A tool to detect and prevent malware attacks: A Survey

Satish N. Chalurkar[1]
[1]Computer Department, VJTI
Matunga(E),Mumbai,India

Nilesh Khochare[2]
[2]Computer Department, VJTI
Matunga(E),Mumbai,India

Dr. B.B.Meshram[3]
[3]Head of Computer Department
VJTI,Mumbai,India

***Abstract*- Short for "malicious software," malware refers to software programs designed to damage or do other unwanted actions on a computer system. "mal" is a prefix that means "bad," making the term "badware," which is a good way to remember it.**

Common examples include viruses, worms, Trojan horses, and spyware. Viruses, for example, can cause havoc on a computer's hard drive by deleting files or directory information. Spyware can gather data from a user's system without the user knowing it. This can include anything from the Web pages a user visits to personal information, such as credit card numbers.

Here, we have given the comparism between the various detection algorithms for detecting the malware attacks. using the API Hooking we can prevent the malware attacks.

***Keywords* - Attackers, Malicious code, Classification, detection, Malware prevention**

I. INTRODUCTION

The widespread use of the Internet caused the number of warnings being made about the dark side of our technological revolution to increase and we are becoming uniquely vulnerable to many mysterious and malicious threats. Malicious software has been and still used to spread mayhem, enact political revenge on a corporate target, steal data, increase access to a network resource, hijack networks, deny companies use of their network, or sometimes simply gain bragging rights.

Collectively known as Malware (Malicious Software) these entities usually get into a computer via some type of internet activity such as downloading software, file-sharing, visiting certain websites and even activities as simple as opening an email especially if it is a joke type email.

To be able to accurately identify, classify, and provide better countermeasures against the many types of malicious threats, one must first understand how the authors of malware think, what motivates them, and what are the tools and techniques they use.

II. RELATED WORK

The first line of defense against malware is to know who are the people responsible for writing or releasing malicious code and what motivates them.

A. Attackers

1) Script Kiddies

This group of people call themselves real hackers but in fact some are novice programmers and others are not even programmers at all. A script kiddie searches the Internet for exploits (malicious code) written by highly skilled programmers and use these exploits to break into systems. Many of the attacks these days are caused by script kiddies who release malicious code and wreak havoc.

2) Crackers

This group of people is known as average attackers who have medium level programming skills. Crackers usually use the same tools used by script kiddies. However, crackers do know how the code of such tools is written and how it behaves. Crackers do not have any ethical boundaries and their motivations behind mounting attacks against systems and networks are money and power.

3) White Hat Hackers

This group of people is highly skilled programmers who use their skills in enhancing the security of software. White hat hackers work in the computer security field and they are indeed known to be the best developers.

4) Black Hat Hackers

The technical skills of black hat hackers are at a level comparable to that of white hat hackers. However, the focus is different. White hat hackers are more concerned about the security issues of software development and algorithms. On the other hand, Black hat hackers are more concerned about the security of systems.

B. Instant Messaging: A New Target for Hackers

Instant messaging is exploding as a means of personal and corporate communications. Individuals chat via IM;

companies rely on beefedup versions of the technology, with its real-time capabilities, for collaborative design work; and e-businesses use IM to provide live, immediate customer service to shopper's .Meanwhile, the technology is finding its way onto mobile devices, including PDAs and smart phones.

However, as IM becomes more popular, particularly for businesses, it has also increasingly become the target of attacks, such as those using malicious code and phishing.

However, e-mail has been a more attractive target than IM for many years. Popular public IM systems such as AIM and Yahoo Messenger are closed and thus don't generally connect to other systems. This limits IM's ability to spread attacks. Also, until recently, IM clients have been simple prosystems with few published vulnerabilities to exploit.

C. Types of Malicious Code

Malicious code comes in a wide variety of forms and is distributed through an ever-growing number of delivery mechanisms. In general, malicious code is any software that impedes the normal operation of a computer or networking device. This software most often executes without the user's consent.

1) Storm

The malware is generally called the Storm worm, although it is also known by names such as Nuwar, Peacomm, Tibs, and Zhelatin. When it launched a series of attacks that generated an estimated 20 times the normal spam volume. The Storm worm has created a massive network of remotely operated zombie computers that the person or people who control the malware have used to launch spam and distributed denial-of-service (DDoS) attacks.

2) E-mail Viruses and Miscellaneous Viruses

A virus is a computer program that initiates an action on a computer without the user's consent.

In general, computer viruses replicate and spread from one system to another. Many viruses merely replicate and clog e-mail systems. Some computer viruses have what is called a *malicious payload*, which is code that can execute commands on computers such as deleting or corrupting files or disabling computer security software. In addition, some computer viruses can attach themselves to another block of code to facilitate propagation. Viruses generally have the following components:

- A ***replication mechanism*** that allows reproduction and enables the virus to move from one computer to other computers
- A ***trigger*** that is designed to execute the replication mechanism or the task of the virus
- A ***task*** **or** ***group of tasks*** that execute on a computer to destroy or alter files, change computer settings or configurations, or otherwise hinder or impede the operations of a computer or networking device.

3) Trojans and other backdoors

A Trojan horse is a malicious program which is embedded inside an innocent program (i.e. a game). Such programs are based on a client/server model where the server program gets installed on the victim's machine and the attacker uses the client program to send commands to the server, hence giving the attacker complete control over the victim's machine.

Examples of what Trojans allow remote users controlling the Trojan to do include the following

- Remove files from the infected computer.
- Download files to the infected computer.
- Make registry changes to the infected computer.
- Delete files on the infected computer.
- Steal passwords and other confidential information.
- Log keystrokes of the computer user.
- Rename files on the infected computer.
- Disable a keyboard, mouse, or other peripherals.
- Shut down or reboot the infected computer.
- Run selected applications or terminate open applications.
- Disable virus protection or other computer security software.

4) Time bombs

One of the first forms of malicious code was a time bomb (or logic bomb), which, when installed, is a dormant code that can be triggered at a future date by a specific event or circumstance. Triggers can be a specific date and time or even a cumulative number of system starts. Many disgruntled programmers have planted time bombs to retaliate against employers. Time bombs have also been installed in extortion attempts.

5) Spyware

The term *spyware* is used to describe any computer technology that gathers information about a person or organization without their knowledge or consent. Spyware can be installed on a computer through several covert means, including as part of a software virus or as the result of adding a new program.

Note that the terms spyware, stealware, and adware are sometimes used to describe the same or similar types of malicious code.

Spyware is used to gather information such as recorded keystrokes (passwords), a list of Web sites visited by the user, or applications and operating systems that are installed on the computer. Spyware can also collect names, credit card numbers, and other personal information. It is usually placed on a computer to gather information about a user that is later sold to advertisers an other interested parties.

6) Rootkit

A rootkit is a type of malicious software that is activated each time your system boots up. Rootkits are difficult to detect because they are activated before your system's Operating System has completely booted up. A rootkit often allows the installation of hidden files, processes, hidden user accounts, and more in the systems OS. Rootkits are able to intercept data from terminals, network connections, and the keyboard.

A rootkit is a collection of tools (programs) that enable administrator-level access to a computer or computer network. Typically, a cracker installs a rootkit on a computer after first obtaining user-level access, either by exploiting a known vulnerability or cracking a password. Once the rootkit is installed, it allows the attacker to mask intrusion and gain root or privileged access to the computer and, possibly, other machines on the network.

A rootkit may consist of spyware and other programs that: monitor traffic and keystrokes; create a "backdoor" into the system for the hacker's use; alter log files; attack other machines on the network; and alter existing system tools to escape detection.

D. Malware Classification

Metamorphic and polymorphic malware are two categories of malicious software programs (malware) that have the ability to change their code as they propagate.

1) *Metamorphic malware*

Metamorphic malware is rewritten with each iteration so that each succeeding version of the code is different from the preceding one. The code changes make it difficult for signature-based antivirus software programs to recognize that different iterations are the same malicious program.

In spite of the permanent changes to code, each iteration of metamorphic malware functions the same way. The longer the malware stays in a computer, the more iterations it produces and the more sophisticated the iterations are, making it increasingly hard for antivirus applications to detect, quarantine and disinfect.

2) *Polymorphic malware*

Polymorphic malware also makes changes to code to avoid detection. It has two parts, but one part remains the same with each iteration, which makes the malware a little easier to identify. For example, a polymorphic virus might have a virus decryption routine (VDR) and an encrypted virus program body (EVB). When an infected application launches, the VDR decrypts the encrypted virus body back to its original form so the virus can perform its intended function. Once executed, the virus is re-encrypted and added to another vulnerable host application. Because the virus body is not altered, it provides a kind of complex signature that can be detected by sophisticated antivirus programs.

III.DISCUSSION

A. Malware Detection Schemes

Commercial products incorporate a wide range of different scanning and classification techniques like check summing, string scanning, smart scanning, X-Ray scanning, code emulation, geometric detection, and heuristics. Since the complexity of these approaches increases and commercial products try to achieve real-time scanning, the use of filters is common. Examples of filters are the exclusion of files that don't have an ".exe" suffix or lack executable headers, like PE or ELF. Other filters are to scan only in the boot-sector for boot-sector-viruses.

1) Top-and-Tail scanning

Means to speed up detection in AV include ***Top-and-Tail scanning***, which only scans certain regions at the start and end of each file, as well as *Entry-Point scanning* that is performed around the entry point of a file. It is obvious that the types of filters have to be tailored specifically to malware families.

2) Check-summing

Check-summing is the fastest but least reliable technique. The file that is to be scanned is check-summed using cryptographic hash functions, like MD5, and compared to a list of malicious programs as well as possible whitelists. A variant of this approach is to hash only parts of the file.

3) String scanning

A little more sophisticated is ***string scanning***, which scans the considered files for common substrings that only found in specific malware. Both approaches are not able to detect strong metamorphic because of the high variability of the infections.

4) Smart scanning

Smart scanning is a special form of wildcard scanning that leaves out irrelevant parts of the inspected file, like obvious junk code. It is able to detect malware that performs metamorphism through the insertion of junk code but fails for semantic code replacement or code reordering. Since real malcode is often encrypted, commercial products often target the encryption of the malware's body and then apply one of the scanning methods described above.

5) X-Ray scanning

X-Ray scanning targets the encryption often used by attempting simple, standard decryption based on simple arithmetic operations.

6) Code emulation

Code emulation executes the start of the malcode in a small virtual machine trying to find the end of the decryption routine included.

B. Signature based Algorithm

Present malware detection software uses signature based and heuristic-based algorithms. Signature-based algorithms creates a unique short strings of bytes for a given virus, later which are used to identify particular viruses in executable files, or memory with few errorrate. Generally speaking, detection strategies based on string signatures uses a database of regular expressions and a string matching engine to scan files and to detect infected ones. Each regular expression of the database is designed to identify a known malicious program.

There are at least three difficulties tied to this approach.

First, the identification of a malware signature requires a human expert and the time to forge a reliable signature is long compared to the time related to a malware attack.

Second, string signature approach can be easily bypassed by obfuscation methods.

Third, as the quantity of malware increase, the ratio of false positives becomes a crucial issue. And removing old malware signatures would open doors for outbreaks of engineered malware.

Signature-based detection is slow and not effective against polymorphic and unknown malwares. However heuristic-based algorithms detects unknown malwares, but they are usually inefficient and inaccurate. The inability of these traditional approaches to detect new form of malicious executables has shifted the focus of virus research to find more generalized and scalable features that can identify malicious behaviour as a process instead of a unique signature.

Therefore, because of so much drawback of signature based algorithm, we have refers the new techniques known as "Adapting Post Processing Techniques of associative Classification for malware detection"

C. Adapting Post Processing Techniques of associative Classification for malware detection

In order to overcome the disadvantages of the widely used signature-based malware detection method, data mining and machine-learning approaches are proposed for malware detection. Naive Bayes method, SVM, and decision tree classifiers are used to detect new malicious executables in previous studies. Associative classification, as a new classification approach integrating association rule mining and classification, becomes one of the significant tools for knowledge discovery and data mining. Due to the fact that frequent itemsets (sets of API calls) discovered by association mining can well represent the underlying semantics (profiles) of malware and benign file datasets, associative classification has been successfully used in the IMDS system developed in and for malware detection. However, there is often a huge number of rules generated in a classification association rule mining practice.

It is often infeasible to build a classifier using all of the generated rules. Hence, how to reduce the number of the rules and select the effective ones for prediction is very important for improving the classifier's ACY and efficiency. Recently, many postprocessing techniques, including rule pruning, rule ranking, and rule selection have been developed for associative classification to reduce the size of the classifier and make the classification process more effective and accurate. It is interesting to know how these post processing techniques would help the associative classifiers for malware detection.

In this paper, we systematically evaluate the effects of the post processing techniques in malware detection and propose an effective way, i.e., CIDCPF, to detect the malware from the "gray list."

A. Rule Pruning

In order to reduce the size of the classifier and make the classification process more effective and accurate, the removal of the redundant or misleading rules is indispensable.

There are five popular rule pruning approaches which mainly focus on preventing these redundant or misleading rules from taking any part in the prediction process of test data objects.

B. $\chi 2$ (chi-square) test

The test is always carried out on each generated rule to find out whether the rule's antecedent is positively correlated with the rule's consequent. It is adopted by classification based on multiple association rules (CMAR) algorithm in its rule discovery step.

1) Redundant rule pruning

This rule pruning method discards specific rules with fewer confidence values than general rules. Several algorithms, such as CMAR and adopt this approach for rule pruning.

2) Database coverage

This pruning approach tests the generated rules against the training dataset, and only keeps the rules, which cover at least one training data object not considered by a higher ranked rule for later classification. This method is created by the classification based on associations (CBA) and used by CMAR and multiclass classification based on association rule (MCAR).

3) Pessimistic error estimation

The method works by comparing the estimated error of a new rule. If the expected error of the new rule is lower than that of the original rule, then the original rule will be replaced by the new rule. CBA and have used it to effectively reduce the number of the generated rules.

4) Lazy pruning

This method discards the rules, which incorrectly classify training objects and keeps all others. It has been used in for rule pruning.

C. Rule Ranking

Rule ranking plays an important role in the classification process, since most of the associative classification algorithms, such as CBA, CMAR, multiclass, multilabel associative classification (MMAC), and MCAR, utilize rule ranking procedures as the basis for selecting the Classifier. Particularly, CBA and CMAR use database coverage pruning approach to build the classifiers, where the pruning evaluates rules according to the rule ranking list. Hence, the highest order rules are tested in advance, and then, inserted into the classifier for predicting test data objects. For rule ranking, there are five popular ranking mechanisms:

a) confidence support size of antecedent (CSA);
b) size of antecedent confidence support (ACS);
c) weighted relative accuracy (WRA);
d) Laplace accuracy; and
e) $\chi 2$ (chi-square) measure.

CSA and ACS are belonging to the pure "support-confidence" framework and have been used by CBA and CMAR for rule ranking. WRA, Laplace accuracy, and $\chi 2$ measure are used by some associative classification algorithms, such as classification based on predictive association rules (CPAR), to weigh the significance of each generated rule.

D. Rule Selection

After pruning and reordering the generated rules, we can select the subset of the rules from the classifier to predict new file samples. There are three common rule selection approaches :

1) Best first rule

This approach selects the first best rule that satisfies the given data object according to the rule list based on certain rule ranking mechanism to predict the new data object. It is used in CBA for predicting test data objects.

2) All rules

This method collects all rules in the classifier that satisfy the new data object, and then, evaluate this collection to identify its class. CMAR uses weighted $\chi 2$ (WCS) testing to predict the class of the new data object.

3) Best k rules

Some associative classification algorithms, like CPAR, select the first best *k* rules that satisfy the new data object, and then, make predictions using certain averaging process.

E. System Architecture

The system architecture of our malware detection is summarized in Fig.1. Basically, the system first uses the feature extractor to extract the API calls from the collected portable executable (PE) files, converts them to a group of 32-bit global IDs as the features of the training data, and stores these features in the signature database. After data transformation, it then

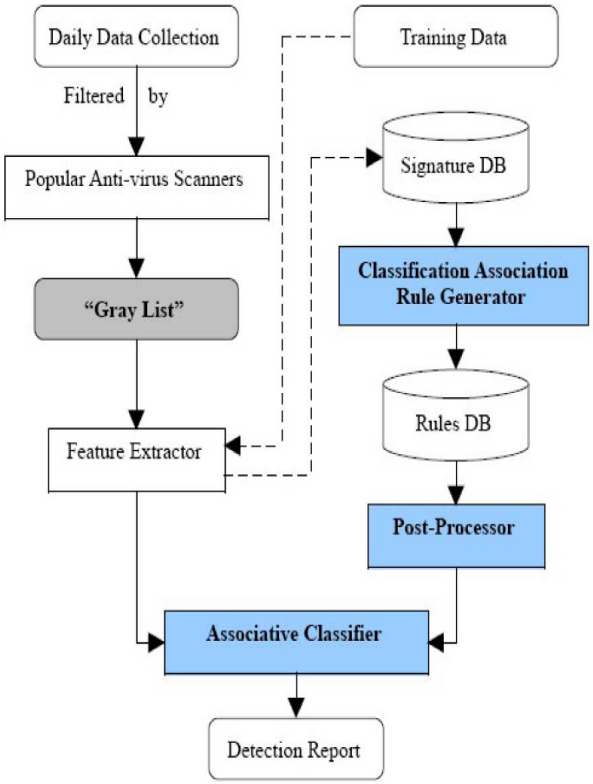

Fig.1 Flow of malware detection

Generates the classification association rules from the training signature database. In the third step, it adapts hybrid post processing techniques of associative classification, including rule pruning, rule ranking, and rule selection to reduce the generated rules. Finally, it builds the classifier using the rules filtered by the postprocessor to detect malware from the "gray list."

F. Classification Association rule generation

Associative classification, as a new classification approach integrating association rule mining and classification, becomes one of the significant tools for knowledge discovery and data mining. It can be effectively used in malware detection, since frequent itemsets are typically of statistical significance and classifiers based on frequent pattern analysis are generally effective to test datasets.

1) Data Collection and Transformation

We obtain 50 000 Windows PE files of which 15 000 are recognized as benign executables and the remaining 35 000 are malicious executables. PE is designed as a common file format for all flavours of Windows operating system, and PE malicious executables are in the majority of the malware rising in recent years.

Based on the system architecture of our previous malware detection system IMDS, we extract the API calls as the features of the file samples and store them in the signature database. There are six fields in the signature database, which are record ID, PE file name, file type ("0" represents Benign file, while "1" is for malicious file), called APIs name, and called API ID and the total number of called API functions. The transaction data can also be easily converted to relational data if necessary. Now the data is ready for classification association rule generation.

2) Classification Association Rule Generation

For malware detection, the first goal is to find out how a set of API calls supports the specific class objectives: class1 = malicious, and class2 = benign.

1) Support and confidence: Given a dataset DB, let $I = \{I1, \ldots, Im\}$ be an item set and $I \rightarrow \text{class}(os, oc)$ be an association rule whose consequent is a class objective. The support and confidence of the rule are defined as follows:

$$os = \text{supp}(I, \text{class}) = \text{count}(I \cup \{\text{class}\})/|\text{DB}| \times 100\%$$

$$oc = \text{conf}(I, \text{class}) = \text{count}(I \cup \{\text{class}\})/\text{count}(I, \text{DB}) \times 100\%$$

Where the function count($I \cup \{$class$\}$) returns the number of records in the dataset DB where $I \cup \{$class$\}$ holds.

2) Frequent itemset: Given *mos* as a user-specified minimum support. *I* is a frequent itemset/pattern in DB if $os \geq mos$.

3) Classification association rule: Given *moc* as a user specified confidence. Let $I = \{I1, \ldots, Im\}$ be a frequent itemset. $I \rightarrow \text{class}\ (os, oc)$ is a classification association rule if $oc \geq moc$.

Apriori and FP-Growth algorithms can be extended to associative classification. For rule generation, we use theOOA_Fast_FP-Growth algorithm proposed in to derive the complete set of the rules with certain support and confidence thresholds, since it is much faster than Apriori

for mining frequent itemsets. The number of the rules is also correlated to the number of the file samples.

With the *os* and *oc* values, we know that this set of API calls appears in 9100 malware samples and not in any benign files. Rules like *R1* with both high support and confidence can be used for Determining a new file sample is malicious or not. For instance, if a new file sample includes the following API calls: ADVAPI32. DLL, Createservicea; USER32.DLL,Callnexthookex; KERNEL32.DLL,Openprocess; KERNEL32.DLL, Createtoolhelp32snapshot; and KERNEL32.DLL, Process32first; then it can be predicted as malicious.

In fact, many classification methods (such as associative classifiers) have been developed to construct classifiers based on association rules. It based on the complete set of the rules generated by the malware detection rule generator, IMDS applied the technique of CBACB to build a classifier as the malware detection module to predict new file samples [35]. Though it achieved good performance, it is interesting to know how post processing techniques, including rule pruning, rule ranking, and rule selection, would help the associative classifiers in the IMDS system for malware detection. Note that the associative classifier is a supervised method that generates the rules to capture the characteristics of file samples based on known malicious and benign samples.

IV.CONCLUSIONS

In this Paper, We have given the major differences between the metamorphic and polymorphic malware. Malware attack can destroy the system and steal the system information. Various types of malware attacks such as Trojans, backdoors, virus, Stealware, Adware, and root kit are given. We can easily identified that the malware are comes in the system by their respective system.Signiture based techniques are given to detect the malware attacks.Signiture based techniques has some limitation. To avoid some limitation, a Adapting Post Processing Techniques of associative Classification for malware detection is given. Using the API HOOKING we can check the system behaviour and then given some preventation techniques to avoid the malware attacks.

REFERENCES

[1] Yanfang Ye, Tao Li, Qingshan Jiang, and Youyu Wang" CIMDS: *Adapting Postprocessing Techniques of Associative Classification for Malware Detection"*

[2] Carlos Raniery P. dos Santos∗, Rafael Santos Bezerra∗, Joao Marcelo Ceron∗," *Botnet Master Detection Using a Mashup-based Approach"*

[3] Zongqu Zhao" *A Virus Detection Scheme Based on Features of Control Flow Graph"*

[4] Mohamad Fadli Zolkipli Aman Jantan *"An Approach for Malware Behavior Identification and Classification"*

[5] M. Shankarapani, K. Kancherla, S. Ramammoorthy, R. Movva, and S. Mukkamala *"Kernel Machines for Malware Classification and Similarity Analysis"*

[6] Felix Leder, Bastian Steinbock, Peter Martini *"Classification and Detection of Metamorphic Malware using Value Set Analysis"*

[7] Desmond Lobo, Paul Watters and Xinwen Wu "RBACS: *Rootkit Behavioral Analysis and Classification System"*

[8] Siddiqui M.A.:" *Data Mining Methods for Malware Detection."*

[9] Moskovitch R, Feher, Tzachar N, Berger E, Gitelman M, Dolev S,et al." *Unknown malcode detection using OPCODE representation. "*

[10] Michael Erbschloe *"A Computer Security Professional's Guide to Malicious Code "*

[11] Neal Leavitt *"Instant Messaging: A New Target for Hackers"*

[12] George Lawton *"Virus Wars: Fewer Attacks, New Threats"*

[13] Brad Smith *"A Storm (Worm) Is Brewing"*

[14] Didier Stevens *"Malicious PDF Documents Explained"*

[15] Leo Breiman. *"Random Forests." Machine Learning*