Classifying Windows Ransomware Based On Runtime Behavior Using Machine Learning Algorithms

by

Lovina Moses Dmello

(Under the Direction of Kyu Hyung, Lee)

Abstract

Ransomware is defined as a type of malware program that infects, locks or takes control of the users system and demands ransom from the user to undo the damage. Ransomware detection is an important factor in security of computer systems. However, Zero-day attacks and polymorphic viruses are not easily detected by signature-based methods. As a result, need for machine learning based detection arises. The purpose of this work is to determine result of feature selection on classification methods when used on top of cuckoo sandbox. Classification algorithms like k-Nearest-Neighnours, Naive Bayes, Support Vector Machines and Random Forest were evaluated. The dataset for this study consisted over 1584 ransomware samples of 11 different ransomware families. Cuckoo sandbox is used to run these samples and see their real time behavior. This work demonstrated the improvement in accuracy obtained using mutual information criteria for feature selection.

Index words: Security, Computer Security, Ransomware, Ransomware Classification, Windows Ransomware

Classifying Windows Ransomware Based On Runtime Behavior Using Machine Learning Algorithms

by

Lovina Moses Dmello

B.E. University of Mumbai, 2015

A Thesis Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Master of Science

Athens, Georgia

2018

Classifying Windows Ransomware Based On Runtime Behavior Using Machine Learning Algorithms

by

Lovina Moses Dmello

Dedication

To Elizabeth and Moses Dmello, my loving parents. Blaise and Rebecca, my supportive siblings.

Acknowledgments

I would like to express my deepest gratitude to my major professor Dr. Kyu Lee for guiding me throughout the course of my research. Thank you so much for motivating me and pushing me to achieve the best results. I would like to express my sincere thanks to Dr. Bill Hollingworth, under whom I got to learn a lot. I am really grateful to Dr. Maria Hybinette, not only for being on my committee but also for being such a wonderful professor.
I would especially like to thank my parents for being there for me in difficult times. I can never forget my siblings Blaise and Rebecca who always cheer me up no matter what. I would also specially thank Vinamra Jain for all his support and encouragement from time to time. All the UGA professors from whom I have learned a lot in these two years. Also special thanks to all my friends back at home and anyone I might have missed. Thank you for always being there for me. It means a lot.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

While the diversity of malware is increasing, anti-virus scanners cannot fulfill the needs of protection, resulting in millions of hosts being attacked. According to Kaspersky Labs (2016), 6 563 145 different hosts were attacked, and 4 000 000 unique malware objects were detected in 2015. In turn, Juniper Research (2016) predicts the cost of data breaches to increase to $2.1 trillion globally by 2019.

High availability of anti-detection techniques, as well as ability to buy malware on the black market result in the opportunity to become an attacker for anyone, not depending on the skill level. Current studies show that more and more attacks are being issued by script-kiddies or are automated(Aliyev 2010). Also, current static and dynamic methods do not provide efficient detection, especially when dealing with zero-day attacks. For this reason, machine learning-based techniques can be used.

In this project we will try to develop proof of concept for the machine learning based malware classification based on cuckoo sandbox. Cuckoo Sandbox will be utilized for the extraction of ransomware sample features. This features will be used as an input for the machine learning algorithms. The goal is to determine which machine learning algorithm accurately distinguishes the ransomware families with low error rate. The study conducted will allow building an additional detection module to cuckoo sandbox. However, the implementation of this module is beyond the scope of this project.

1.1 What is Ransomware ?

Ransomware is a type of malware that prevents or limits users from accessing their system, either by locking the system's screen or by locking the user's files unless a ransom is paid. Modern ransomware families like crypto-ransomware encrypt certain files on infected systems and forces users to pay the ransom through online payment methods to get a decryption key. It is one among many types of attacks organizations have struggled to deal with over the last few years. Ransomware gains access to a business or individual's servers and encrypts the data. Hackers demand a ransom, typically something that is affordable which an organization pays, rather than going through the potentially arduous and more expensive processes of recovering the data through infrastructure safeguards.Two Major classes of ransomware are Locker-ransomware, it locks the victim out of the operating system making it impossible to access the desktop and files but does not encrypt the files. The second one, crypto-ransomware: which encrypts files on the system and forces user to pay a ransom in order to regain access to the data.

1.2 The Problem

Cybersecurity, also referred to as information technology security, focuses on protecting computers, networks, programs and data from unintended or unauthorized access, change or destruction. One of the most problematic element of cybersecurity is the quick and constantly evolving nature of security risks. Number of attack perpetrators have always been bigger than the number of people trying to protect against attacks.

Several companies discovered that fighting for protection was becoming an ever increasing exercise. The biggest security infrastructure firms like Symantec, MacAfee, Palo Alto, checkpoint, etc. united to work in common initiatives, such as developing web apps against DoD attacks (web apps not in the sense of website but in firewall web apps, also called next-generation firewalls). One Important thing to pay attention over here is that they do not exchange their solutions, they exchange their attacks. This is called SecIntel Exchange.The whole idea over here is to understand how attacks are done and what types of exploits are there. However, this process is expensive and time-consuming. While a company has a team of 10 people to protect, the world will have thousands of people trying to break it. Also, signature based approaches that are commonly usedc do not provide enough opportunity to learn and understand threats that can be used in implementing prevention mechanism. In order to learn and understand malware behavior based techniques that implement dynamic approach are possible solutions. Given such circumstances, it is desirable to have a system classify new variants of malware quickly without the overhead of cost and time into families so that prevention mechanisms can be developed.

1.3 Why Machine Learning

Machine Learning systems automatically learn programs from data. we don't really code the program, but it is inferred from the data. Just like trying to mimic how the brain learns. Machine Learning algorithms can pick up the information collected quickly and give faster results. The main benefit of Machine Learning algorithms is it will learn and predict based on experience and results. It means initially if it takes 1 day to learn, next time it will take 20 hours and after that, it will take 12 hours and so on.

1.4 Why Behavior based approach

Signature based detection is an anti-malware approach that identifies the presence of malware by matching its signature with the database of known signatures. However, it is susceptible to evasions. Since the signatures are derived from known malwares, this signatures can be easily evaded by code re-ordering and inserting no-ops. Also signature based attacks cannot detect polymorphic malwares as the signatures of these new forms is not present in the database.

Whitelisting is another malware detection technique which permits only approved software

to install and run. But, this method creates a very strict and rigid rules about what software can be downloaded and installed. It can create an annoying computer experience where users are subjected to pop-up warnings constantly. It also limits users ability to easily download and use new software. Further, whitelisted applications can be vulnerable. For e.g. if you whitelist a browser then any malware that operate inside the browser will not be detected. In fact, lot of malware inject themselves into the browser.
Behavior based anti-malware approach monitor behavior of the program to determine if its malicious. Several type of behavior based detection exist. Common thing about these techniques is that they make a decision based on the actual behavior. They do not have the shortcoming of the signature based approaches but they do are susceptible to mimicry attacks also false positives.Technical indicators revealed with nehavioral analysis can include domain names, IP addresses, file path locations, registry keys, additional files located on the system or network. Additionally, it will identify communication with an attacker-controlled external server for command and control purposes or in an attempt to download additional malware files. Behavioral Analysis can reveal features which can solve the problem more efficiently with greater accuracy. Behavior based approach is more recent and more intelligent approach.

1.5 Our Proposal

We propose a system to identify most significant ransomware features based on their sandbox execution and use them to detect ransomware families. After manually analyzing and reverse engineering several known ransomware samples we concluded that the key features that could be used to classify them into their families were Registry Key Operations, API stats, and Dynamic link libraries.

1.6 Major Contributions

- We propose a framework to identify most significant ransomware dynamic features and use them to classify ransomware. Mutual Information Criteria helps identify most relevant features among a large set of features without reducing the performance of machine learning classifier.

- We evaluate the accuracy of K-Nearest Neighbour, Naive Bayes, Support Vector Machine, and Random Forest. We found out that random forest classifier outperforms Naive Bayes and it is competitive with respect to support vector machine (SVM). Also, this algorithms are easy to train and adapt.

Chapter 2

Background

Never before in the history of humankind have people across the world subjected to extortion on a massive scale as they are today. In recent years, personal use of computers and the internet has exploded and, along with this massive growth, cybercriminals have emerged to feed off this burgeoning market, targeting innocent users with a wide range of malware. The vast majority of these threats are aimed at directly or indirectly making money from the victims. Today, ransomware has emerged as one of the most troublesome malware categories of our time. There are two basic types of ransomware in circulation. The most common type today is crypto ransomware, which aims to encrypt personal data and files. The other, known as locker ransomware, is designed to lock the computer, preventing victims from using it. Despite having similar objectives, the approaches taken by each type of ransomware are quite different. In this research, we will try to some of this approaches.

2.1 Ransomware Evolution

The modern-day ransomware has evolved considerably since its origins 26 years ago with the appearance of the AIDS Trojan. The AIDS Trojan was released into the unsuspecting world through snail mail using 5 floppy disks in 1989. Despite the public being unprepared for this new type of threat all those years ago, the AIDS Trojan was ultimately unsuccessful due to a number of factors. Back then, few people used personal computers, the World Wide Web was just an idea, and the internet was mostly used by experts in the field of science and technology. The availability and strength of encryption technology was somewhat limited at that time. Along with this, international payments were harder to process than they are today.The evolution of ransomware, particularly crypto ransomware, accelerated in recent years.

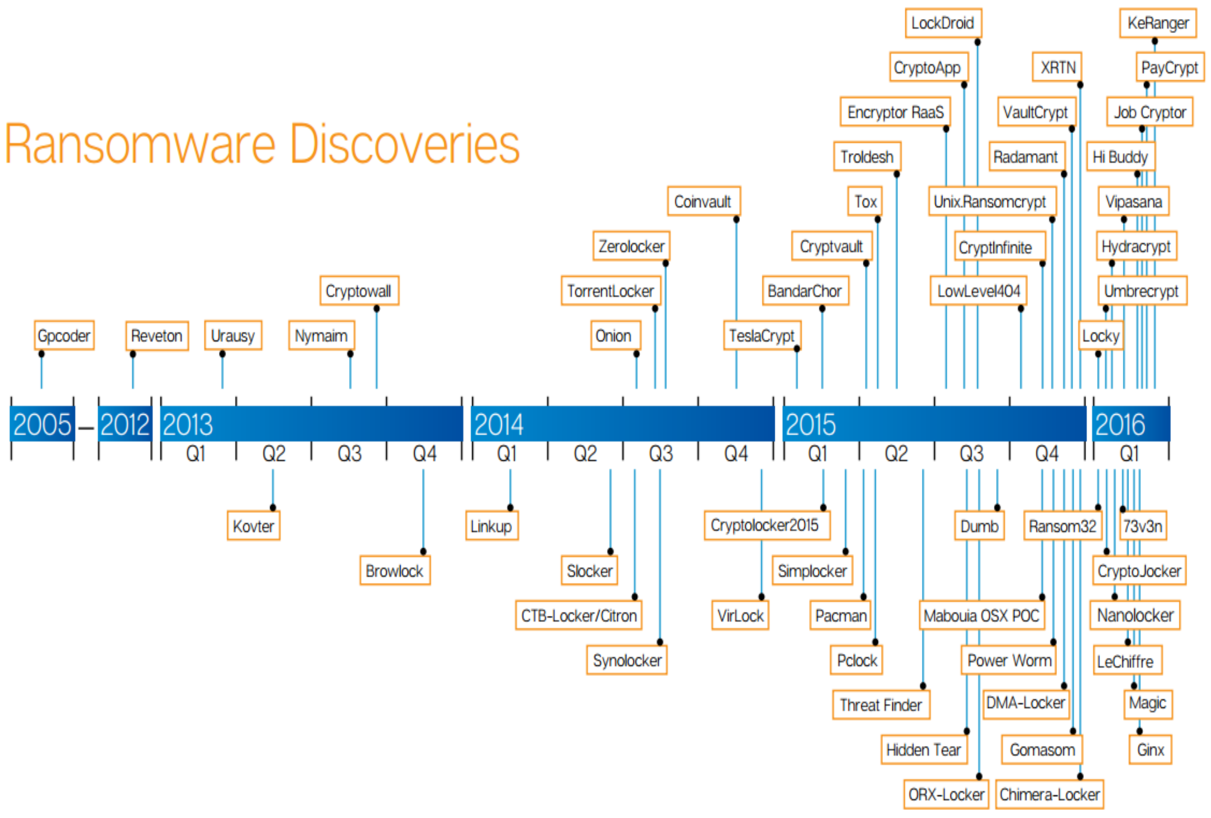


Figure 2.1: Timeline of Ransomware Discovered in past 10 years.

Figure 2.1 shows just how many types of encrypting malware researchers have discovered in the past 10 years. There are numerous variants for each type. It is difficult to map all the existing ransomware out there because most of the ransomware attacks go unreported.

Ransomware install's itself on user's computer via several techniques.Table 2.1 shows some Notable features of ransomware along with their family names and year. As we look at the recent history of ransomware, it is useful to consider the overall picture of money payment extortion threats over the past 10 years to get an idea of where modern-day ransomware evolved from. Figure 2.2 shows how the market for extortion malware has been divided up

Table 2.1: Ransomware and their Notable Features.

Name	**Year**	**Notable Features**
GPCoder	2005-2008	Spreads via emails; encrypts a large set of files
Archiveus	2006	First Ransomware to use RSA encryption
WinLock	2010	Blocks PCs by displaying a ransom message
Reveton	2012	Warning purportedly from a law enforcement agency
DirtyDecrypt	2013	Encrypts eight different file formats
CryptLocker	2013	Fetches a public key from the C & C
CryptoWall	2013	Requires TOR browser to make payments
Android Defender	2013	First Android locker-ransomware
TorDroid	2014	First Android crypto-ransomware
Critroni	2014	Similar to CryptoWall
TorrentLocker	2014	Stealthiness:indistinguishable from SSH connections
CTB-Locker	2014	Uses Elliptic Curve Cryptography, TOR and Bitcoins
CryptoWall 3.0	2015	Uses exclusively TOR for payment
TeslaCrypt	2015	Adds the option to pay with PayPal My Cash Cards
Hidden Tear	2015	Open source ransomware released for educational purposes
Chimera	2015	Threatens to publish users personal files
CryptoWall 4.0	2015	Encrypts also filenames
Linux.Encoder.1	2015	Encrypts Linuxs home and website directories
DMA-Locker	2016	Comes with a decrypting feature built-in
PadCrypt	2016	Live Chat Support
Locky Ransomware	2016	Installed using malicious macro in a Word document
CTB-Locker for WebSites	2016	Targets Wordpress
KeRanger	2016	First ransomware for Apples Mac computers
Cerber	2016	Offered as RaaS (& quote in Latin)
Samas	2016	Pentesting on JBOSS servers
Petya	2016	Overwrites MBT with its own loader and encrypts MFT
Rokku	2016	Use of QR code to facilitate payment
Jigsaw	2016	Press victims into paying ransom
CryptXXX	2016	Monitors mouse activities and evades sandboxed environment
Mischa	2016	Installed when PETYA fails to gain administrative privileges
RAA	2016	Entirely written in Javascript
Satana	2016	Combines the features of PETYA and MISCHA
Stampado	2016	Promoted through aggressive advertising campaigns on the Dark web
Fantom	2016	Uses a rogue Windows update screen
Cerber3	2016	Third iteration of the Cerber ransomware

each year since 2005. While each threat never disappeared entirely, its easy to identify how preferences shifted from one type of extortion malware to another.

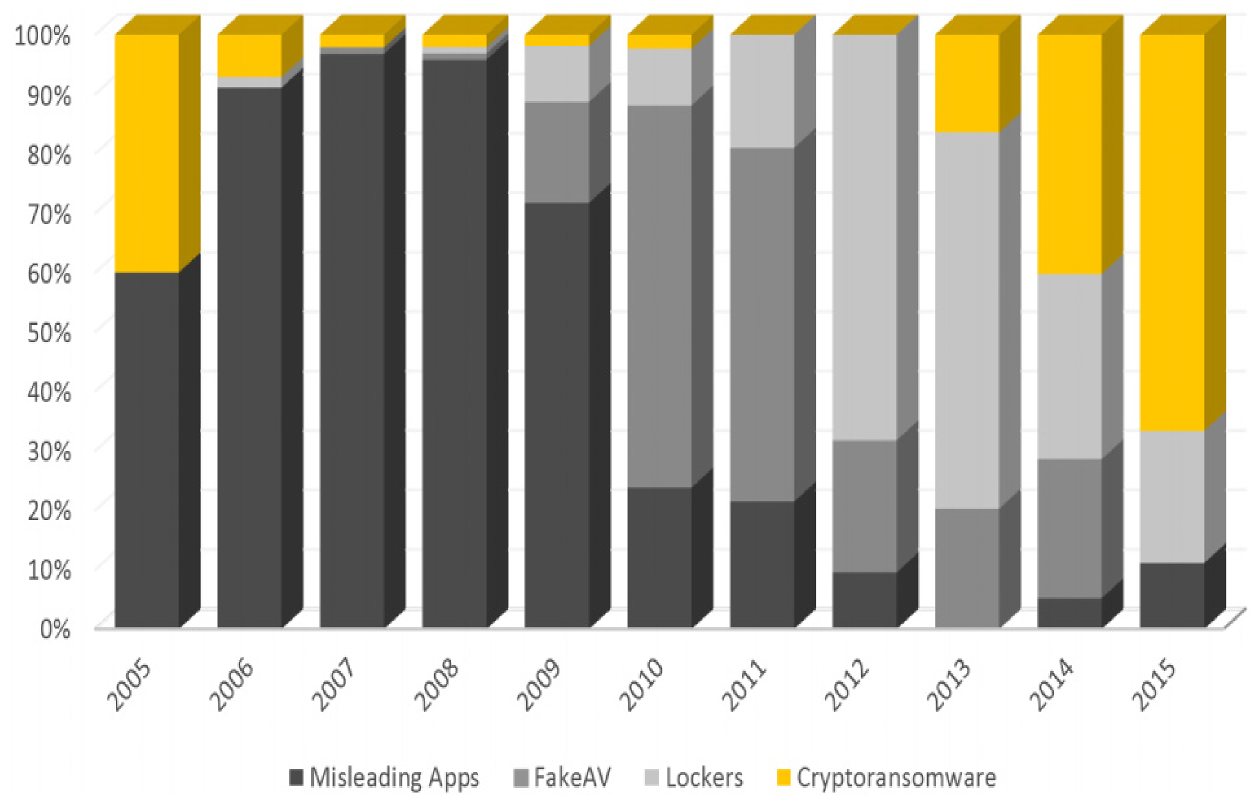


Figure 2.2: Percentage of new families of misleading apps, fake AV, locker ransomware and crypto ransomware identified between 2005 and 2015.

2.2 Propagation

Carrying out digital extortion using ransomware is a carefully planned and executed process for cybercriminals. Even a single weakness in the operation could cause the whole scheme to fail. There are much more elements to a ransomware attack than just the malware.

2.2.1 Spam email

email spam using social-engineering themes has been the method of choice for distributing all types of malware including ransomware. Cybercriminals use a botnet to send the spam. These cybercriminals may also offer a spamming service to other attackers for a fee. Sometimes .src (Microsoft Windows Screensaver) files are disguised as .pdf with a fake icon, to lure victims into opening it; in any case, a malware sample is downloaded, and this malware downloads or drops the ransomware sample.

2.2.2 Downloaders & botnets

This method is one of a number of ways to distribute malware known as downloaders. Once the downloader infects a computer, its job is to download secondary malware onto the compromised system. The cybercriminals behind downloaders offer a malware-installation service onto already compromised computers, at a price to other malware authors. Trojan botnets have also been known to download ransomware onto computers they have infected. This is usually done by cybercriminals as a final way of monetizing infected computers that they control.

2.2.3 Malvertisement

Similarly, malicious advertisements known as malvertisments can get pushed onto legitimate websites in order to redirect traffic to a site hosting an exploit kit. In one case, we even observed unintentional cross-contamination as a result of a click-fraud malware infection, where clicking on the malvertisment led to a ransomware infection. In both cases, cyber-criminals can use real-time bidding to purchase traffic or ad space of interest that can allow them to geographically target victims and operate without borders.

2.2.4 Traffic distribution system (TDS)

A common method used by these distribution services is to buy redirected web traffic from a Traffic Distribution Service (TDS) vendor and point it to a site hosting an exploit kit. In a lot of cases, the redirected traffic originates from adult content-related websites. If the exploit kit is successful in exploiting a vulnerability in the visiting victims computer, it can lead to what is commonly referred to as the drive-by-download of malware.

2.2.5 Self-propagation

Some ransomware also contains functionality to spread. For example, on Android, there are some samples that not only lock the device or encrypt files, but employ worm-like capabilities to spread to all contacts within the devices address book by sending social-engineering SMS messages.

2.3 Method of Payment

Over the years, the options and preferred methods of payment have changed as different services became available. In 1989, the AIDS crypto ransomware Trojan demanded payment by way of a check sent to a post office box in Panama.Since then, other payment methods have been used by ransomware. These methods include money wire transfers and sending premium-rate text messages to the attackers number, as seen in Trojan. Ransomlock in 2009. More recently, the use of payment voucher systems such as Paysafecard, MoneyPak, UKash, CashU, and MoneXy have and are still being used by some ransomware threats. The arrival of cryptocurrencies in the form of Bitcoins in 2009 shook up the money transfer landscape.The increasingly widespread acceptance of bitcoins made it easier for victims to purchase them to make ransom payments and then for the cybercriminals to convert them back into hard cash later.

2.3.1 Prefered Method of Payment

In general, we found that crypto ransomware tends to favor cryptocurrencies as the preferred payment method whereas locker ransomware prefers to use payment voucher systems. A possible reason for this is because of the way that the two different types of ransomware work. Locker ransomware locks the computer leaving it largely unusable. Therefore it would not be possible for victims to buy online currencies or access Bitcoin wallets using the computer to make payment. If the computer is locked, it would be easier for victims to buy payment vouchers from a local shop or outlet and then enter the payment code.

2.4 Machine Learning Algorithms

Here we give a theoretical background on machine learning methods.First, the overview of the machine learning field is discussed, followed by the description of methods relevant to this study.

2.4.1 Machine Learning Basics

The rapid development of data mining techniques and methods resulted in Machine Learning forming a separate field of Computer Science. It can be viewed as a subclass of the Artificial Intelligence field, where the main idea is the ability of a system (computer program, algorithm, etc.) to learn from its own actions. It was firstly referred to as "field of study that gives computers the ability to learn without being explicitly programmed" by Arthur Samuel in 1959. A more formal definition is given by T. Mitchell: "A computer program is said to learn 13 from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." (Mitchell 1997). The basic idea of any machine learning task is to train the model, based on some algorithm, to perform a certain task: classification, clusterization, regression, etc. Training is done based on the input dataset, and the model that is built is subsequently used to make predictions. The output of such model depends on the initial task and the implementation.

Possible applications are: given data about house attributes, such as room number, size, and price, predict the price of the previously unknown house; based on two datasets with healthy medical images and the ones with tumor, classify a pool of new images; cluster pictures of animals to several clusters from an unsorted pool.

2.4.2 Supervised and Unsupervised Learning

Here we want to introduce the two machine learning approaches - supervised and unsupervised learning. In Supervised Learning, learning is based on labeled data. In this case, we have an initial dataset, where data samples are mapped to the correct outcome. The housing prices case is an example of supervised learning: here we have an initial dataset with houses, its attributes, and its prices. The model is trained on this dataset, where it knows the correct results. Examples of supervised learning are regression and classification problems:

- **Regression:** Predict the value based on previous observations, i.e. values of the samples from the training set Usually, we can say that if the output is a real number/is continuous, then it is a regression problem.The main goal of regression algorithms is the predict the discrete or a continuous value. In some cases, the predicted value can be used to identify the linear relationship between the attributes. Suppose the increase in the product advantage budget will increase the product sales. Based on the problem difference regression algorithms can be used. some of the basic regression algorithms are linear regression, polynomial regression etc.

- **Classification:** classification is the problem of identifying to which of a set of categories a new sample belongs, on the basis of a training set of data containing observations whose category membership is known. An example would be assigning a given email into ”spam” or ”non-spam” classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.).The main goal of classification is to predict the target class (Yes/ No). If the trained model is for predicting any of two target classes.

It is known as binary classification. Considering the student profile to predict whether the student will pass or fail. Considering the customer, transaction details to predict whether he will buy the new product or not. These kind problems will be addressed with binary classification. If we have to predict more the two target classes it is known as multi-classification. Considering all subject details of a student to predict which subject the student will score more. Identifying the object in an image. These kind problems are known as multi-classification problems.

In contrast to Supervised Learning, in Unsupervised Learning, there is no initial labeling of data. Here the goal is to find some pattern in the set of unsorted data, instead of predicting some value. A common subclass of Unsupervised Learning is Clustering:

- **Clustering:** Finds the hidden patterns in the unlabeled data and separates it into clusters according to similarity. An example can be the discovery of different customer groups inside the customer base of the online shop.

2.4.3 KNN (K-Nearest Neighbours)

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. KNN algorithm is one of the simplest classification algorithms. Even with such simplicity, it can give highly competitive results.

Lets first start by establishing some definitions and notations. We will use x to denote a feature (aka. predictor, attribute) and y to denote the target (aka. label, class) we are trying to predict.KNN falls in the supervised learning family of algorithms. Informally, this means that we are given a labeled dataset consisting of training observations (x, y) and would like to capture the relationship between x and xy. More formally, our goal is to learn a function $h : X \to Y$ so that given an unseen observation $x, h(x)$ can confidently predict the corresponding output y.

The value of k plays a crucial role in the prediction accuracy of the algorithm. However, selecting the k value is a non-trivial task. Smaller values of k will most likely result in lower accuracy, especially in the datasets with much noise, since every instance of the training set now has a higher weight during the decision process. Larger values of k lower the performance of the algorithm. In addition to that, if the value is too high, the model can overfit, making the class boundaries less distinct and resulting in lower accuracy again. As a general approach, it is advised to select k using the formula below:

$$k = \sqrt{x}$$

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given unseen observation. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance given by $d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + ... + (x_n - x'_n)^2}$ but other measures can be more suitable for a given setting and include the Manhattan, Chebyshev and Hamming distance. More formally, given a positive integer K, an unseen observation x and a similarity metric d, KNN classifier performs the following two steps:

- It runs through the whole dataset computing dd between x and each training observation. Well call the K points in the training data that are closest to x the set A. Note that K is usually odd to prevent tie situations.

- It then estimates the conditional probability for each class, that is, the fraction of points in A with that given class label. (Note $I(x)$ is the indicator function which evaluates to 1 when the argument x is true and 0 otherwise)

$$P(y = j | X = x) = \frac{1}{K} \sum_{i \in A} I(y^i = j)$$

The drawback of the KNN algorithm is the bad performance on the unevenly distributed datasets. Thus, if one class vastly dominates the other ones, it is more likely to have more neighbors of that class due to their large number, and, therefore, make incorrect predictions.

2.4.4 Naive Bayes

It is a classification technique based on Bayes Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as Naive. It can be used for both binary and multi-class classification problems. The main point relies on the idea of treating each feature independently. Naive Bayes method evaluates the probability of each feature independently, regardless of any correlations, and makes the prediction based on the Bayes Theorem. That is why this method is called naive in real-world problems features often have some level of correlation between each other.

Naive Bayes model is easy to build and particularly useful for very large datasets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

To understand the algorithm of Naive Bayes, the concepts of class probabilities and conditional probabilities should be introduced first.

- **Class Probability:** is a probability of a class in the dataset. In other words, if we select a random item from the dataset, this is the probability of it belonging to a certain class.

- **Conditional Probability:** is the probability of the feature value given the class.

1. Class probability is calculated simply as the number of samples in the class divided by the total number of samples:

$$P(C) = \frac{count(instanceinC)}{count(instanceinNtotal)}$$

2. Conditional probabilities are calculated as the frequency of each attribute value divided by the frequency of instances of that class.

$$P(V|C) = \frac{count(instancewithVandC)}{count(instancewithV)}$$

3. Given the probabilities, we can calculate the probability of the instance belonging to a class and therefore make decisions using the Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

4. Probabilities of the item belonging to all classes are compared and the class with the highest probability if selected as a result.

Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c)$, $P(x)$ and $P(x|c)$. Look at the equation below:

$$P(c|x) = \frac{P(x|c).P(c)}{P(x)}$$
$$P(c|x) = P(x_1|c)xP(x_2|c)x...xP(x_n|c)xP(c)$$

- $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes).
- $P(c)$ is the prior probability of class.
- $P(x|c)$ is the likelihood which is the probability of predictor given class.
- $P(x)$ is the prior probability of predictor.

The advantages of using this method include its simplicity and easiness of understanding. In addition to that, it performs well on the data sets with irrelevant features, since the probabilities of them contributing to the output are low. Therefore they are not taken into account when making predictions. Moreover, this algorithm usually results in a good performance in terms of consumed resources, since it only needs to calculate the probabilities of the features and classes, there is no need to find any coefficients like in other algorithms. As already mentioned, its main drawback is that each feature is treated independently, although in most cases this cannot be true. (Bishop 2006).

2.4.5 SVM (Support Vector Machine)

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyperplane that differentiates the two classes very well (look at the below snapshot).

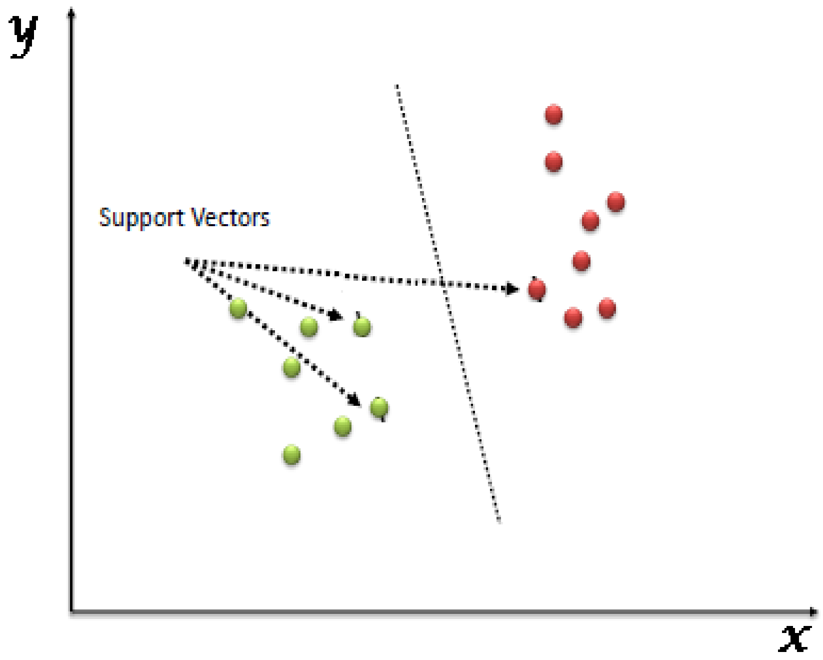


Figure 2.3: Support Vector Machine Planes.

Intuitively, we understand that the further from the hyperplane our classes lie, the more accurate predictions we can make. That is why, although multiple hyperplanes can be found per problem, the goal of the SVM algorithm is to find such a hyperplane that would result in the maximum margins.

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).
The Algorithm can be described as follows:

1. We define X and Y as the input and output sets respectively. $(x1, y1), ...(xm, ym)$ is the training set.

2. Given x, we want to be able predict y. We can refer to this problem as to learning the classifier $y = f(x, a)$, where a is the parameter of the classification function.

3. $F(x, a)$ can be learned by minimizing the training error of the function that learns on training data. Here, L is the loss function, and $Remp$ is referred to as empirical risk. $Remp(a) = \frac{1}{m}\sum_{1}^{m} l(f(xi, a), yi) =$ Training Error

4. We are aiming at minimizing the overall risk, too. Here, $P(x, y)$ is the joint distribution function of x and y. $R(a) = \int l(f(x, a), y)dp(x, y) =$ Test Error

5. We want to minimize the training error+complexity term. So we choose the set of hyperplanes, So $f(x) = (w, x) + b : \frac{1}{m}\sum_{i=1}^{m} l(w.xi + b, yi) + ||w||^2$ subject to mini $|w.xi| = 1$

SVMs are generally able to result in good accuracy, especially on clean datasets. Moreover, it is good with working with the high-dimensional datasets, also when the number of dimensions is higher than the number of the samples. However, for large datasets with a lot of noise or overlapping classes, it can be more effective. Also, with larger datasets training time can be high. (Jing and Zhang 2010).

- pros :
  - It works really well with clear margin of separation.
  - It is effective in high dimensional spaces.

  - It is effective in cases where number of dimensions is greater than the number of samples.
  - It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

- cons:

  - It doesnt perform well, when we have large data set because the required training time is higher
  - It also doesnt perform very well, when the data set has more noise i.e. target classes are overlapping
  - SVM doesnt directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

2.4.6 RFC (Random Forest Classifier)

Random forest algorithm is a supervised classification algorithm. As the name suggests, this algorithm creates the forest with a number of trees.

In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results. Decision tree concept is more to the rule-based system. Given the training dataset with targets and features, the decision tree algorithm will come up with some set of rules. The same set rules can be used to perform the prediction on the test dataset. Suppose you would like to predict that your daughter will like the newly released animation movie or not. To model the decision tree you will use the training dataset like the animated cartoon characters your daughter liked in the past movies.

So once you pass the dataset with the target as your daughter will like the movie or not to the decision tree classifier. The decision tree will start building the rules with the characters your daughter like as nodes and the targets like or not as the leaf nodes. By considering the path from the root node to the leaf node. You can get the rules.

The simple rule could be if some x character is playing the leading role then your daughter will like the movie. You can think few more rule based on this example. Then to predict whether your daughter will like the movie or not. You just need to check the rules which are created by the decision tree to predict whether your daughter will like the newly released movie or not.In decision tree algorithm calculating these nodes and forming the rules will happen using the information gain and Gini index calculations.In random forest algorithm, Instead of using information gain or Gini index for calculating the root node, the process of finding the root node and splitting the feature nodes will happen randomly. Will look about in detail in the coming section.

Random Forest PseudoCode:

- Randomly select k features from total m features.
- Among the k features, calculate the node d using the best split point.
- Split the node into daughter nodes using the best split.
- Repeat 1 to 3 steps until l number of nodes has been reached.
- Build forest by repeating steps 1 to 4 for n number times to create n number of trees.

Random forest prediction pseudocode:

- Takes the test features and use the rules of each randomly created decision tree to predict the oucome and stores the predicted outcome (target)
- Calculate the votes for each predicted target.
- Consider the high voted predicted target as the final prediction from the random forest algorithm.

To perform the prediction using the trained random forest algorithm we need to pass the test features through the rules of each randomly created trees. Suppose lets say we formed 100 random decision trees to from the random forest.

Each random forest will predict different target (outcome) for the same test feature. Then by considering each predicted target votes will be calculated. Suppose the 100 random decision trees are prediction some 3 unique targets x, y, z then the votes of x is nothing but out of 100 random decision tree how many trees prediction is x. Likewise for other 2 targets (y, z). If x is getting high votes. Lets say out of 100 random decision tree 60 trees are predicting the target will be x. Then the final random forest returns the x as the predicted target.

This concept of voting is known as majority voting.

Chapter 3

Related Work

Never before in the history of human kind have people across the world been subjected to extortion on a massive scale as they are today. In recent years, personal use of computers and the internet has exploded and, along with this massive growth, cybercriminals have emerged to feed off this burgeoning market, targeting innocent users with a wide range of malware. The vast majority of these threats are aimed at directly or indirectly making money from the victims. Today, ransomware has emerged as one of the most troublesome malware categories of our time.

[20] This paper proposes a method to extract valuable features of windows PE files. They have used information gain to calculate the frequencies of raw features and principal component analysis to reduce the dimensionality of selected features. The architecture of this system has three main modules, PE-Miner, feature selection and transformation and machine learning algorithms. PE miner parses the PE tables of all executables to find out all PE header information, DLL names, and API functions inside each DLL as raw features. Information Gain value of each PE header, and calling frequency of each DLL and API function inside DLL are computed and those raw features with Information Gain values and calling frequencies greater than a threshold are selected then Principal Component Analysis is applied to further transform and reduce the number of features. Finally, machine learning algorithm is applied to the labelled feature vectors. The authors dataset consisted of 2,36,756 malicious and 10,592 clean programs. In the final dataset the number of features included are 88 PE header files, 130 DLL and 2453 API function features to train the system. Three classification algorithms SVM, J48 and NB classifiers were used and the detection accuracy of J48

outperformed the others with an accuracy of 99.6

[18] Shows how to perform behavioral analysis using dynamic approach. The author performs experiment to learn and understand the current malware. The method is divided into 4 steps such as malware collection, behavior identification, custom behavior analysis and statistical report. Experiment was performed at a higher learning institution that had more than 4,000 network users. For malware collection the author used two tools Honey Clients and Anum which ran on the network for 30 days in order to collect thousands of different malware sample. In the Behavior Identification step two sandboxes, CWSandbox and Anubix are used. Each sample is ran on both the sandboxes to get variants of malware behaviors. In the custom behavior analysis all the collected malware samples are grouped in two main groups, worm and Trojan and their types. There are 3 types of worm and 7 types of Trojans. Total of 1073 samples were analyzed and manually classified in this experiment. No other approach for classification was used.

[8] shows how ransomware have evolved considerably in the last few years both in terms of features and spreading capabilities.The authors also pinpoint future directions of ransomware evolution, such as targeting the wearable market (the so-called ransomware).[2] the authors propose a framework to automatically analyze malware behavior through ML. In particular, the framework allows researchers to identify novel classes of malware having similar behavior, and to assign labels to the newly discovered classes.In contrast to that, we have shown the importance of feature selection to reduce the complexity of machine learning algorithms without affecting the performance.

In [7] Authors have analyzed 15 different families by showing that large majority of samples implement locking or encrypting techniques through naive techniques. The authors describe simple techniques to detect and stop ransomware, such as monitoring abnormal file activities.

[21]In this paper the author tries to make use of different machine learning algorithms namely Nave Bayes, k-Nearest Neighbor and j48 decision tree to detect zero-day malwares. The paper sums up the results as: malware authors are easily able to fool the detection engine

by applying obfuscation techniques. Identifying benign files as malware is becoming very high, failing to detect obfuscated malware is high, current detection rate is decreasing and current detectors are unable to detect zero-day attacks. The system unpacks the malware and disassembles the binary executable to retrieve the assembly program. Then extracts API calls and important machine-code features from assembly program. Then, maps the API calls with MSDN library and analyze the malicious behavior to get the API sequence from the binaries. After getting the API sequences from the binaries the signature database is updated based on API calls. The extracted sequence is compared to a sequence in the database and is passed through a similarity module to know if its malware or benign. The signature database was made using existing malware datasets. They have gathered 51,223 malwares and 15,480 benign executables for the experiment. Experiments were performed using 8 different machine learning algorithms. It does not need the API sequence of new unseen malware as by finding out its similarity with malware sequences it is classified into benign or malware.

[1]HelDroid is a system for recognizing Android ransomware using their typical characteristic, such as functions to lock screen. The detection results for HelDroid are better due to the fact that the sophistication, and variance, of ransomware for Android, is quite limited with respect to that for Windows. [4] shows android malware classification using SVM and conformal prediction.

[19] This paper discusses various anti-malware approaches and their disadvantages. Signature based detection is an anti-malware approach that identifies the presence of malware by matching its signature with the database of known signatures. However, it is susceptible to evasions. Since the signatures are derived from known malwares, this signatures can be easily evaded by code re-ordering and inserting no-ops. Also signature based attacks cannot detect polymorphic malwares as the signatures of these new forms is not present in the database. Whitelisting is another malware detection technique which permits only approved software to install and run. But, this method creates a very strict and rigid rules about what

software can be downloaded and installed. It can create an annoying computer experience where users are subjected to pop-up warnings constantly. It also limits users ability to easily download and use new software. Further, whitelisted applications can be vulnerable. For e.g. if you whitelist a browser then any malware that operate inside the browser will not be detected. In fact, lot of malware inject themselves into the browser. Behavior based anti-malware approach monitor behavior of the program to determine if its malicious. Several type of behavior based detection exist. Common thing about these techniques is that they make a decision based on the actual behavior. They do not have the shortcoming of the signature based approaches but they do are susceptible to mimicry attacks also false positives. Behavior based approach is more recent and more intelligent approach.

Chapter 4

Tracing Ransomware Behavior

In this Chapter we discuss how we generate the behavioral data used for classification

4.1 Cuckoo Sandbox

The study is based on and targeted to Cuckoo Sandbox. It is clear that to apply the machine learning algorithms to any problem, it is essential to represent the data in some form. For this purpose, Cuckoo Sandbox was used. The reports generated by the sandbox, describing the behavioral data of each sample, were preprocessed, and malware features were extracted from there. However, it is important to understand the functionality of the sandbox and the structure of the reports first. Cuckoo Sandbox is the open-source malware analysis tool that allows getting the detailed behavioral report of any file or URL in a matter of seconds. According to Cuckoo Foundation (2015), currently, supported file formats include:

- Generic Windows executables
- DLL files
- PDF documents
- Microsoft Office documents
- URLs and HTML files
- PHP scripts
- CPL files

- Visual Basic (VB) scripts
- ZIP files
- Java JAR
- Python files
- Almost anything else

Cuckoo has a highly customizable modular architecture, allowing it to be used both as a standalone application as well as integrated into the larger frameworks. The main components of Cuckoos infrastructure are a host machine (the management software) and a number of guest machines (virtual or physical machines for analysis). Its operation scenario is quite straight forward: as soon as the new file is submitted to the server, a virtual environment is dynamically allocated for it, the file is executed, and all the actions performed in the system are recorded.

4.2 Reports and Features

To apply machine learning algorithms to the problem, we need to figure out what kind of data should be extracted and how it should be presented. Some works in the field are utilizing string properties or file formats properties as a basis for feature representation. For example, for Windows-based malware samples, the data contained in PE headers is often used as a base for analysis. However, implementing format-specific feature extraction is not the best solution, since formats of analyzed files can vary dramatically. (Hung 2011). Other works rely on the so-called n-grams. Byte n-grams are overlapping substrings, collected in a sliding-window fashion where the windows of fixed size slides one byte at a time. Word n-grams and character n-grams are widely used in natural language processing, information retrieval, and text mining. (Reddy and Pujari 2006). However, such approach has several disadvantages. The major difficulty in considering byte n-grams as a feature is that the set of all byte n-grams obtained from the set of byte strings of malware as well as of the benign programs is very large and it is not useful to apply classification techniques directly on these. (Reddy and Pujari 2006). In addition to that, such approach limits the ability of detection of polymorphic malware. In this case, the samples generating the same behavior will result in different strings, and, therefore, different n-grams. Because of the above-mentioned reasons, in this study, it is decided to rely on the actual behavior of the files, that is monitored by the sandbox. Overall, we can identify the following features extracted by the sandbox:

- **Files**
- **Registry Keys**
- **Mutexes**
- **Processes**
- **IP addresses and DNS queries**
- **API calls**

Below we discuss which of them should be used:

- **Files:** The reports contain information about opened files, written files, and created files. This kind of information is good in predicting the malware family since any malware files trigger many modifications to the file systems. It can be used for the quite accurate malware classification in most cases.
- **Registry Keys:** On Windows systems, the registry stores the low-level system settings of the operation system and its applications. Any sample that is run on the system triggers a high amount of the registry changes the Cuckoo reports can outline the registry keys opened, read, written, deleted. The information on the registry modifications can be a good source of information on the system changes caused by malware and can be used for malware detection.

- **Mutexes:** The mutex stands for the Mutual Exclusion. This is a program object that allows multiple threads to share the same resource. Every time a program is started, a mutex with a unique name is created. Mutex names can be good identifiers of specific malware samples. However, for the families, they cannot result in the accurate result on a large scale, since the number of mutexes created per sample is dramatically lower than the dataset. That is why the small change related to the bug or non-started process would result in the dramatic change of the prediction results.

- **Processes:** Common identifier of the specific malware sample is the name of the created process. However, very rarely it can be used for identification of the malware family since in the common cases the process names are the same as the hash of the sample. As an alternative, the malware sample can inject itself into the system process. That is why this feature is bad for the family identification.

- **IP addresses and DNS queries:** Cuckoo provides information about the network traffic in the PCAP format, from which the data about contacted IP addresses as well as DNS queries can be extracted. This data accurately identifies the IP addresses of the command and control servers of attackers and, therefore, can accurately identify the malware family in most cases. However, often the attackers change the domain names or IP addresses of their servers or spoof them. Therefore, it is unreliable to rely solely on this kind of information.

- **API calls:** API stands for Application Programming Interface and refers to the set of tools that provide an interface for communication between different software components. API calls are recorded during the execution of the malware and refer to the specific process. They outline everything happening to the operating system, including the operations on the files, registry, mutexes, processes and other features mentioned earlier. For example, API calls OpenFile, OpenFileEx, CreateFile, CopyFileEx, etc. define the file operations, calls OpenMutex, CreateMutex and CreateMutexEx describe mutexes opened and created, etc. API call traces present the wide description of the sample behavior, including all the properties mentioned above. In addition to that, they include a wide set of distinct values. Moreover, they are simple to describe in numeric format, and that is why they were chosen as features. Here, the feature set will be defined by the number of unique API calls and the return codes.

4.3 System Overview

This system uses Cuckoo as its dynamic analysis component. In detail, we first dynamically analyze the ransomware samples in a sanbox environmeent. After executing the sample cuckoo generates a report from which we used the following features :

- Windows API calls (i.e., the traces of invocation of native functions and windows API calls).
- Registry Key Operations (in particular which registry are read opened and written).
- Dll Files Used. (which Dll files are used).
- Extracted Stings

After the monitoring phase a feature selection algorithm is applied to select the most relevant features. Finally these feature matrix is used in the SVM, Naive Bayse, KNN and Ransom Forest Classifier which returns the family name.

4.4 Dataset

The ransomware samples in our dataset were downloaded from VirusShare a website that maintains a continuously updated database of malware for several OSes. Table 5.2 reports the full list of ransomware families used for our experiments.To analyze the samples we used the Cuckoo Sandbox, a well-known tool to automate malware analysis. The dataset consists of 1584 total ransomware samples. We tried to collect both old and new variants of

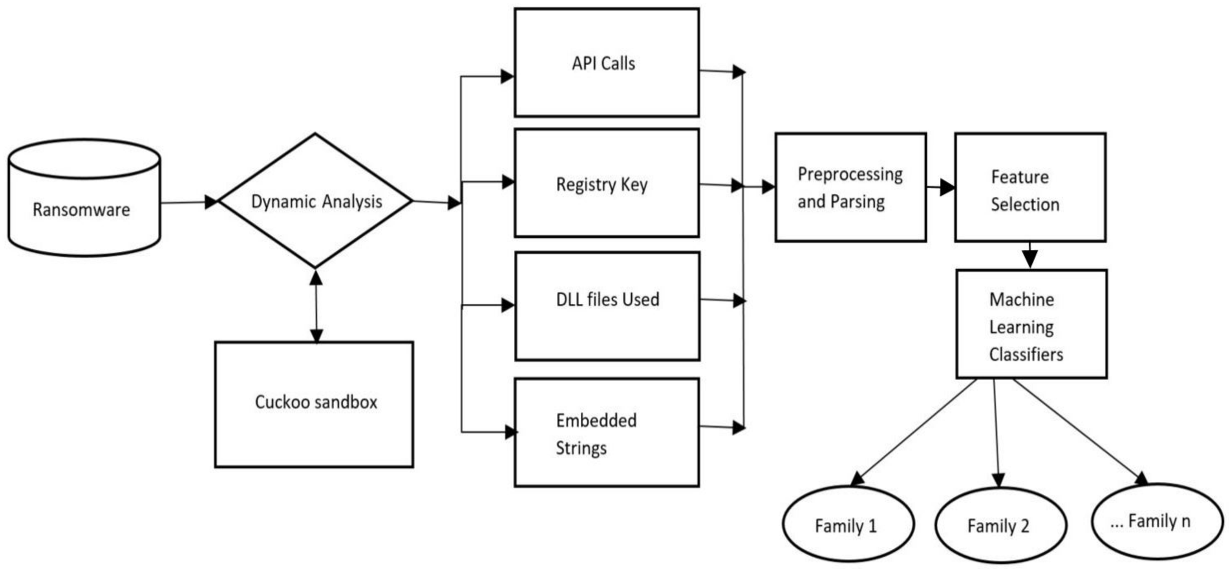
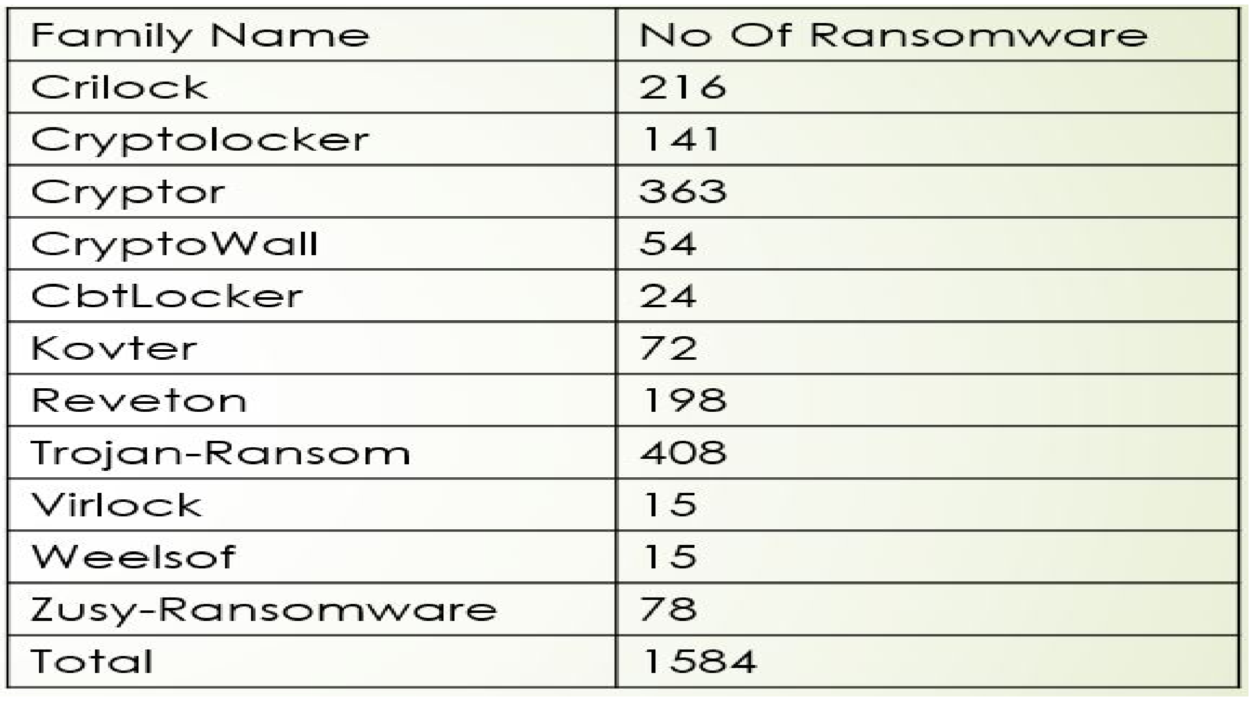


Figure 4.1: SandBox Training and Analysis

windows ransomware. There are variations in AV vendors naming strategy, therefore, it was not easy to get a common name for each ransomware family. So we used a software called Auto Malware Labelling to get the labels for ransomware.

4.5 Sandbox

The System used during execution was Windows XP 32 bit with Microsoft office, adobe viewer and some random word and images(.jpeg,.png) stored in different directories. We used this version of Windows as its weaker security protections enables us to observe more ransomware behavior. No other user application was currently running in the machine while analyzing the malware and the system was not actively used by any other people. Before we tested next ransomware sample we restored the state of the virtual machine back to the original/safe state. We did not have network connectivity to the virtual machine and focused mainly on host-based features.



Family Name	No Of Ransomware
Crilock	216
Cryptolocker	141
Cryptor	363
CryptoWall	54
CbtLocker	24
Kovter	72
Reveton	198
Trojan-Ransom	408
Virlock	15
Weelsof	15
Zusy-Ransomware	78
Total	1584

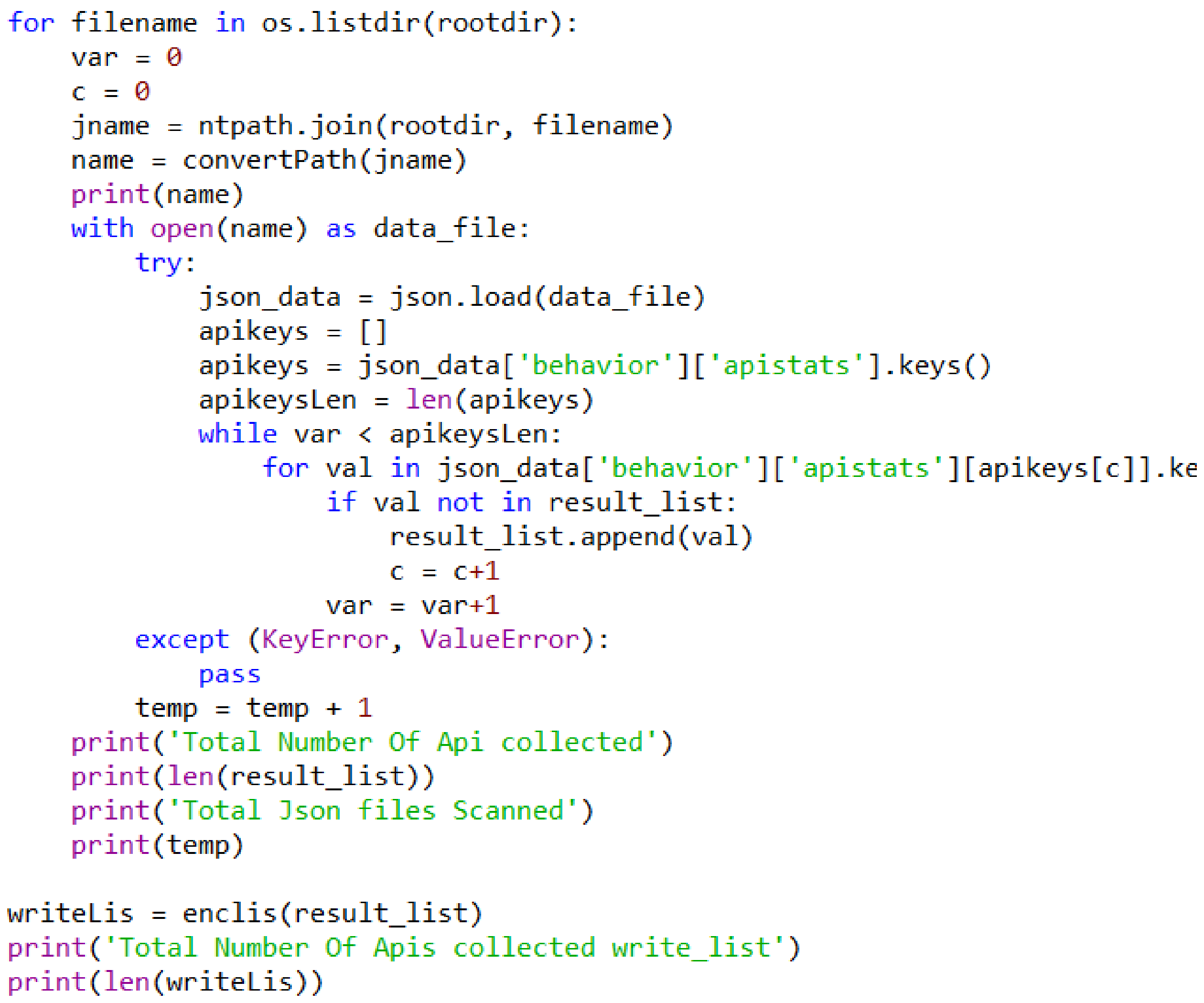
Figure 4.2: Number Of Ransomware

4.6 Preprocessing of Data

All the reports generated by the cuckoo sandbox were in JSON (JavaScript Object Notation) format. The reports had a lot of data into it and we were particularly interested in certain dynamic features. So these data had to be preprocessed. Figure 4.3 shows an example of how the JSON file looked.

We build a feature parser that retrieved all these reports and generated a super label which had all the API stats names, DLL files names and RegKey names from the 1584 ransomware sample.

Then we created a matrix with the column names as this SuperLabel and each row for each ransomware sample with '1' denoting the presence of that feature in that report and '0' denoting the absence of that feature in that report.Table 4.1 shows the details and number



```python
for filename in os.listdir(rootdir):
    var = 0
    c = 0
    jname = ntpath.join(rootdir, filename)
    name = convertPath(jname)
    print(name)
    with open(name) as data_file:
        try:
            json_data = json.load(data_file)
            apikeys = []
            apikeys = json_data['behavior']['apistats'].keys()
            apikeysLen = len(apikeys)
            while var < apikeysLen:
                for val in json_data['behavior']['apistats'][apikeys[c]].ke
                    if val not in result_list:
                        result_list.append(val)
                        c = c+1
                    var = var+1
        except (KeyError, ValueError):
            pass
        temp = temp + 1
    print('Total Number Of Api collected')
    print(len(result_list))
    print('Total Json files Scanned')
    print(temp)

writeLis = enclis(result_list)
print('Total Number Of Apis collected write_list')
print(len(writeLis))
```

Figure 4.3: Example of code used for API Labels extraction

of features in the DataMatrix. After analyzing all the samples of ransomware total number of feature we collected was 15,967. Out of this features top 1521 features belonged to the following general category.

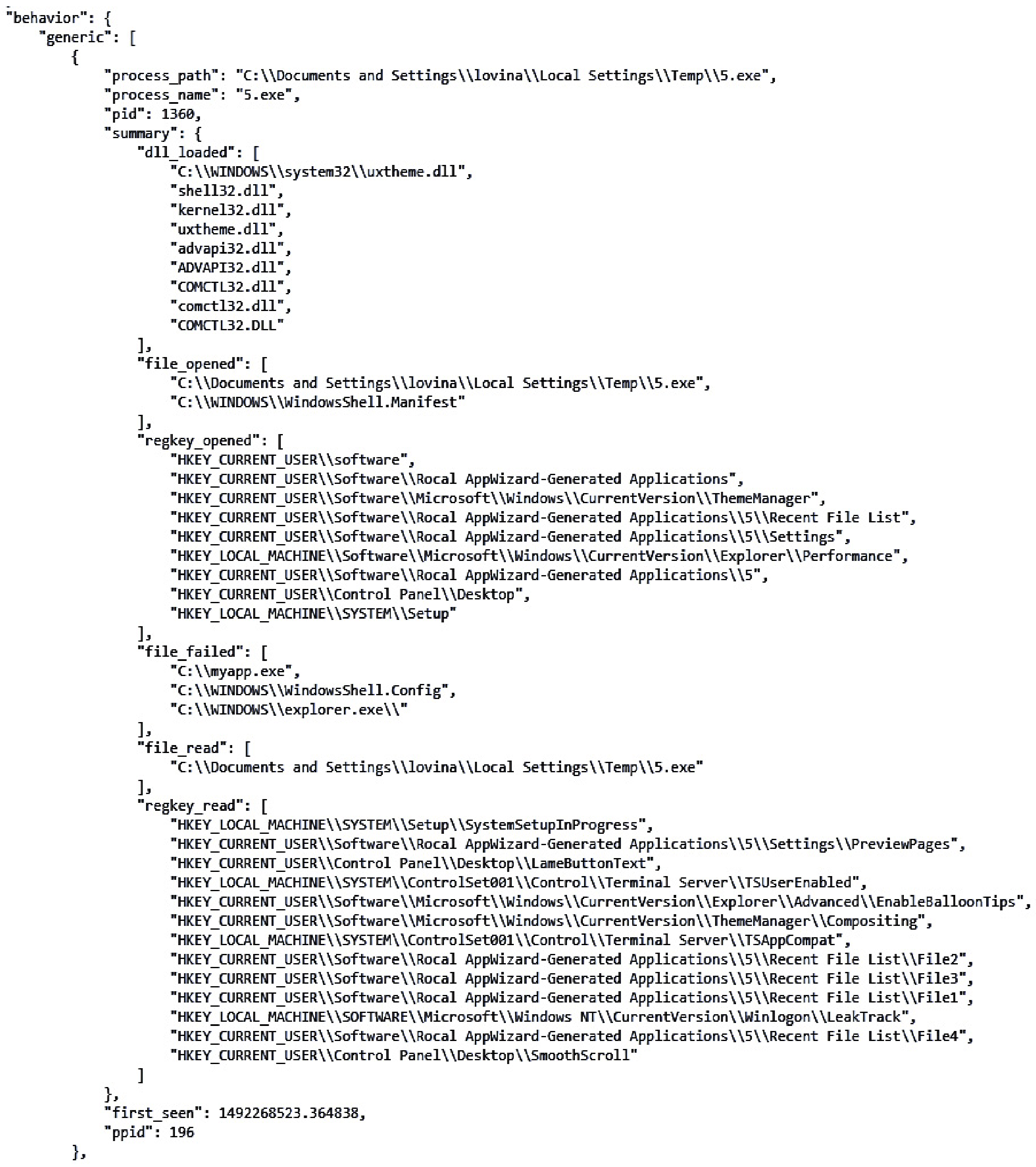
Table 4.1: Number Of Features and their details.

Feature	**Contained Deatils**	**Total Number**
Api Files	the traces of invocation of native functions and windows API calls	246
Registry Key Operations	which registry are read, opened, and written	475
Dll Files Used	which DLL files are opened, read and written	650
Embedded Strings	String used by the ransomware	150
	Total	**1521**

Figure 5.1 shows how the JSON file is parsed to get the labels from all the JSON reports. Since we have a huge number of feature it does not mean that all of them contribute to the classification algorithm. As a result, we need to perform feature selection. Feature selection enables us to reduce the number of features and build simple machine learning algorithms. It also reduces the training time and increases accuracy considerably.

We consider only the presence or absence of that particular feature. Hence, for these binary features, the Mutual Information criterion is a good method to select the most relevant ones, since it allows us to quantify how much discrimination each feature adds to the classifier. Considering that X and Y are two discrete random variables with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$, the Mutual Information $MI(X, Y)$ is defined as the relative entropy between the joint distribution and the product of the marginal distribution :

$$MI(X,Y) = \sum_{x\in X} \sum_{y\in Y} p(x,y) \log \frac{p(x,y)}{p(x).p(y)}$$



```
"behavior": {
    "generic": [
        {
            "process_path": "C:\\Documents and Settings\\lovina\\Local Settings\\Temp\\5.exe",
            "process_name": "5.exe",
            "pid": 1360,
            "summary": {
                "dll_loaded": [
                    "C:\\WINDOWS\\system32\\uxtheme.dll",
                    "shell32.dll",
                    "kernel32.dll",
                    "uxtheme.dll",
                    "advapi32.dll",
                    "ADVAPI32.dll",
                    "COMCTL32.dll",
                    "comctl32.dll",
                    "COMCTL32.DLL"
                ],
                "file_opened": [
                    "C:\\Documents and Settings\\lovina\\Local Settings\\Temp\\5.exe",
                    "C:\\WINDOWS\\WindowsShell.Manifest"
                ],
                "regkey_opened": [
                    "HKEY_CURRENT_USER\\software",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications",
                    "HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\ThemeManager",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications\\5\\Recent File List",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications\\5\\Settings",
                    "HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Performance",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications\\5",
                    "HKEY_CURRENT_USER\\Control Panel\\Desktop",
                    "HKEY_LOCAL_MACHINE\\SYSTEM\\Setup"
                ],
                "file_failed": [
                    "C:\\myapp.exe",
                    "C:\\WINDOWS\\WindowsShell.Config",
                    "C:\\WINDOWS\\explorer.exe\\"
                ],
                "file_read": [
                    "C:\\Documents and Settings\\lovina\\Local Settings\\Temp\\5.exe"
                ],
                "regkey_read": [
                    "HKEY_LOCAL_MACHINE\\SYSTEM\\Setup\\SystemSetupInProgress",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications\\5\\Settings\\PreviewPages",
                    "HKEY_CURRENT_USER\\Control Panel\\Desktop\\LameButtonText",
                    "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Terminal Server\\TSUserEnabled",
                    "HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\Advanced\\EnableBalloonTips",
                    "HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\ThemeManager\\Compositing",
                    "HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Terminal Server\\TSAppCompat",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications\\5\\Recent File List\\File2",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications\\5\\Recent File List\\File3",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications\\5\\Recent File List\\File1",
                    "HKEY_LOCAL_MACHINE\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Winlogon\\LeakTrack",
                    "HKEY_CURRENT_USER\\Software\\Rocal AppWizard-Generated Applications\\5\\Recent File List\\File4",
                    "HKEY_CURRENT_USER\\Control Panel\\Desktop\\SmoothScroll"
                ]
            },
            "first_seen": 1492268523.364838,
            "ppid": 196
        },
```

Figure 4.4: Example of Json report obtained from cuckoo sandbox

Chapter 5

Evaluation

Here we discuss the experimental evaluation to access the accuracy to detect and classify ransomware by measuring the performance and the detection rate of the classifier compared to other machine learning techniques. We used recursive feature elimination to select the most relevant features for all the algorithms.

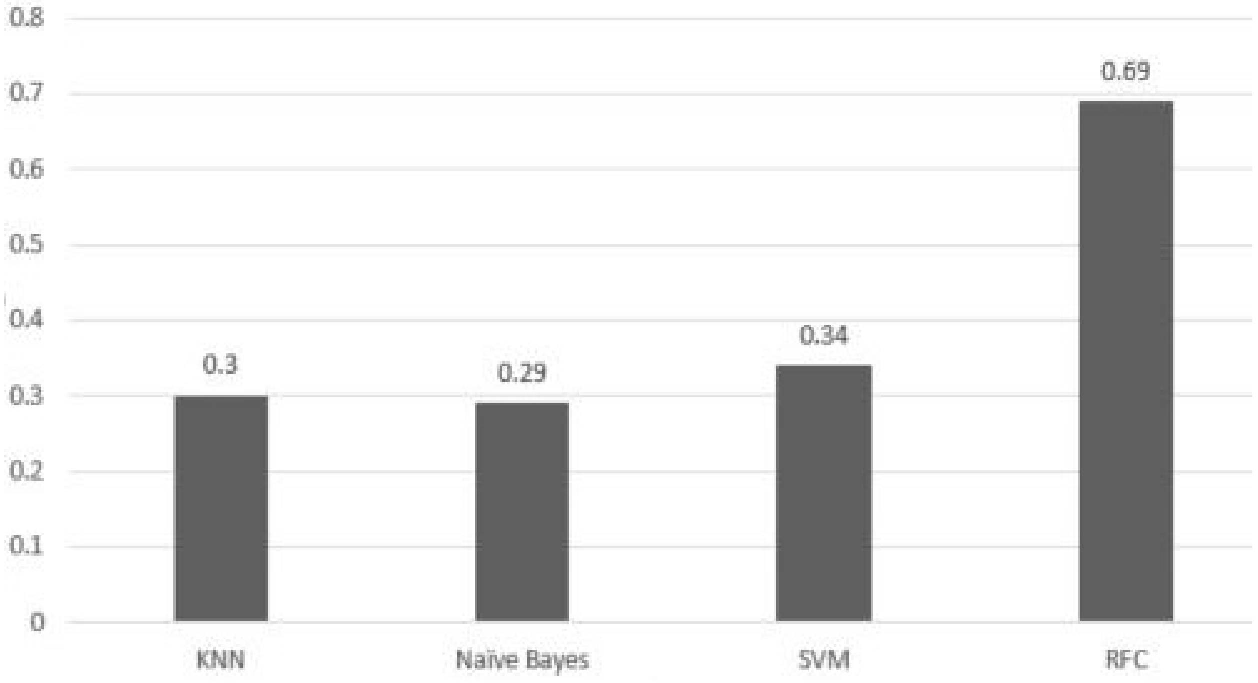


Figure 5.1: f-score of algorithms without applying any feature selection algorithm.

5.1 Comparison with Feature Selection

We evaluated the performance of the 4 machine learning classifiers with the number of features selected with the recursive feature elimination criteria. we explored between 50 and

1500 features to see which combination performs better. For each value explored, we created 100 random splits of the whole dataset with 70% of the samples for training and 30% as test samples.

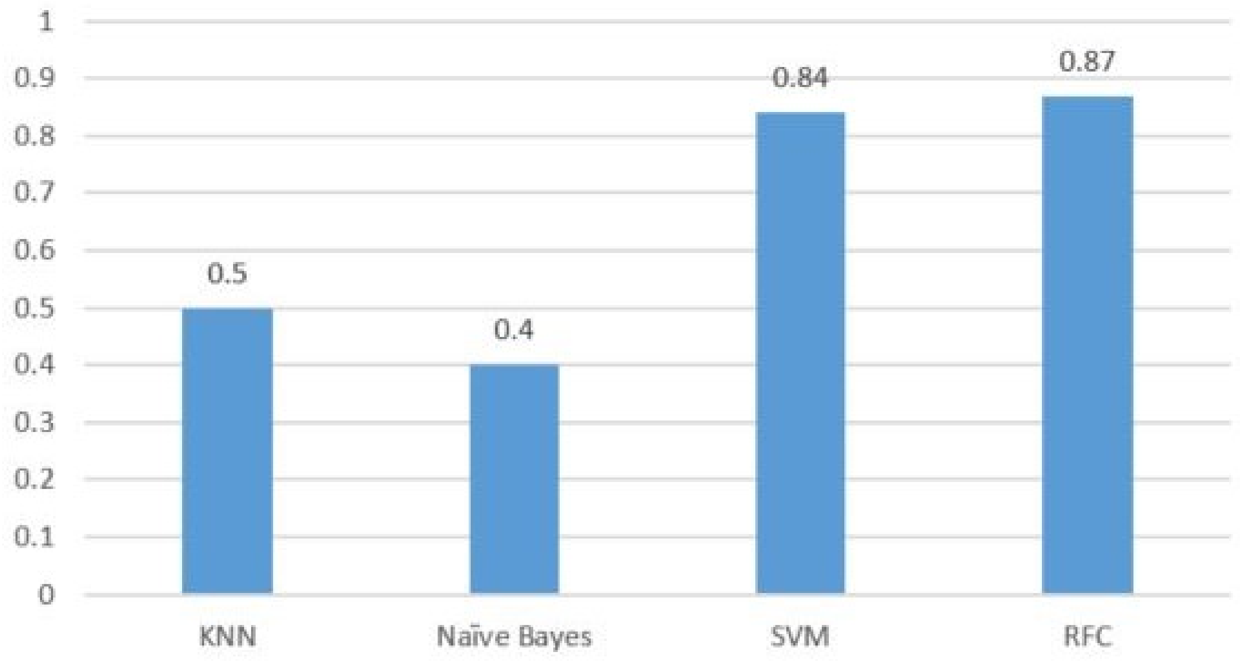


Figure 5.2: f-score after applying feature selection algorithm with top 2000 features.

5.1 shows the f-score of all the algorithms without applying any feature selection algorithm and 5.2 shows the f-score of all the algorithms after applying feature selection with top 2000 features. It can be observed that SVM and RFC are competitive with each other having very good f-score of 0.84 and 0.87 respectively. Also in SVM algorithm we can see a great improvment from 0.34 without feature selection to 0.84 with feature selection.

By looking at 5.2, it is also interesting to observe that the maximum f-score of the algorithms is achieved for top 2000 features. Thus, adding more features to the classifiers does not improve the accuracy. This result shows the importance of feature selection to reduce the complexity of the machine learning algorithms without affecting the performance.

From the table 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7 and 5.8 Accuracy of K nearest neighbor, Naive Bayes, SVM and RFC without feature selection and with feture selection can be better compared. We can see that mulutual information criteria allows us to select important and relevant features which results in accuracy to improve. The accuracy is significantly high for SVM. But overall Random Forest classifier has the highest accuracy.

Class Name	**precision**	**recall**	**f1-score**	**support**
Cbtlocker	0.67	0.25	0.36	8
Crilock	0.57	0.28	0.37	72
Crypto Wall	0.25	0.28	0.26	18
Crypto locker	0.26	0.26	0.26	47
Cryptor	0.25	0.3	0.27	121
Kovter	0.6	0.25	0.35	24
Reveton	0.24	0.35	0.29	66
Trojan-ransom	0.3	0.32	0.31	136
Virlock	0.5	0.2	0.29	5
Weelsof	0.5	0.2	0.29	5
Zusy-ransomware	0.29	0.27	0.28	26
avg/total	0.34	0.3	0.3	528

Table 5.1: Precision, recall and f-score without feature selection of KNN

Class Name	**precision**	**recall**	**f1-score**	**support**
Cbtlocker	0.75	0.38	0.5	8
Crilock	0.78	0.5	0.61	72
Crypto Wall	0.41	0.39	0.4	18
Crypto locker	0.44	0.47	0.45	47
Cryptor	0.43	0.5	0.46	121
Kovter	0.73	0.46	0.56	24
Reveton	0.38	0.48	0.43	66
Trojan-ransom	0.51	0.54	0.52	136
Virlock	0.67	0.4	0.5	5
Weelsof	0.67	0.4	0.5	5
Zusy-ransomware	0.57	0.46	0.51	26
avg/total	0.52	0.49	0.5	528

Table 5.2: Precision, recall and f-score with feature selection of KNN

Class Name	**precision**	**recall**	**f1-score**	**support**
Cbtlocker	0.67	0.25	0.36	8
Crilock	0.58	0.29	0.39	72
Crypto Wall	0.21	0.22	0.22	18
Crypto locker	0.27	0.28	0.27	47
Cryptor	0.24	0.28	0.26	121
Kovter	0.56	0.21	0.3	24
Reveton	0.24	0.33	0.28	66
Trojan-ransom	0.28	0.32	0.3	136
Virlock	0.5	0.2	0.29	5
Weelsof	0.5	0.2	0.29	5
Zusy-ransomware	0.26	0.23	0.24	26
avg/total	0.33	0.29	0.29	528

Table 5.3: Precision, recall and f-score without feature selection of Naive Bayes

Class Name	**precision**	**recall**	**f1-score**	**support**
Cbtlocker	0.67	0.25	0.36	8
Crilock	0.64	0.42	0.5	72
Crypto Wall	0.38	0.33	0.35	18
Crypto locker	0.35	0.38	0.37	47
Cryptor	0.34	0.41	0.38	121
Kovter	0.69	0.38	0.49	24
Reveton	0.28	0.39	0.33	66
Trojan-ransom	0.42	0.43	0.42	136
Virlock	0.5	0.2	0.29	5
Weelsof	0.5	0.2	0.29	5
Zusy-ransomware	0.5	0.35	0.41	26
avg/total	0.43	0.4	0.4	528

Table 5.4: Precision, recall and f-score with feature selection of Naive Bayes

Class Name	**precision**	**recall**	**f1-score**	**support**
Cbtlocker	0.67	0.25	0.36	8
Crilock	0.62	0.33	0.43	72
Crypto Wall	0.21	0.22	0.22	18
Crypto locker	0.29	0.30	0.29	47
Cryptor	0.31	0.35	0.33	121
Kovter	0.60	0.25	0.35	24
Reveton	0.27	0.38	0.32	66
Trojan-ransom	0.34	0.38	0.36	136
Virlock	0.50	0.20	0.29	5
Weelsof	0.50	0.20	0.29	5
Zusy-ransomware	0.29	0.27	0.28	26
avg / total	0.37	0.34	0.34	528

Table 5.5: Precision, recall and f-score without feature selection of SVM

Class Name	**precision**	**recall**	**f1-score**	**support**
Cbtlocker	1.00	0.50	0.67	8
Crilock	0.97	0.86	0.91	72
Crypto Wall	1.00	0.78	0.88	18
Crypto locker	0.75	0.85	0.80	47
Cryptor	0.74	0.87	0.80	121
Kovter	1.00	0.79	0.88	24
Reveton	0.90	0.83	0.87	66
Trojan-ransom	0.82	0.86	0.84	136
Virlock	1.00	0.60	0.75	5
Weelsof	1.00	0.80	0.89	5
Zusy-ransomware	1.00	0.81	0.89	26
avg / total	0.86	0.84	0.84	528

Table 5.6: Precision, recall and f-score with feature selection of SVM

Class Name	**precision**	**recall**	**f1-score**	**support**
Cbtlocker	1.00	0.62	0.77	8
Crilock	0.90	0.65	0.76	72
Crypto Wall	1.00	0.61	0.76	18
Crypto locker	0.54	0.68	0.60	47
Cryptor	0.53	0.71	0.61	121
Kovter	1.00	0.62	0.77	24
Reveton	0.75	0.68	0.71	66
Trojan-ransom	0.69	0.72	0.70	136
Virlock	1.00	0.40	0.57	5
Weelsof	1.00	0.40	0.57	5
Zusy-ransomware	1.00	0.65	0.79	26
avg / total	0.73	0.68	0.69	528

Table 5.7: Precision, recall and f-score without feature selection of RFC

Class Name	**precision**	**recall**	**f1-score**	**support**
Cbtlocker	1.00	0.62	0.77	8
Crilock	0.95	0.88	0.91	72
Crypto Wall	1.00	0.83	0.91	18
Crypto locker	0.82	0.85	0.83	47
Cryptor	0.79	0.88	0.83	121
Kovter	1.00	0.83	0.91	24
Reveton	0.87	0.88	0.87	66
Trojan-ransom	0.85	0.89	0.87	136
Virlock	1.00	0.80	0.89	5
Weelsof	1.00	0.80	0.89	5
Zusy-ransomware	1.00	0.85	0.92	26
avg / total	0.87	0.87	0.87	528

Table 5.8: Precision, recall and f-score with feature selection of RFC

Chapter 6

Limitations and Future Work

In our experiments, we have shown that even if the initial set of features was rather large (around 31, 000), in the end only 2000 have been shown to be sufficient. One important result that we were able to provide is that recursive feature elimination is very effective in deriving the most important features. In particular, the registry key and API calls are the two classes with most relevant features.

A first limitation concerns the analysis, and detection, of those samples of ransomware that are silent for some time, or that wait for the user to do something. To mitigate this ransomwares behavior, a solution is to inject some real, or script-generated, user actions into the sandboxed environment. Finally, note that the dataset consists of 1584 working samples of ransomware belonging to 11 different classes. We analyzed only those ransomware which could run properly.

Dynamic Solution should take into account and reduce the time of feature analysis to be as short as possible. Also should be able to distinguish between actions performed by legitimate encryption software and a crypto-ransomware. In any case, we believe that dynamic analysis is needed to improve the classification of ransomware.

REFERENCES

[1] Andronio, Nicol, Stefano Zanero, and Federico Maggi. "Heldroid: Dissecting and detecting mobile ransomware." International Workshop on Recent Advances in Intrusion Detection. Springer International Publishing, 2015.

[2] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic analysis of malware behavior using machine learning. J. Comput. Secur., 19(4):639668, Dec. 2011.

[3] P. T. Nolen Scaife, Henry Carter and K. R. Butler. Cryptolock (and drop it): Stopping ransomware attacks on user data. In 2016 IEEE 36th International Conference on Distributed Computing Systems, pages 303312, 2016

[4] Dash, Santanu Kumar, et al. "Droidscribe: Classifying android malware based on runtime behavior." Security and Privacy Workshops (SPW), 2016 IEEE. IEEE, 2016.

[5] Sgandurra, Daniele, et al. "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection." arXiv preprint arXiv:1609.03020 (2016).

[6] J. Z. Kolter and M. A. Maloof. Learning to detect and classify malicious executables in the wild. The Journal of Machine Learning Research, 7:27212744, 2006.

[7] A. Kharraz, W. K. Robertson, D. Balzarotti, L. Bilge, and E. Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In DIMVA - 12th International Conference, 2015, Milan, Italy, July 9-10, 2015, Proceedings, pages 324, 2015

[8] K. Savage, P. Coogan, and H. Lau. The evolution of ransomware. http://www.symantec.com/content/en/us/enterprise/media/securityresponse /whitepapers/the-evolution-of-ransomware.pdf,2015

[9] Talos Group. Teslacrypt - decrypt it yourself. http://blogs.cisco.com/security/talos/teslacrypt,2015.

[10] Kenichi Terashita and Roland Dela Paz. Cerber Ransomware Marks Its Presence in the Wild, Catches up with CryptoWall and Locky. https://blog.fortinet.com/2016/05/26/cerber-11ransomware-marks-its-presence-in-the-wildcatches-up-with-cryptowall-and-locky,2016.

[11] C. A. Jeff White and M. Yates. Ransomware: Locky, teslacrypt, other malware families use new tool to evade detection. http://researchcenter.paloaltonetworks.com/2016/04/unit42-ransomware-locky-teslacrypt-other-malwarefamilies-use-new-tool-to-evade-detection,2016.

[12] C. M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.

[13] Cyber Threat Alliance. Lucrative ransomware attacks: Analysis of the cryptowall version 3 threat. http://cyberthreatalliance.org/cryptowallreport.pdf,2015.

[14] K. Dandurant. Cryptowall attacks durham police files. http://www.seacoastonline.com/article/20140607/NEWS/406070322, 2014.

[15] A. Gazet. Comparative analysis of various ransomware virii. Journal in Computer Virology, 6(1):7790, 2010.

[16] C. Cortes and V. Vapnik, Support-vector networks, Machine Learning, vol. 20, 1995

[17] Bhardwaj, Akashdeep, et al. "Ransomware digital extortion: a rising new age threat." Indian Journal of Science and Technology 9 (2016): 14.

[18] Zolkipli, Mohamad Fadli, and Aman Jantan. "Malware behavior analysis: Learning and understanding current malware threats." Network Applications Protocols and Services (NETAPPS), 2010 Second International Conference on. IEEE, 2010.

[19] Mujumdar, Ashwini, Gayatri Masiwal, and Dr BB Meshram. "Analysis of signature-based and behavior-based anti-malware approaches." signature 2.6 (2013).

[20] Baldangombo, Usukhbayar, Nyamjav Jambaljav, and Shi-Jinn Horng. "A static malware detection system using data mining methods." arXiv preprint arXiv:1308.2831 (2013).

[21] Alazab, Mamoun, et al. "Zero-day malware detection based on supervised learning algorithms of API call signatures." Proceedings of the Ninth Australasian Data Mining Conference-Volume 121. Australian Computer Society, Inc., 2011.

[22] Kateryna Chumachenko, "Machine Learning Methods For Malware Detection and Classification"

[23] Cuckoo Foundation. 2015. Cuckoo Sandbox Book. www document. Available at: http://docs.cuckoosandbox.org/en/latest/introduction/what/. [Accessed 15 February 2017]

[24] Juniper Research. 2016. Cybercrime will cost businesses over $2 trillion by 2019. WWW document. Available at: https://www.juniperresearch.com/press/press-releases/cybercrimecost-businesses-over-2trillion.[Accessed on 15 February 2017]

[25] Swain, Philip H., and Hans Hauska. 1977. The Decision Tree classifier: design and potential. IEEE Transactions on Geoscience Electronics.

[26] Symantec Security Response. 2016. Locky ransomware on aggressive hunt for victims. www document. Available at: https://www.symantec.com/connect/blogs/locky-ransomware-aggressive-huntvictims. [Accessed 15 February 2017]

[27] Thirumuruganathan, Saravanan. 2010. A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm. www document. Available at: https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailedintroduction-to-k-nearest-neighbor-knn-algorithm/. [Accessed 15 February 2017]

[28] Villeneuve, Nart, and James T. Bennett. 2014. XtremeRAT: Nuisance or Threat? www document. Available at: https://www.fireeye.com/blog/threatresearch/2014/02/xtremerat-nuisance-or-threat.html. [Accessed 15 February 2017]

[29] VirusTotal. 2017. VirusTotal. www page. Available at: https://virustotal.com/. [Accessed 15 February 2017]

[30] Mimoso, Michael. 2016. Master Decryption Key Released for TeslaCryptRansomware. www document. Available at: https://threatpost.com/masterdecryption-key-released-for-teslacrypt-ransomware/118179/.[Accessed 15 February 2017]

[31] Mitchell, Tom. 1997. Machine Learning. McGrawHillScience/Engineering/Math.

[32] Moffie, Micha, Winnie Cheng, David Kaeli, and Qin Zhao. 2006. Hunting Trojan Horses. Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability

[33] OBrien, Dick. 2016. Dridex: Tidal waves of spam pushing dangerous financial Trojan. Symantec Corporation.

[34] Pirscoveanu, Radu-Stefan. 2015. Clustering Analysis of Malware Behavior. Aalborg University .

[35] Prasad, B. Jaya, Haritha Annangi, and Krishna Sastry Pendyala. 2016. Basic static malware analysis using open source tools.

[36] Reddy, Krishna Sandeep, and Arun Pujari. 2006. N-gram analysis for computer virus detection.

[37] Rieck, Konrad, Philipp Trinius, Carsten Willems, and Thorsten Holz. 2011. Automatic Analysis of Malware Behavior using Machine Learning. Journal of Computer Security.

[38] Savage, Kevin, Peter Coogan, and Hon Lau. 2015. The Evolution of Ransomware. Symantec Corporation.

[39] Schneider, Jeff. 1997. Cross Validation. Carnegie Mellon University.

[40] Singhal, Priyank, and Nataasha Raul. 2015. Malware Detection Module using Machine Learning Algorithms to Assist in Centralized Security in Enterprise Networks.