.

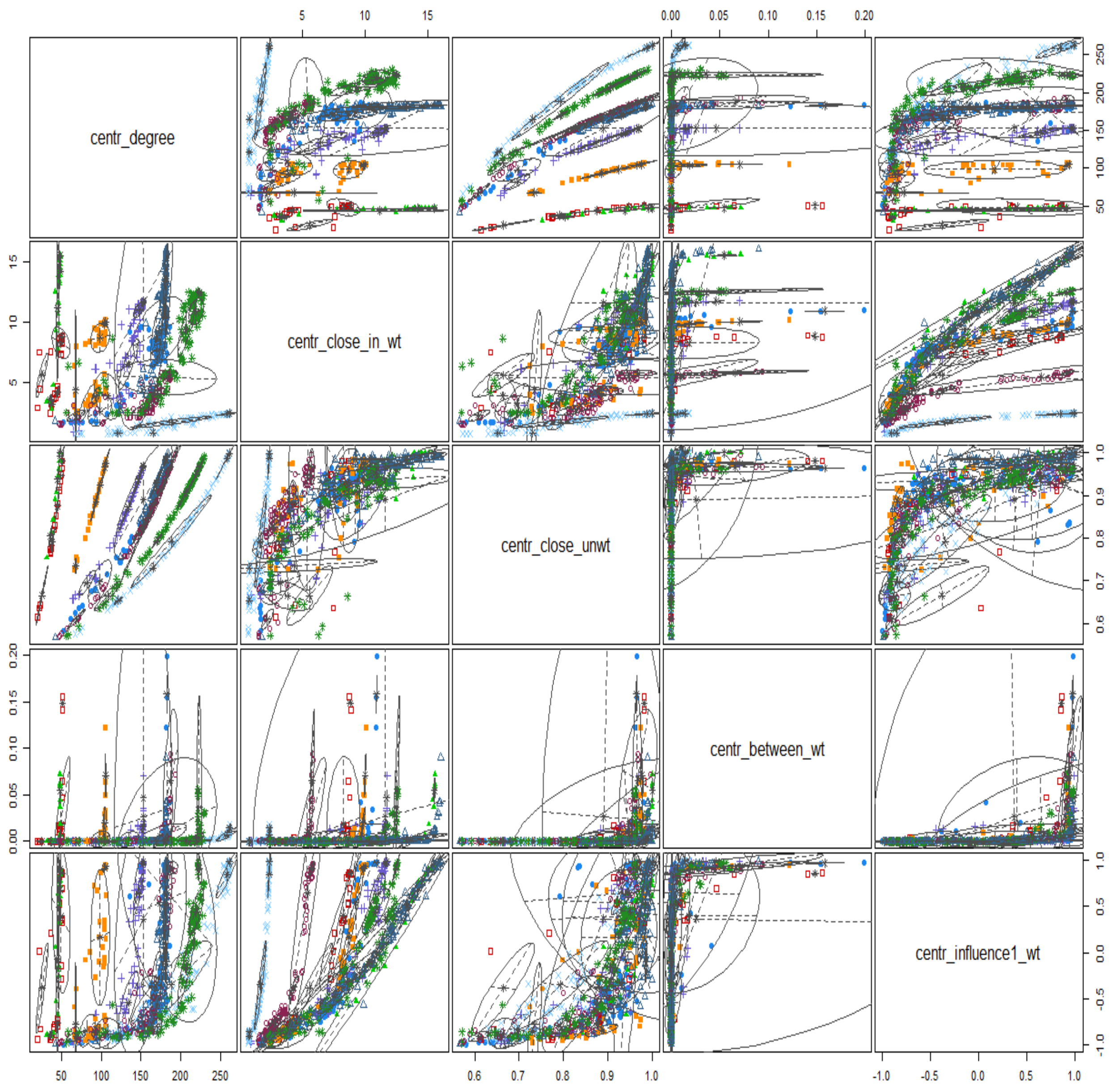


Figure 5-16 Classification of Kaggle Dataset

The error in detection was measured as the root mean squared difference between actual and detected number of clusters in the data. Mismatch between the two can be result from small number of dataset, see Figure 5-17.

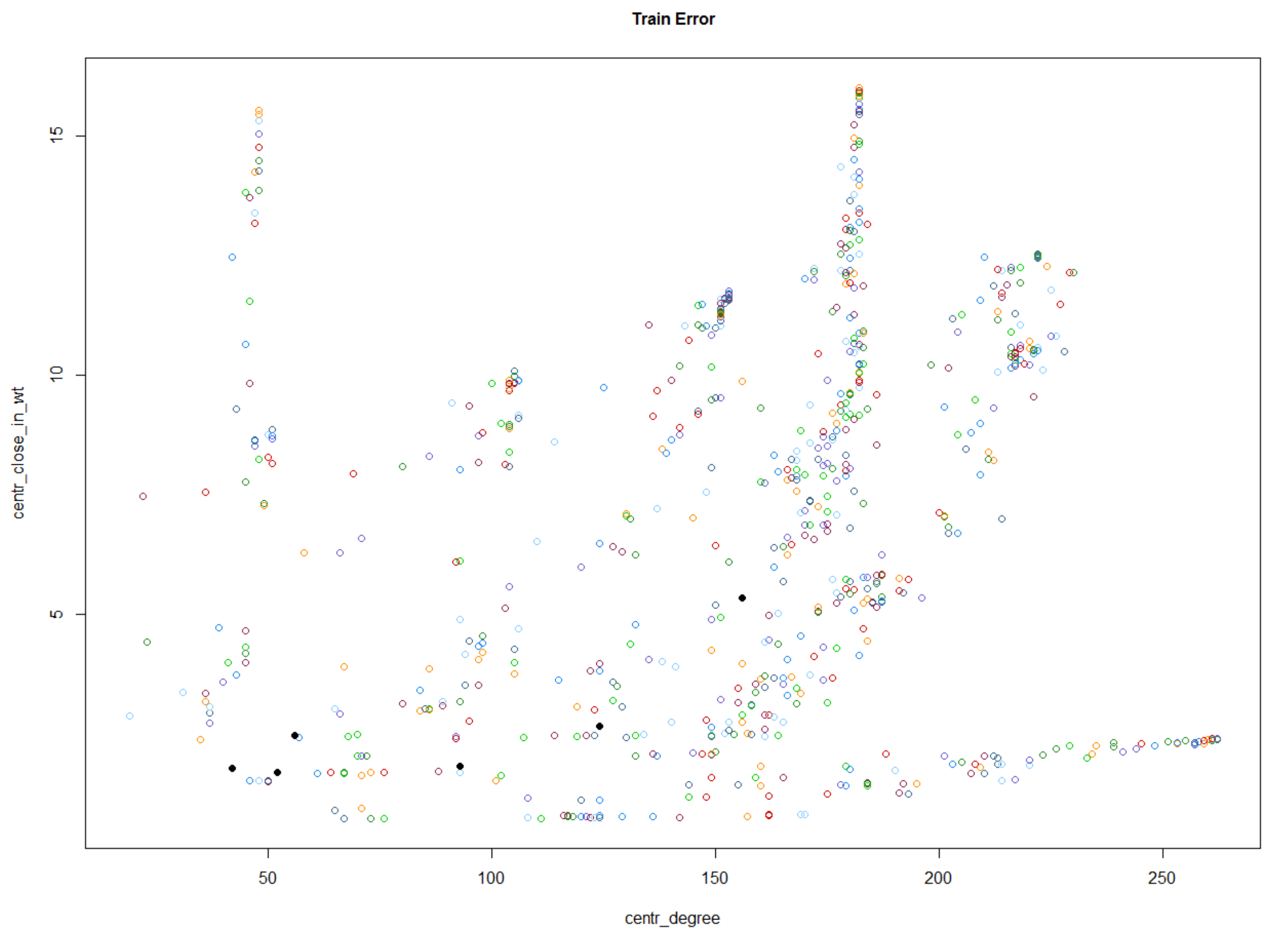


Figure 5-17Error Distribution

Estimation the optimal number of clusters depicted in Figure 5-18. This estimation determined the number of cluster in Kaggle dataset based on gap statistic model. According to this observation, it is possible to define *N=9* as the optimal number of cluster in the data based on segmentation results.

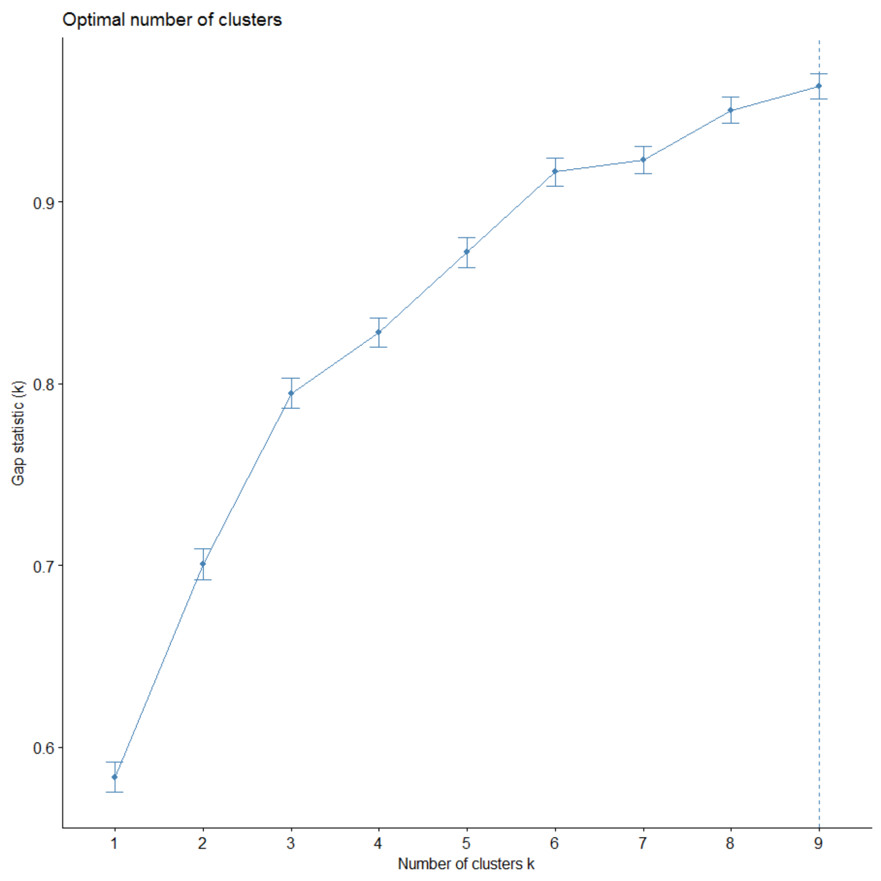


Figure 5-18 Optimal number of Kaggle Clusters

5.3 Discussion

A classification-based approach is presented by using common-factor analytic mixture models. Segmentation analysis attempts to identify significant opcode features of malware types in different families and also to predict group. How to select informative features from more than two classes is explored and examined. SAS Enterprise Miner provides a very powerful visualization of opcode relations. The results showed that the Link Analysis technique for the variable constructor is an efficient method from a feature engineering technique perspective. The key features within a cluster which present high similarity or correlation extracted by segment analysis. Our experimental results show that the combination of link analysis and data segmentation analysis in features engineering can be used to cluster malware families with a small penalty.

Chapter Six

Conclusion and Future Directions

6 Overview

In this chapter discussing the research objectives and lessons learned are presented. The research results offer an attractive feature selection model a means to reduce feature dimension with an adaptable and scalable data mining techniques. The method shows to give excellent classification performance on Kaggle and APTs malware datasets.

6.1 Research objectives revisited

The aim of this research was to better understand feature selection by using the concept of data mining to uncover unknown patterns of advanced malware which can be used as a classification technique. This work is very much inspired by the opcode mining technique. Link Analysis as a powerful data mining technique is used to discover useful associations and sequences hidden in opcode structures. Segment analysis can utilize link analysis result to discover interesting and similar patterns in one group.

Previous studies relied on n-gram, opcode frequency or sequence patterns whereas we utilized association rule mining to determine opcode hidden patterns in one malware family. To examine the difference between each malware groups, link analysis was conducted. We used opcode weights to calculate the prevalence of opcode observations to test for segmentation clustering. To avoid redundant rules, we used segmentation analysis. These rules can automatically form a different type of malware to the respective family if they feed to the training stage.

The Link Analysis rules satisfy some evaluation criterion of family classification. It can be used for creating new derived variables to use in feature construction techniques. Then, tree-based dimensional reduction techniques as such segmentation analysis are used to find the promising features.

Due to sophisticated techniques, static analysis is performed on malware samples to reveal their intended behaviour, which was hidden (Bilar, 2007). Additionally, this method provides more meaningful features for detection purpose compare with behaviour analysis approach(Cosovan, Benchea and Gavrilut, 2015). Most existing rule inductive learning algorithms have been developed based on single rule evaluation measures. Research results indicate that Link Analysis method has led to a desirable number of families based on segmentation analysis. This model of feature engineering is developed based on the characteristics of malicious files and code instructions. It is strongly believed that this research will lead to further research and new detection techniques for polymorphic and metamorphic malware, prior to execution or transmission.

6.2 Recommendation and Future Work

Future work should entail evaluating network analysis of malware families to better assess attribution and intrinsic relationships that may exist. Some useful next steps to this type of analysis include adding an observation of a significant relationship. The relationships between rules in a set are complicated. There are many conditions need to be considered. Measures investigate the subjective aspect of rule interestingness or significant pattern can be considered for further work. Besides, increase the performance of the clustering is another objective.

7 Reference

Ahmadi, M. *et al.* (2013) 'Malware detection by behavioural sequential patterns', *Computer Fraud and Security*. Elsevier Ltd, 2013(8), pp. 11–19. doi: 10.1016/S1361-3723(13)70072-1.

Ahmadi, M. *et al.* (2015) 'Novel Feature Extraction, Selection and Fusion for Effective Malware Family Classification'. doi: 10.1145/2857705.2857713.

Ahmadi, M. and Giacinto, G. (2015) 'Novel Feature Extraction , Selection and Fusion for Effective Malware Family Classification', pp. 183–194.

Alam, S. *et al.* (2015) 'A framework for metamorphic malware analysis and real-time detection', *Computers & Security*. Elsevier Ltd, 48, pp. 212–233. doi: 10.1016/j.cose.2014.10.011.

Alazab, M. (2015) 'Profiling and classifying the behavior of malicious codes', *Journal of Systems and Software*. Elsevier Inc., 100, pp. 91–102. doi: 10.1016/j.jss.2014.10.031.

Azab, A. *et al.* (2015) 'Mining Malware To Detect Variants'. doi: 10.1109/CTC.2014.11.

Azmoodeh, A. *et al.* (2018) 'Robust Malware Detection for Internet Of (Battlefield) Things Devices Using Deep Eigenspace Learning', 3782(c), pp. 1–9. doi: 10.1109/TSUSC.2018.2809665.

Baldangombo, U., Jambaljav, N. and Horng, S.-J. (2013) 'A Static Malware Detection System Using Data Mining Methods', *International Journal of Artificial Intelligence & Applications*, 4(4), p. 113. Available at: http://arxiv.org/abs/1308.2831.

Ban, T. *et al.* (2015) 'A Study on Association Rule Mining of Darknet Big Data'.

Bekerman, D. *et al.* (2015) 'Unknown malware detection using network traffic classification', *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 134–142. doi: 10.1109/CNS.2015.7346821.

Bernardi, M. L. *et al.* (2017) 'A Fuzzy-based Process Mining Approach for Dynamic Malware Detection'.

Bhatt, P., Yano, E. T. and Gustavsson, P. (2014) 'Towards a framework to detect multi-stage advanced persistent threats attacks', *Proceedings - IEEE 8th International Symposium on Service Oriented System Engineering, SOSE 2014*, pp. 390–395. doi: 10.1109/SOSE.2014.53.

Bilar, D. (2007) 'Opcodes as predictor for malware', 1(2), pp. 156–168.

Bist, A. S. and Campus, Q. G. (2014) 'Fuzzy Logic for Computer Virus Detection', 3(2), pp. 771–773.

Bot, O. (2017) 'HACGA : An artifacts-based clustering approach for malware classification', pp. 5–12.

Canfora, G. *et al.* (2015) 'Effectiveness of opcode ngrams for detection of multi family android malware', *Proceedings - 10th International Conference on Availability, Reliability and Security, ARES*

2015, pp. 333–340. doi: 10.1109/ARES.2015.57.

Chandran, S., Hrudya, P. and Poornachandran, P. (2015) ‘An efficient classification model for detecting advanced persistent threat’, *2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015*, pp. 2001–2009. doi: 10.1109/ICACCI.2015.7275911.

Cheng, J. Y. C., Tsai, T. S. and Yang, C. S. (2013) ‘An information retrieval approach for malware classification based on Windows API calls’, *Proceedings - International Conference on Machine Learning and Cybernetics*, 4, pp. 1678–1683. doi: 10.1109/ICMLC.2013.6890868.

Comar, P. M. *et al.* (2013) ‘Combining Supervised and Unsupervised Learning for Zero-Day Malware Detection’, pp. 2022–2030.

Cosovan, D., Benchea, R. and Gavrilut, D. (2015) ‘A practical guide for detecting the java script-based malware using hidden markov models and linear classifiers’, *Proceedings - 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2014*, pp. 236–243. doi: 10.1109/SYNASC.2014.39.

Damshenas, M., Dehghantanha, A. and Mahmoud, R. (2013) ‘International Journal of Cyber-Security and Digital Forensics (IJCSDF) 2 (4): 10-29 The Society of Digital Information and Wireless Communications , 2013 (ISSN : 2305-0012) A SURVEY ON MALWARE PROPAGATION , ANALYSIS , AND DETECTION International Jou’, 2(4), pp. 10–29.

Das, S. *et al.* (2016) ‘Semantics-based online malware detection: Towards efficient real-time protection against malware’, *IEEE Transactions on Information Forensics and Security*, 11(2), pp. 289–302. doi: 10.1109/TIFS.2015.2491300.

Deepa, K., Radhamani, G. and Vinod, P. (2015) ‘Investigation of feature selection methods for android malware analysis’, *Procedia Computer Science*, 46(Icict 2014), pp. 841–848. doi: 10.1016/j.procs.2015.02.153.

Deka, D., Sarma, N. and Panicker, N. J. (2017) ‘Malware detection vectors and analysis techniques: A brief survey’, *2016 International Conference on Accessibility to Digital World, ICADW 2016 - Proceedings*, pp. 81–85. doi: 10.1109/ICADW.2016.7942517.

Ding, Y. *et al.* (2014) ‘Control flow-based opcode behavior analysis for Malware detection’, *Computers and Security*. Elsevier Ltd, 44(2007), pp. 65–74. doi: 10.1016/j.cose.2014.04.003.

Ekhtoom, D. *et al.* (2016) ‘A Compression-Based Technique to Classify Metamorphic Malware’.

Epishkina, A. and Zapechnikov, S. (2016) ‘A Syllabus on Data Mining and Machine Learning with Applications to Cybersecurity’, pp. 194–199.

Fan, M. *et al.* (2016) ‘Frequent Subgraph Based Familial Classification of Android Malware’, *Proceedings - International Symposium on Software Reliability Engineering, ISSRE*, pp. 24–35. doi: 10.1109/ISSRE.2016.14.

Firdausi, I. *et al.* (2010) ‘Analysis of Machine learning Techniques Used in Behavior-Based Malware Detection’, *Advances in Computing, Control and Telecommunication Technologies (ACT), 2010 Second*

International Conference on, pp. 10–12. doi: 10.1109/ACT.2010.33.

Fraley, J. B. (2016) ‘Polymorphic Malware Detection Using Topological Feature Extraction with Data Mining’.

Gandotra, E., Bansal, D. and Sofat, S. (2016) ‘Detecting Zero Day Malware’, *Ised*.

Gharacheh, M. *et al.* (2016) ‘Detection of Metamorphic Malware based on HMM: A Hierarchical Approach’, *International Journal of Intelligent Systems and Applications*, 8(4), pp. 18–25. doi: 10.5815/ijisa.2016.04.02.

Hassani, H. and Zarei, J. (2015) ‘Interval Type-2 fuzzy logic controller design for the speed control of DC motors’, *Systems Science & Control Engineering*, 3(1), pp. 266–273. doi: 10.1080/21642583.2015.1013644.

Homayoun, S. *et al.* (2017) ‘Know Abnormal , Find Evil : Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence’, 6750(c), pp. 1–11. doi: 10.1109/TETC.2017.2756908.

Hu, W. and Tan, Y. (2017) ‘On the robustness of machine learning based malware detection algorithms’, *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1435–1441. doi: 10.1109/IJCNN.2017.7966021.

Hu, X. *et al.* (2013) ‘MutantX-S: Scalable Malware Clustering Based on Static Features’, *USENIX Annual Technical Conference*, pp. 187–198. doi: 10.1.1.389.5696.

Huang, H. De *et al.* (2012) ‘TWMAN+: A type-2 fuzzy ontology model for malware behavior analysis’, *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pp. 2821–2826. doi: 10.1109/ICSMC.2012.6378176.

Huang, H. De *et al.* (2013) ‘An IT2FLS-based malware analysis mechanism: Malware analysis network in Taiwan (MIT)’, *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pp. 4652–4657. doi: 10.1109/SMC.2013.792.

Idrees, F. *et al.* (2017) ‘PIndroid: A novel Android malware detection system using ensemble learning methods’, *Computers & Security*. Elsevier Ltd, 68, pp. 36–46. doi: 10.1016/j.cose.2017.03.011.

Islam, R. *et al.* (2013) ‘Classification of malware based on integrated static and dynamic features’, *Journal of Network and Computer Applications*. Elsevier, 36(2), pp. 646–656. doi: 10.1016/j.jnca.2012.10.004.

Islamic, I. and Minna, T. (no date) ‘Android Malware Classification Using Static Code Analysis and Apriori Algorithm Improved with Particle Swarm Optimization’, pp. 123–128.

Kaur, R. and Singh, M. (2014) ‘A survey on zero-day polymorphic worm detection techniques’, *IEEE Communications Surveys and Tutorials*, 16(3), pp. 1520–1549. doi: 10.1109/SURV.2014.022714.00160.

Khazaee, S. and Faez, K. (2014) ‘A Novel Classification Method Using Hybridization of Fuzzy Clustering and Neural Networks for Intrusion Detection’, *International Journal of Modern Education and Computer Science(IJMECS)*, 6(11), p. 11. doi: 10.5815/ijmecs.2014.11.02.

Khazaee, S. and Sharifi Rad, M. (2013) ‘Using fuzzy c-means algorithm for improving intrusion detection performance’.

Le, D. C., Zincir-Heywood, A. N. and Heywood, M. I. (2017) ‘Data analytics on network traffic flows for botnet behaviour detection’, *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*. doi: 10.1109/SSCI.2016.7850078.

Liangboonprakong, C. and Sornil, O. (2013) ‘Classification of malware families based on N-grams sequential pattern features’, *Proceedings of the 2013 IEEE 8th Conference on Industrial Electronics and Applications, ICIEA 2013*, pp. 777–782. doi: 10.1109/ICIEA.2013.6566472.

Lin, Y.-S., Jiang, J.-Y. and Lee, S.-J. (2014) ‘A Similarity Measure for Text Classification and Clustering’, *IEEE Transactions on Knowledge and Data Engineering*, 26(7), pp. 1575–1590. doi: 10.1109/TKDE.2013.19.

Ling, Y., Putra, U. and Ling, Y. (2017) ‘Short Review on Metamorphic Malware Detection in Hidden Markov Models International Journal of Advanced Research in Short Review on Metamorphic Malware Detection in Hidden Markov Models’, (June).

Liu, D. *et al.* (2014) ‘Network Traffic Anomaly Detection Using Adaptive Density-Based Fuzzy Clustering’, *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications*, 16, pp. 823–830. doi: 10.1109/TrustCom.2014.109.

Liu, J. *et al.* (2013) ‘FENOC: An ensemble one-class learning framework for malware detection’, *Proceedings - 9th International Conference on Computational Intelligence and Security, CIS 2013*, pp. 523–527. doi: 10.1109/CIS.2013.116.

Liu, L. *et al.* (2017) ‘Automatic malware classification and new malware detection using machine learning’, *Frontiers of Information Technology & Electronic Engineering*, 18(9), pp. 1336–1347. doi: 10.1631/FITEE.1601325.

Liu, Y. *et al.* (2014) *Link Analysis Using SAS ® Enterprise Miner^{TM}*. Available at: https://support.sas.com/rnd/app/data-mining/enterprise-miner/papers/2014/linkAnalysis2014.pdf.

Louk, M. *et al.* (2014) ‘An Effective Framework of Behavior Detection- Advanced Static Analysis for Malware Detection’, *International Symposium on Communications and Information Technologies (ISCIT)*, pp. 361–365. doi: 10.1109/ISCIT.2014.7011932.

Luo, L., Ming, J. and Wu, D. (2017) ‘Semantics-Based Obfuscation-Resilient Binary Code Similarity Comparison with Applications to Software and Algorithm Plagiarism Detection’, 43(12), pp. 1157–1177.

Mastjik, F. and Varol, C. (2015) ‘Comparison of Pattern Matching Techniques on Identification of Same Family Malware’, 4(May), pp. 224–228.

Mehra, M. and Pandey, D. (2016) ‘Event triggered malware: A new challenge to sandboxing’, *12th IEEE International Conference Electronics, Energy, Environment, Communication, Computer, Control: (E3-C3), INDICON 2015*, pp. 1–6. doi: 10.1109/INDICON.2015.7443327.

Mendel, J. M., John, R. I. and Liu, F. (2006) ‘Interval Type-2 Fuzzy Logic Systems Made Simple’, *Fuzzy*

Systems, IEEE Transactions on, 14(6), pp. 808–821. doi: 10.1109/TFUZZ.2006.879986.

Milosevic, N., Dehghantanha, A. and Choo, K. R. (2017) ‘Machine learning aided Android malware classification R’. Elsevier Ltd, 61, pp. 266–274. doi: 10.1016/j.compeleceng.2017.02.013.

Mohaisen, A., Alrawi, O. and Mohaisen, M. (2015) ‘AMAL: High-fidelity, behavior-based automated malware analysis and classification’, *Computers and Security*. Elsevier Ltd, 52, pp. 251–266. doi: 10.1016/j.cose.2015.04.001.

Mohammadi, M. and Hamzeh, A. (2017) ‘Proposing an efficient approach for malware clustering’.

Mohammed, W., Mohammed, S. and Saraee, M. M. (2016) ‘Sematic Web Mining Using Fuzzy C-means Algorithm SCIENCEDOMAIN international Sematic Web Mining Using Fuzzy C-means Algorithm’, (January). doi: 10.9734/BJMCS/2016/25471.

Mooney, C. H. and Roddick, J. F. (2013) ‘Sequential pattern mining -- approaches and algorithms’, *ACM Computing Surveys*, 45(2), pp. 1–39. doi: 10.1145/2431211.2431218.

Narouei, M. *et al.* (2015) ‘DLLMiner : structural mining for malware detection’, (April), pp. 3311–3322. doi: 10.1002/sec.

Narra, U. *et al.* (2016) ‘Clustering versus SVM for malware detection’, *Journal of Computer Virology and Hacking Techniques*. Springer Paris, 12(4), pp. 213–224. doi: 10.1007/s11416-015-0253-z.

Nauman, M., Azam, N. and Yao, J. (2016) ‘A three-way decision making approach to malware analysis using probabilistic rough sets’, *Information Sciences*. Elsevier Inc., 374, pp. 193–209. doi: 10.1016/j.ins.2016.09.037.

Nissim, N. *et al.* (2014) ‘Novel active learning methods for enhanced PC malware detection in windows OS’, *Expert Systems with Applications*. Elsevier Ltd, 41(13), pp. 5843–5857. doi: 10.1016/j.eswa.2014.02.053.

Nissim, N. *et al.* (2015) ‘Detection of malicious PDF files and directions for enhancements: A state-of-the art survey’, *Computers and Security*. Elsevier Ltd, 48, pp. 246–266. doi: 10.1016/j.cose.2014.10.014.

O’Kane, P., Sezer, S. and McLaughlin, K. (2014) ‘N-gram density based malware detection’, *2014 World Symposium on Computer Applications and Research, WSCAR 2014*. doi: 10.1109/WSCAR.2014.6916806.

Pai, S. *et al.* (2016) ‘Clustering for malware classification Clustering for malware classification’, *Journal of Computer Virology and Hacking Techniques*. Springer Paris, (April). doi: 10.1007/s11416-016-0265-3.

Pandeeswari, N. and Kumar, G. (2016) ‘Anomaly Detection System in Cloud Environment Using Fuzzy Clustering Based ANN’, *Mobile Networks and Applications*. Mobile Networks and Applications, 21(3), pp. 494–505. doi: 10.1007/s11036-015-0644-x.

Poulsen, R. (2013) ‘SAS Global Forum 2013 Statistics and Data Analysis Multivariate Statistical Analysis in SAS : Segmentation and Classification of Behavioral Data ABSTRACT SAS Global Forum 2013’, pp. 1–14.

Rathnayaka, C. and Jamdagni, A. (2017) ‘An Efficient Approach for Advanced Malware Analysis Using Memory Forensic Technique’, *2017 IEEE Trustcom/BigDataSE/ICESS*, pp. 1145–1150. doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.365.

Rezaei, S. and Afraz, A. (2016) ‘Malware Detection using Opcodes Statistical Features’, pp. 151–155.

Runwal, N., Low, R. M. and Stamp, M. (2012) ‘Opcode graph similarity and metamorphic detection’, *Journal in Computer Virology*, 8(1–2), pp. 37–52. doi: 10.1007/s11416-012-0160-5.

Santos, I. *et al.* (2013) ‘Opcode sequences as representation of executables for data-mining-based unknown malware detection’, *Information Sciences*, 231, pp. 64–82. doi: 10.1016/j.ins.2011.08.020.

Santos, I. *et al.* (no date) ‘Idea : Opcode-sequence-based Malware Detection’.

Sarvani, A. (no date) ‘Clustering The Polymorphic Malware Traces’.

Schultz, M. G. *et al.* (2001) ‘Data mining methods for detection of new malicious executables’, *Proceedings. 2001 IEEE Symposium on Security and Privacy, 2001. S&P 2001.*, pp. 38–49. doi: 10.1109/SECPRI.2001.924286.

Scott, J., Fellow, S. and Technology, C. I. (2017) ‘Signature Based Malware Detection is Dead’, (February).

Security, I. and Report, T. (2017) ‘No Title’, (April).

Shabtai, A. *et al.* (2012) ‘Detecting unknown malicious code by applying classification techniques on OpCode patterns’, *Security Informatics*, 1(1), p. 1. doi: 10.1186/2190-8532-1-1.

Shalaginov, A. (2017) ‘Dynamic feature-based expansion of fuzzy sets in Neuro-Fuzzy for proactive malware detection’. doi: 10.23919/ICIF.2017.8009812.

Shalaginov, A., Grini, L. S. and Franke, K. (2016) ‘Understanding Neuro-Fuzzy on a class of multinomial malware detection problems’, *Proceedings of the International Joint Conference on Neural Networks*, 2016–Octob, pp. 684–691. doi: 10.1109/IJCNN.2016.7727266.

Sharma, A. and Sahay, S. K. (2016) ‘An effective approach for classification of advanced malware with high accuracy’, *International Journal of Security and its Applications*, 10(4), pp. 249–266. doi: 10.14257/ijsia.2016.10.4.24.

Shenwen, L., Yingbo, L. and Xiongjie, D. (2015) ‘Study and research of APT detection technology based on big data processing architecture’, *2015 IEEE 5th International Conference on Electronics Information and Emergency Communication*, (2012), pp. 313–316. doi: 10.1109/ICEIEC.2015.7284547.

Shijo, P. V. and Salim, A. (2015) ‘Integrated static and dynamic analysis for malware detection’, *Procedia Computer Science*. Elsevier Masson SAS, 46(Icict 2014), pp. 804–811. doi: 10.1016/j.procs.2015.02.149.

Siddequi, M., Wang, M. C. and Lee, J. (2008) ‘Detecting Internet Worms Using Data Mining Techniques’, *Journal of Systemics, Cybernetics & Informatics*, 6, pp. 48–53. doi:

10.1145/1593105.1593239.

Souri, A. and Hosseini, R. (2018) ‘A state ‑ of ‑ the ‑ art survey of malware detection approaches using data mining techniques’, *Human-centric Computing and Information Sciences*. Springer Berlin Heidelberg. doi: 10.1186/s13673-018-0125-x.

Suarez-tangil, G. *et al.* (2014) ‘Expert Systems with Applications D ENDROID : A text mining approach to analyzing and classifying code structures in Android malware families’, *Expert Systems With Applications*. Elsevier Ltd, 41(4), pp. 1104–1117. doi: 10.1016/j.eswa.2013.07.106.

Thompson, W. (2008) ‘SAS Global Forum 2008 Data Mining and Predictive Modeling and Predicting Risk in the Telecom Industry Glendon Cross , AT & T Corporation SAS Global Forum 2008 Data Mining and Predictive Modeling’, pp. 1–14.

Thomson, G. (2011) ‘APTs: A poorly understood challenge’, *Network Security*. Elsevier Ltd, 2011(11), pp. 9–11. doi: 10.1016/S1353-4858(11)70118-0.

Thunga, S. P. and Neelisetti, R. K. (2015) ‘Identifying metamorphic virus using n-grams and Hidden Markov Model’, *2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2015*, pp. 2016–2022. doi: 10.1109/ICACCI.2015.7275913.

Tong, F. and Yan, Z. (2017) ‘A hybrid approach of mobile malware detection in Android’, *Journal of Parallel and Distributed Computing*. Elsevier Inc., 103, pp. 22–31. doi: 10.1016/j.jpdc.2016.10.012.

Ucci, D., Aniello, L. and Baldoni, R. (2017) ‘Survey on the Usage of Machine Learning Techniques for Malware Analysis’, 1(1). doi: 10.1109/INTECH.2016.7845073.

Wang, C. *et al.* (2016) ‘A malware variants detection methodology with an opcode based feature method and a fast density based clustering algorithm’, *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, ICNC-FSKD 2016*, pp. 481–487. doi: 10.1109/FSKD.2016.7603221.

Wang, H. T. *et al.* (2013) ‘Clustering of similar malware behavior via structural host-sequence comparison’, *Proceedings - International Computer Software and Applications Conference*, pp. 349–358. doi: 10.1109/COMPSAC.2013.60.

Wang, M. C. (2014) ‘Data mining methods for malware detection using instruction sequences DATA MINING METHODS FOR MALWARE DETECTION USING’, (January 2008).

Wang, P. and Wang, Y. S. (2015) ‘Malware behavioural detection and vaccine development by using a support vector model classifier’, *Journal of Computer and System Sciences*. Elsevier Inc., 81(6), pp. 1012–1026. doi: 10.1016/j.jcss.2014.12.014.

Wei, C., Sprague, A. and Warner, G. (2009) ‘Clustering malware-generated spam emails with a novel fuzzy string matching algorithm’, *Sac*, (205), pp. 889–890. doi: http://doi.acm.org/10.1145/1529282.1529473.

Ye, Y. (2017) ‘A Survey on Malware Detection Using Data Mining Techniques’, 50(3).

Yewale, A. and Singh, M. (2017) ‘Malware detection based on opcode frequency’, *Proceedings of 2016 International Conference on Advanced Communication Control and Computing Technologies, ICACCCT 2016*, (978), pp. 646–649. doi: 10.1109/ICACCCT.2016.7831719.

Zhang, Y. *et al.* (2011) ‘Fuzzy neural network for malware detect’, *Proceedings - 2010 International Conference on Intelligent System Design and Engineering Application, ISDEA 2010*, 1, pp. 780–783. doi: 10.1109/ISDEA.2010.314.

Zhao, G. *et al.* (2015) ‘Detecting APT malware infections based on malicious DNS and traffic analysis’, *IEEE Access*, 3(5), pp. 1132–1142. doi: 10.1109/ACCESS.2015.2458581.

Zolotukhin, M. and Hämäläinen, T. (2014) ‘Detection of zero-day malware based on the analysis of opcode sequences’, *2014 IEEE 11th Consumer Communications and Networking Conference, CCNC 2014*, pp. 386–391. doi: 10.1109/CCNC.2014.6866599.

Appendices

Appendix A: Segmentation Profile Node Report-Kaggle

```
Frequencies: _group

                                                Percent of
Segment      Segment        Frequency          Total
Variable     Value            Count          Frequency

 _group      Ramnit               52          18.3746
 _group      Tracur               47          16.6078
 _group      Gatak                38          13.4276
 _group      Simda                38          13.4276
 _group      Vundo                36          12.7208
 _group      Lolipop              31          10.9541
 _group      Obfuscator           21           7.4205
 _group      KelihosV_1           10           3.5336
 _group      KelihosV_3           10           3.5336



Variable: _group Segment: Ramnit Count: 52
Decision Tree Importance Profiles

Variable                    Worth     Rank

_centr_influence1_unwt    0.27233      1
_centr_close_in_wt        0.24387      2
_centr_degree             0.16460      3
_centr_close_unwt         0.09183      4
_centr_between_wt         0.04828      5



Variable: _group Segment: Tracur Count: 47
Decision Tree Importance Profiles

Variable                    Worth     Rank

_centr_degree             0.16978      1
_centr_influence1_unwt    0.08682      2
_centr_close_in_wt        0.07389      3
_centr_close_unwt         0.03446      4
_centr_between_wt         0.01051      5



Variable: _group Segment: Gatak Count: 38
Decision Tree Importance Profiles

Variable                     Worth     Rank
```

```
_centr_close_in_wt        0.083503      1
_centr_influence1_unwt    0.080400      2
_centr_degree             0.057014      3
_centr_close_unwt         0.033149      4
_centr_between_wt         0.004634      5



Variable: _group Segment: Simda Count: 38
Decision Tree Importance Profiles

Variable                   Worth     Rank

_centr_influence1_unwt    0.17166      1
_centr_close_in_wt        0.08586      2
_centr_degree             0.03084      3
_centr_close_unwt         0.03067      4
_centr_between_wt         0.00904      5



Variable: _group Segment: Vundo Count: 36
Decision Tree Importance Profiles

Variable                   Worth     Rank

_centr_influence1_unwt    0.11885      1
_centr_degree             0.07602      2
_centr_close_in_wt        0.06236      3
_centr_close_unwt         0.05760      4
_centr_between_wt         0.00947      5



Variable: _group Segment: Lolipop Count: 31
Decision Tree Importance Profiles

Variable                   Worth     Rank

_centr_degree             0.10151      1
_centr_influence1_unwt    0.07562      2
_centr_close_in_wt        0.07305      3
_centr_close_unwt         0.02801      4
_centr_between_wt         0.02173      5



Variable: _group Segment: Obfuscator Count: 21
Decision Tree Importance Profiles

Variable                    Worth     Rank

_centr_degree             0.083687      1
_centr_close_in_wt        0.027060      2
_centr_close_unwt         0.023447      3
_centr_influence1_unwt    0.017557      4
```

```
_centr_between_wt         0.005840      5



Variable: _group Segment: KelihosV_1 Count: 10
Decision Tree Importance Profiles

Variable                    Worth     Rank

_centr_degree             0.047982      1
_centr_close_in_wt        0.010275      2
_centr_influence1_unwt    0.007999      3
_centr_close_unwt         0.007001      4
_centr_between_wt         0.002654      5



Variable: _group Segment: KelihosV_3 Count: 10
Decision Tree Importance Profiles

Variable                    Worth     Rank

_centr_degree             0.048740      1
_centr_influence1_unwt    0.027410      2
_centr_close_in_wt        0.025877      3
_centr_close_unwt         0.018145      4
_centr_between_wt         0.006611      5
```

Appendix B: Segmentation Profile Node Report-APTs

```
Frequencies: _group

                                            Percent of
Segment      Segment     Frequency          Total
Variable      Value        Count          Frequency

 _group       APT28            44          32.5926
 _group       APT33            26          19.2593
 _group       APT29            25          18.5185
 _group       APT1             22          16.2963
 _group       APT30            18          13.3333



Variable: _group Segment: APT28 Count: 44
Decision Tree Importance Profiles

       Variable            Worth     Rank

_centr_close_in_unwt      0.43940      1
_centr_close_in_wt        0.43940      2
_centr_close_out_unwt     0.43940      3
_centr_close_out_wt       0.43940      4
_centr_close_unwt         0.43940      5
_centr_close_wt           0.43940      6
_centr_influence2_wt      0.42643      7
_centr_degree_out         0.27233      8
_centr_influence1_unwt    0.24858      9
_centr_influence2_unwt    0.23344     10



Variable: _group Segment: APT33 Count: 26
Decision Tree Importance Profiles

Variable                   Worth     Rank

_centr_influence2_wt      0.31100      1
_centr_influence1_unwt    0.29830      2
_centr_influence2_unwt    0.29753      3
_centr_degree             0.28511      4
_centr_degree_out         0.26903      5
_centr_degree_in          0.26257      6
_centr_close_wt           0.23503      7
_centr_cluster            0.23396      8
_centr_close_out_wt       0.23119      9
_centr_close_in_wt        0.22109     10



Variable: _group Segment: APT29 Count: 25
Decision Tree Importance Profiles

Variable                   Worth     Rank
```

```
_centr_influence2_wt      0.27452      1
_centr_degree             0.21035      2
_centr_degree_out         0.20070      3
_centr_close_in_wt        0.19637      4
_centr_close_out_wt       0.19637      5
_centr_close_wt           0.19637      6
_centr_cluster            0.17903      7
_centr_influence1_unwt    0.17490      8
_centr_degree_in          0.16511      9
_centr_close_out_unwt     0.16447     10


Variable: _group Segment: APT1 Count: 22
Decision Tree Importance Profiles

Variable                   Worth     Rank

_centr_influence2_wt      0.24571      1
_centr_influence2_unwt    0.21405      2
_centr_influence1_unwt    0.19313      3
_centr_degree_out         0.16819      4
_centr_close_out_wt       0.15450      5
_centr_close_out_unwt     0.15202      6
_centr_degree             0.14849      7
_centr_degree_in          0.14841      8
_centr_close_in_wt        0.14407      9
_centr_close_wt           0.14407     10


Variable: _group Segment: APT30 Count: 18
Decision Tree Importance Profiles

Variable                   Worth     Rank

_centr_influence2_wt      0.23111      1
_centr_influence2_unwt    0.21708      2
_centr_close_in_wt        0.19930      3
_centr_close_out_wt       0.19619      4
_centr_close_wt           0.19619      5
_centr_degree_in          0.18111      6
_centr_between_unwt       0.16531      7
_centr_influence1_unwt    0.16208      8
_centr_degree_out         0.15915      9
_centr_degree             0.15726     10
```