Malware Analysis

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering

in

Information Security

Submitted By

Chahak

(801533004)

Under the supervision of:

Dr. Anil Kumar Verma

Associate Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004, PUNJAB, INDIA

JUNE 2017

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Malware Analysis*", in partial fulfillment of the requirements for the award of degree of Master of Engineering in Information Security submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Anil Kumar Verma and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

CHAHAK

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Sanjay Madan)
Department of Cyber Security
CDAC, Mohali

(Dr. Anil Kumar Verma)
Associate Professor
Computer Science and Engineering
Thapar University

Acknowledgement

Words can't express my gratitude towards my guide, Dr. Anil Kumar Verma, Associate Professor, Computer Science and Engineering Department, Thapar University, who has seen me through the entire tenure of the work presented in this thesis. I have been fortunate to have an advisor who gave me the freedom to explore the ocean of research on my own and at the same time guided me throughout the thesis. He is an inspiration in the truest sense of the world.

I am also thankful to Dr. Maninder Singh, Head of Department, CSED and Ms. Jhilik Bhattacharya and Ms. Sreelekha Pandey, P.G. Coordinators, for their constant supervision of the thesis work.

I would also like to thanks my classmates who were always there to offer me, the help and facilities required for the successful completion of my thesis.

Most importantly, I would like to thanks my parents, and the Almighty for always steering towards the right direction out of the blue, to help me to stay calm in the oddest of the times and keep moving even at times when there was no hope.

Chahak

801533004

ME-IS

(2015-2017)

Abstract

Malwares are a trending menace in today's cyber world. They are installed surreptitiously in the system and the results are alarmingly dangerous. Many static analysis approaches and anti-virus tools can be bypassed by the malwares. By analyzing the exact behavior, tendency and execution of the code, dynamic malware analyses have somehow overcome these chicaneries. Analyzing the difference between the desired nodes as well as observing the runtime behavior of malware differentiates dynamic behavior from static. An appropriate tool studies the malware in lieu of its behavior, function and execution and is able to handle multiple processes.

Objectifying the scope and functionality of a malware sample is the motive of malware analysis. Unfortunately the amount of specimens to be analyzed by the vendors is rapidly growing on a daily basis. Analyzing the sample during execution time is known as Dynamic Analysis whereas Static analysis is done by inspecting the program and Memory Analysis is defined by studying the memory and registry. Using static approaches leads to a huge level of complications and challenges as it limits itself to combat the malicious content due to the unavailability of the source most of the times. Dynamic analysis overcomes these issues and provides detailed information when a monitored program is executed.

Keywords: Malware Analysis, Feature Extraction, Sandbox Environment, System Calls.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Program Interface
CIA	Confidentiality-Integrity-Availability
DLL	Dynamic Link Library
DTM-DSR	Dynamic Trust Mechanism-Dynamic Source Routing
EE	Emulated Environment
GFI	Ground Fault Interrupter
GUI	Graphic User Interface
IDS	Intrusion Detection System
IFT	Information Flow Tracking
IT	Internal Trust
KOBG	Kernel Object Behavioral Graph
OSI	Open System Interconnection
PAN	Personal Area Network
PWS	Process Wide Scope
SPS	Single Process Scope
SW	System Wide
VE	Virtual Environment
WCBG	Weighted Common Behavioral Graph

CHAPTER 1
INTRODUCTION

1.1 Introduction to Malware Analysis:

Malware is any software which is malicious in nature. Any such file is termed a malware which causes threat to the computer or any network. We basically had two intrinsic approaches for malware analysis i.e. Static analysis and Dynamic analysis. One more type of analysis discovered in the recent times is termed as Memory Analysis. In static analysis we study the code before it is run in the environment. In static analysis of the code we get to know the loop holes of the code and study the patches within the code before running it. [1]. Cyber security professionals face a very challenging task as the computer systems are threatened on a regular basis by the malicious codes i.e. Malwares. As the technology is enhancing on an exponential basis, a gradual increase in risk factors is being observed due to these Spams, Root kits, Trojans, Worms, Viruses etc, all categorizing under one species names Malware. The examination of these is segregated in terms of code which is already stated above; referred to as static analysis. The other one is based on behavioral study of the suspicious code also termed as dynamic analysis. While studying the samples, many a times we face a problem due to the lack of availability of source codes, hence inconclusive results are obtained as such programs are disassembled. Whereas, in the dynamic analysis even if the code is self-modified by the malware, it is unpacked by itself and circumvents all the constraints which come along with static analysis so as to run the sample in a controlled environment.

1.2 Motivation:

A malware analyst's task becomes facile and automated when malware samples are safely executed in a protected and simulated environment, all system calls are well monitored and a comprehensive report is generated automatically. This could be achieved in a Sandbox environment [5].

Sandbox is a simulated environment in which separate qualities and granularities of results can be obtained by following two different approaches:

1. Before executing the malware an image of the entire system is taken and then it is compared to the whole system state.
2. While executing the malware its behavior is scrutinized by making use of a specialized tool such as a debugger.

Implementation of a malware brings into account a lot of dynamic modifications in the system such as generation of a file during execution, deletion of a file before termination etc. The first approach overlooks these effects and analyses only the increscent effects. It accouches more homespun details which are at times commensurable for over viewing the task of a given binary. Though the second approach is a tough task to implement but it is preferred in the Sandbox environment as it provides us with a more amplified and elaborated set of results.

Static analysis however makes it possible to scrutinize the entire malware executions all in a single go by analyzing the source code but it is not always possible to implement this technique due to the dearth of the source codes. Also the occurrence of executable binaries whose modifications are undocumented can't be affirmed. [6].

Another name for Dynamic analysis is the Behavior-Based malware analysis. By consolidating Behavior-based analysis in a Sandbox, all appropriate system calls can be monitored and domained. Also an automated machine-readable report can be propagated through which a lot of features can be explicated such as the creation of any malicious file during the malware execution, any tampering being done with the file on Windows registry, loading certain DLL's by the malware before its execution, procurement of certain areas in the virtual memory, opening certain network connections and sending information over it and other features like what all kernel drivers and other storage devices the malware sample intrude upon.

1.3 Purpose:

Future attacks can be controlled, espied and stopped only by analyzing the malware samples and its behavior now. This scrutinizing is done by cyber security experts by making use of some professional malware analysis tools. Many crucial forensic details have to be winnowed by the experts to evaluate the life-cycle of the malware and aggrandize the threat intelligence. Even some leading-edge activities of the malware sample can be tested, rehashed and delineated by the usage of appropriate tools. Beginning with the initial threat and execution path of the malware and ending up to the callback destinations, everything is brooded upon by malware analysis.

An analysis tool lists out all the techniques used in the analyzation of a malware sample. These techniques are the methods which a particular tool embraces. A few of such commonly used and most relevant methods have been picked up and analyzed in this work. When any malicious code is cross-examined or probed, the most challenging task for an analyst is to choose an appropriate tool in a pool of abundance where each tool is clogged by the level of implementation associated with it.

A paradigmatic tool is one that postulates and amalgamates numerous options for analysis and endeavors a user-friendly environment. Such a tool offers a better backdrop for the analysis as well. For example- If user-mode is opted for the employment of a tool, the diagnosis of system calls, congenital API's can be bypassed and burked. Therefore, making the right choice of a tool which is fecund and renders comprehensive results after the malware analysis is really exigent.

1.4 Thesis Outline:

The whole thesis is organized into 6 chapters. Chapter 1 is Introduction which describes about the Malware Analysis, Motivation, purpose and finally the thesis outline. Chapter 2 is literature Survey which describes about the characteristics and applications of Malware Analysis, types of analysis, techniques used and different kinds of tools available and how to make the right decision in choosing an appropriate tool for the analysis. Chapter 3 describes the problem statement and objective. Chapter 4 is proposed criteria which describes about the proposed model and techniques in choosing an appropriate tool. Chapter 5 is Implementation and Result which describes about the implementation part and the corresponding results. Finally, in chapter 6, we conclude and discuss the future scope of the work presented.

CHAPTER 2

LITERATURE SURVEY

2.1 Dynamic Malware Analysis:

By generating the analysis report, after the execution and audit of the malware in a secured domain facilitates the electrocuting of the packed binaries i.e. the ones whose code is non-existent for analysis statically.

When we switch on to the dynamic analysis, it brings some exhortatory with it in lieu of numerous advantages. If not helved cautiously, an entire computer network could be blocked, leaving behind a completely dinged system. Hereby the analysis is at disposal with various tools & techniques in order to feature mainly its strength. Copious sandboxes like Anubis, Cuckoo, Joe, Norman, threat Expert, Document Analyzer etc are available to support the Malware Analysis Dynamically. Depending upon the apprehensive content in a binary, it is carried out for longer on shorter durations. The incitement which drives the dynamic analysis is to chalk up the changes caused in the file system or registry, any creation of mutexes or the approaches adopted by the malware, all of which can be collaborated in a single term i.e. Feature Extraction.

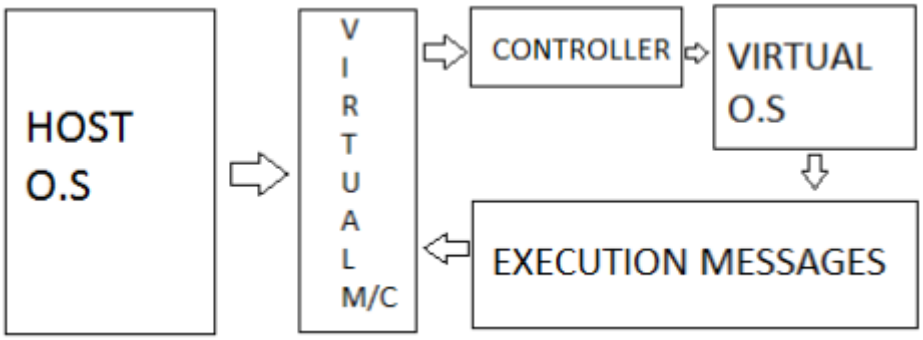


Fig 2.1 Virtual Execution of Malware

2.2 Dynamic analysis- A brief Physiognomy:

1) It bypasses the concern of unpacking the binaries and discombobulation which occurs with it.

2) The contrivance for this process involves two elementary angles i.e. A Comparative approach and a Run Time behavior analysis.

3) A comparison report is accomplished on the basis of examining the two states of the system by running a malware sample for a specific time and observing the changes contrived in the system afterwards.

4) By the application of tools available for the dynamic malware analysis one can parameterize the operations which lead to the documenting of malware characteristics. For e.g. Scrutining the operating system services which a malware has corralled.

5) To save the network and system from the disruptions caused by the run time execution, a Sandbox (derived from ballistics where a sand box is used for the testing of arms) can be used in many ways for an approach which is quiet astute.

6) Numerous techniques are incorporated in the schema of Dynamic Malware Analysis like – Function Call Monitoring, function Parameter analysis and Tracking the flow of Information.

7) It can be achieved on any Operating System i.e. Dynamic Malware Analysis is Platform independent.

8) A Dynamic Malware Analysis can be customized in lieu of the requirements of the security Aspects. Convening the environment to proceed the study, a self-tailored toolkit can be designed.

9) Lastly, the most important factor is the Cost Budgeting. The tools and techniques of Dynamic malware analysis can execute diversiform simulations on a single physical computer. This does not require a costly inveterate set-up. Rather it can be put into action via inexpensive hubs, systems, drives etc.

2.3 Regarding Tools (Consanguine attempts in the dynamic Malware Analysis):

Tools available for malware analysis work on variable and unique approaches to axiomatically scrutinize the malicious contents in a system. The execution of malware varies from malware to malware. Some are executed in a simulated or emulated environment while others run themselves via real Windows system.

In defiance of many similarities amongst the tools, each one has an ascendancy to produce an amplified, behavior based analysis report which barbarizes the entire process to a high degree. When a malware is executed in a simulated environment, the entire windows system is again put into effect to counterfeit the entire computer and a connected network as well. There is also a possibility to run a malware sample using a live internet connection. This diaphanous aspect of the environment can't be detected by the malware and their execution in this simulation can be carried out without the fear of being caught. This simulation has no side effect on other processes as they being run in the real system. For e.g. in a Norman sandbox, a lot of beneficial information may be missed in the process of ignoring the audit of such implementation.

2.4 Indoctrination of Static and Dynamic Mannerism:

Due to the breakneck increase in the quantity and quality of malware, an exclusive technique of apperception had been used by most of the anti-virus tools i.e. the Signature Based Detection which dissects the binary file making use of Static Analysis methods. However this technique has proved itself inefficient, therefore we switched to the Behavioral Analysis scheme. Overlooking the inadequacy of both schemes, their eminence can be integrated for a new method which incorporates the study of binary code and dynamic behavior as well in conjunction with some techniques of Machine Learning. All the Dexter programs and malware samples are utilized as a form of data drilling. Combining the precision rate of static i.e. 95.8% and of dynamic i.e. 97.1 %, we can armor 98.7% of success rate in Malware Analysis.

This is the era of malwares. A new malware is being created with the birth of every second. This outdates the Signature Based technique as it requires constant refurbishment. The main advantage of static model of analysis is that it opts for binary code readings which articulate the operations and framework of the op-codes. On the other hand, Run-Time behavioral analysis of a program is a boon for Dynamic method as it is very arduous to befuddle this approach. The shortcomings of this technique of running a sample in a protected field only for a limited time slot drains the heyday and does not ensures a downright characteristic of a malware sample. This is because unlike in a real time environment, a malicious code performs intermittently in a secured simulation. Therefore a joint method which amalgamates the assets of both techniques would be highly conducive and ingenious. [5][19].

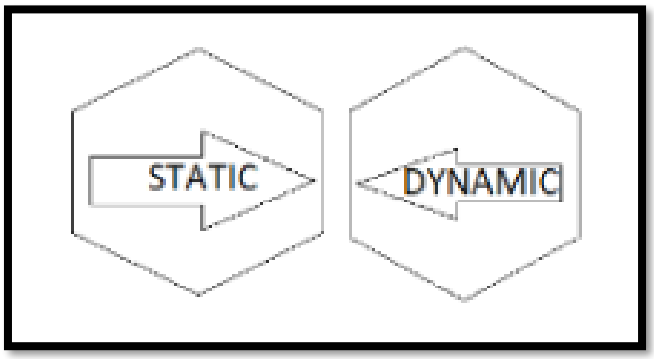


Figure 2.2: Merging Static and Dynamic Methods

2.4.1 Framework of the Proposal:

In the figure below, a detailed critique of the integrated proposal of Static and Dynamic analysis is diagramed. A community website named Virus Share had been referred to for the collection of malicious samples. The most valuable feature used as a Static attribute, the PSI i.e. Printable Strings Information is catheterized from the binaries. Every malware sample undergoes a process of distinguishing as per the global list of strings available which is created from the strings obtained after elicitation process in the beginning. The widely used tool for behavioral analysis is the CW Sandbox. This tool has been the base of all experiments being performed for this dissertation.

This hybrid analysis is accomplished on the dataset which consists of both calamitous and congenial files. Extracting PSI features is the main highlight of the static analysis procedure while the Dynamic analysis focuses upon the API (application programming interface) call sequence.

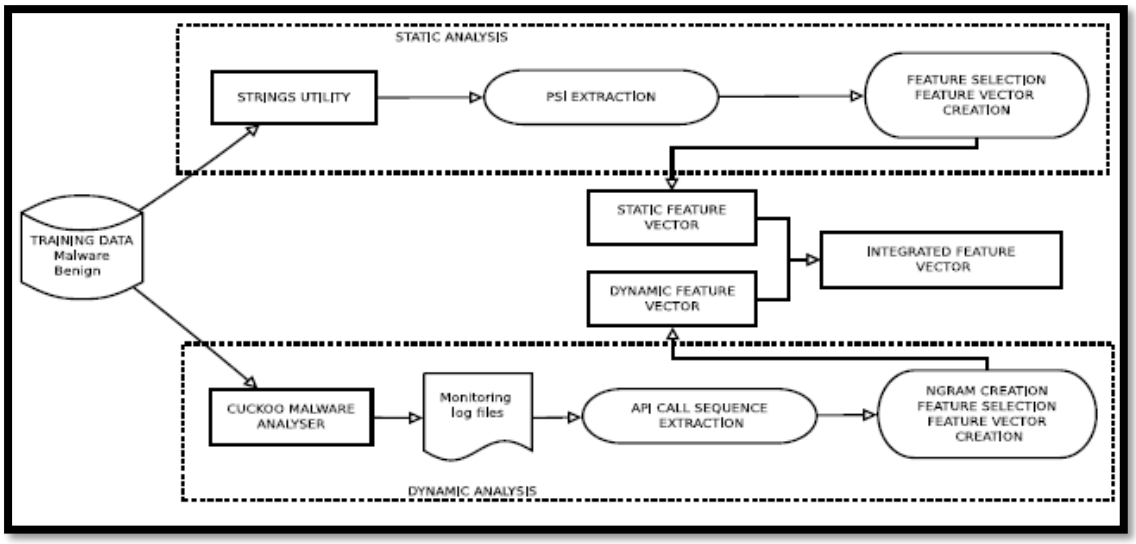


Fig.2.3 Architecture of the Indoctrination

2.4.2 Phizog of Static Perusal:

In this section only the boons of static analysis have been focused upon and mentioned.

1. All the detailed scrutiny is extremely easy to ordain as this mode of analysis does not limit itself to any categorical set of memoranda.
2. The conviction and certitude of the outcome of the analysis cannot be questioned as it is declared even before the analysis take place as the behavior of a malware cannot be apocryphaled.
3. To achieve a facile ciphering of cost reckoning and efficacy of the administration, static method of analysis is more liable to be chosen.

Algorithm for avulsing the feature via static analysis is as follows:

```
Data: Dataset :Δ containing malware and benign files f_i
Result: Feature vector and classification results
begin
    foreach f_i ∈ Δ do
        Extract strings from f_i
        Process the raw data to generate useful PSI;
        Create a table of PSI for each f_i sorted according to the frequency;
    end
    foreach f_i ∈ Δ do
        foreach PSI in the table do
            if frequency of PSI > threshold then
                Add PSI to the feature_list;
            end
        end
    end
    Create a binary feature_vector with each PSI in the feature_list as attributes;
    foreach f_i ∈ Δ do
        foreach PSI ∈ feature_list do
            if PSI is present in Table associated with f_i then
                Set value of the attribute in the vector true;
            else
                Set value of the attribute in the vector false;
            end
        end
    end
    Input feature_vector to different learning algorithms in WEKA;
end
```

Fig.2.4 Algorithm for Static feature extraction

File 1: [GetProcessWindowStation, FindFirstFile, GetLongPathName, HeapReAlloc]

File 2: [FindFirstFile, GetLongPathName, GetProcessHeap, GetLastError]

File 3: [GetLastError, FindFirstFile, GetLongPathName, GetProcAddress]

Many algorithms are looked at when we talk about classifying the features excerpted from the various binary sources. These malware features are then fed as an input to these algorithms. This section features upon the classification of malware attributes based on static analysis. There are two techniques for the same:

1. FLF(Function Length Frequency)
2. PSI(Printable Strings Information)

FLF: The number of byte codes present in a function determines its length. In the FLF method we cogitate the iteration of these function lengths and other is the orderliness of these functions. The list regarding the function lengths of any available database can be captured and a list is made in a chronological order starting from shortest to longest lengths and then to design its pattern these lengths are epitomized. The reason why this observation was dubious regarding the subsuming of articulated and fecund information in lieu of malware classification is the function length pattern which was espied. The pattern came out to be similar for the malwares of similar genus though the caboodle of function and their length was observed to be discrete. In brief Function Length Frequency (FLF) refers to the enumeration of the function in variable ranges of length.

PSI: Another way to detect and scrutinize the presence of malicious content in an unpacked binary is to elicit the printable strings encompassed within it. A "Digital Fingerprint" figures in the printable strings hobnobbed with the code which is called in by the library. This is because some strings are not deleterious by themselves but are rendered to by the library calls. An explicit position is allotted to a particular string. Therefore once the global list is created, an entire lore of strings which appears in the database can be devised.

Also, this list is being referred to for comparing a malware sample. Later on, a binary vector is used to denote the strings confided in the sample and are accounted as a true/false binary value.

Amalgamation of both extraction methods i.e. FSF and PSI proves out to be even more beneficial. For the purpose of iterating the mechanism of disassembling, IDA is preferred. These dismantled dissolutions are dispatched to a database. Afterwards, both benign and malignant files undergo a process of extraction of their feature vectors and finally the accounted result is transshipped for classification to an engine.

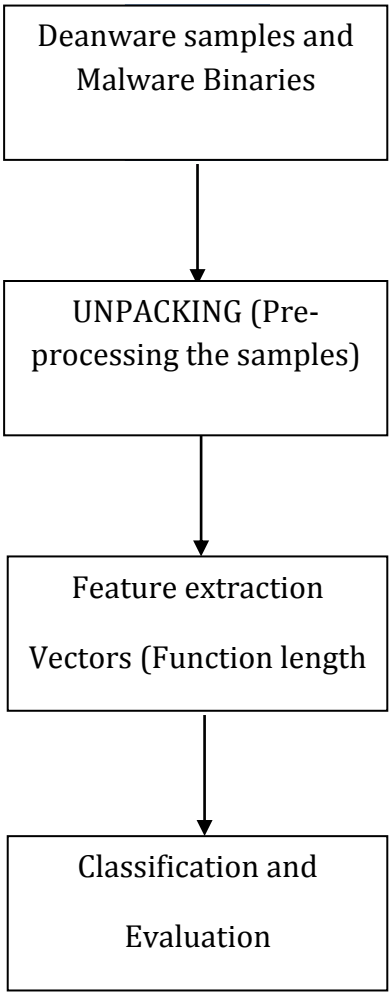


Fig. 2.5 Feature Extraction Based On Static Methods

2.4.3 Phizog of Dynamic Perusal:

To carry out the runtime analysis, the dynamic scheme proposes certain modus operandi for its procedures.

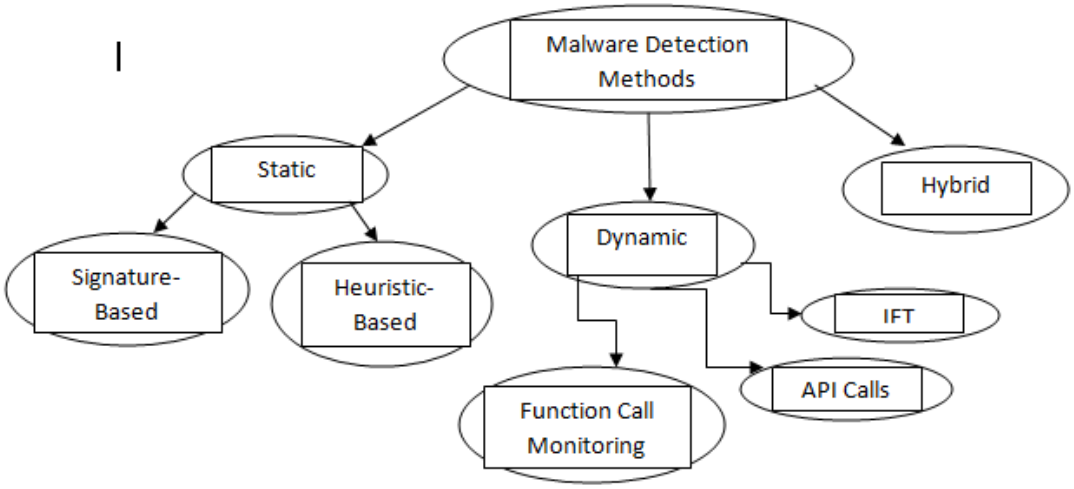


Fig.2.6 Categorization of Malware Detection Techniques

Function Call Monitoring: A code could be dexterously reused and emphasized with the support of functions. A particular vocation like calculation, file creation etc can be exhibited by a code (which is nothing but a set of instructions) by the use of functions. The scrutiny of a docket is intrguined by the presence of functions in it. A function has a characteristic to portray details of an analysis excerpted from the pursuit of implementations in a form which phonologically possesses a fecund delineation. To exemplify, a sort function executing a specific algorithm may not be credited till the time the results are analogous with the sorted input. The comportment of the program can undergo capsulisation with the help of such abstractions made in the process of assaying the codes. Execution of the program demands for the calling of certain necessary functions in the code. This can be audited by interrupting these calls. Hooking is such a method with the help of which this motto can be accomplished. In this method, a hook-function is summoned along with the betrothed function. The performance and functionality of the imperious analysis is put into action by this hook function. This Hook Function is responsible for the control and direction of the inspected program such as logging a file, estimating the input parameters or keeping a track of the summons.

Application Programming Interface: During the execution of a binary file, some function calls are made to the API as well. To distillate these API calls we make use of the behavioral analysis scheme. For the experimentation of the same, we have made use of a malware analyzer called Cuckoo Sandbox, installed in a virtual machine (VMWARE) in Ubuntu 10.04. [Virtual machine is used for providing simulation a secured environment] A malware analyzer administers the malicious file and produces a detailed report of the dissolution and investigation only after the run time execution of the binary in a secured terrain. The particulars regarding the API calls made while executing a binary are contained in a log file. Other fine-points like alteration of the registry; memory address of the Heap is also incorporated in this tally.

The OS bestows the API's so that the low level hardware could be admittanced in consideration of the application programs by the dint of system calls. To detect whether a particular file possesses any malicious content, the ubiquity of an API in the log is not a plenteous parameter. This is because for the conduction of catty avocation, the assaulter makes use of the same API. The files which categorize under the same class, refrain a larger amount of collation as compared to the files of different classes. To investigate the call sequence we use a more principled approach i.e. the n-gram method of analysis. It is a contractual array of n-items from a given sequence of data. It is an anticipating model to conjecture the next item in the sequence. In this dissertation, the call sequence we are taking into consideration is the API-call-grams. The count of analogous n-grams between the two files becomes decumbent as soon as the aggregate magnitude of n-grams increases. The n-grams which consist of 1 gram are referred to as Unigrams; Bigrams consist of 2 grams and so on. The dissection which is established via unigrams is quite similar as inspecting the presence of an API in a log. In the dissertation, we have carried out the experiments based on 3-API-call-grams and 4-API-call-grams.

Algorithm for Feature Vector Extraction Using API:

A Feature vector is generated for each file which undergoes the analysis. These files consist of both malicious and congenial data. Before undergoing the processing of these files for the extrication of API call sequence and the engenderment of 3 and 4-API-call-grams, the tally data file of the behavioral analysis is to be generated. After

this, the 3-API-call-grams and 4-API-call-grams are buttoned down according to the repetitiveness of their occurrences. In order to systemize the list of both 3 and 4-API-call-grams along with the oscillation of their contingency, these frequencies are compared to a threshold value. Corresponding to which, an API call-gram is annexed to the coterminous feature list if the frequency is greater than the threshold. Next, considering the 3 API-call-grams and 4-API-call-grams as a peculiarity, a binary feature vector is generated for both the API-call-grams as follows. For every API-call-gram belonging to the feature list, the value of their trait in the vector is set to 'true' if that particular call-gram is existent in the table which is leagued with the set of binary files. On the contrary, the vector value is set to 'false'.

This feature vector generated can be incorporated to various learning algorithms, for e.g. WEKA (Waikato Environment for Knowledge Analysis), which is an accumulation of Machine Learning algorithms used for data mining tasks. Summarizing, a coalesce of articulated functions which serve as a platform to perform operations like maneuvering the files, communicating over network etc is termed as an Application Programming Interface. On every layer of abstraction, these API's are available and perform contradistinctive tasks as per the requirement of that layer. For e.g. at the TCP layer, it broods upon the content which is being conducted via TCP packets or permitting the applications for the creation into the raw sockets without any intermediate.

The API scheme can be amalgamated with the PSI method for the extraction of binary files.

2.4.4 Ramification of API Misapplication:

API's serve as a porthole into a numerous utilization and so it could be exploited. While attackers and hackers fail to notice certain trajectories to implement an attack, API's help them to route the harmful clues within the indigenous circumference of the application. They provide great privilege for a treacherous ascendency by intensifying the expanse of area available for launching attacks in addition to consigning the applications under the ambit of the hacker. The shielded echelons which are vested inside the DMZ i.e. the demilitarized zone tend to maneuver towards the client application which is inhabited on the internet. This is achieved by the existence of a chapped or granularity boundary which endorses the function calls to perform the

above action. Hackers are constantly bestowed with the thoroughfares by the virtue of which they can carry on destructive exploits. This asset is given by the API's which set the granularity boundary into motion towards the client application. As a result of this privilege, applications which are galvanized by the API's process all the function calls on their own. In this way numerous amenabilities are exposed and the attack domain is emphasized which poses a serious threat to the organizations which broadcast their API's to online network infrastructure.

On studying about the API's, we come to know regarding their exclusiveness. Every API is onliest. An unexampled threat which is established on the grounds of its elementary pursuit is borne by every case in point. That is why the elbowroom of the conceivable attacks which correspond against the API's is a parameter which is very appalling. Hence putting the security of API's in a questionable arena. The possible categories of attacks against the API's can be buttoned down as follows:

1. Any slip-up or glitch which persists in the session tracking, endorsement, and authentication is easily manipulated by the identity attacks. These paroxysms consummate the diaspora of awful utilities from the web towards the API eventuality.
2. The data which is not encoded and does not undergo the signature making is easily abused and can be manipulated for no good use. This is done by the Man-In-The-Middle attacks which also obstructs the legitimate dealings, agreements and executions going on over the network.
3. The information which is sent over an API via function calls: like the HTTP header, query framework and guidelines, URL specifications, etc can be misused and manipulated wrongly via the parameter attacks.

The prototype magnum of a Parameter attack is considered to be the SQL Injection. The purpose and helt-bent of the fundamental SQL template can be completely and primitively modified by triggering a valid and legal input into the database query. Certain parameters are drafted for this purpose such as the Snippets in the JavaScript. This injection of scripts in the system takes undue advantage of the system and results in interpretation of the acquiesced data as a script itself. A productive and efficacious technique for alleviation against the API attacks can be architectured by taking into account all the above classification of attacks. When a capricious file is to be

downloaded from the internet or any serendipitous file is to be executed or run in a local machine, an API permits for carrying out all such operations. The indoctrination of both such imprimatur by the API's is responsible for the septicity with 'Drive-by-Download' attacks.

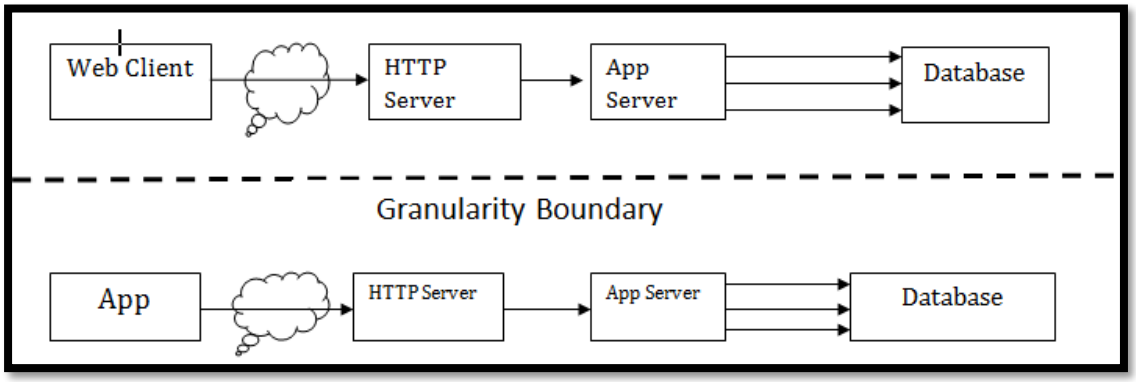


Fig.2.7 Attack Surface provided by API's

2.4.5 Exploratory Establishment and Interpretation:

The PSI scheme is taken into consideration for static analysis. Number of malicious file is almost double to that of the benevolent file. The combined data undergoes the process of scrutining via printable strings applicability. As mentioned earlier, the concomitant specifications comprise of an Ubuntu 14.04 machine. A simulation is setup in which each binary file is processed under the strings utility. The outcome of the analysis report consists of the PSI attributes. The strings which have a length greater than 8 bytes are catheterized and are further supplied as an advocacy to the algorithm chosen for final feature extraction.

The static analysis is followed by the behavioral analysis mechanism which is carried out in the CW Sandbox. This is a malware examiner which carries out the analysis on every binary file used in the static method. The characteristic of Dynamic Analysis chosen is the API call sequence and the output file constitutes to the production of a record which comprehends the API call sequence. The conjectural setup consists of Ubuntu 10.04, long term support (LTS) operating system and patterned in a way to be environment friendly with Oracle VM Virtual Box version 4.2.16 which consisted of three virtual windows XP machines also referred to as analysis host machines. This host machine is the workplace for the execution of all the binary files as well as the nascence for N-grams API sequence file in the dataset. A machine learning tool WEKA is exerted to classify the feature vectors used in the analysis.

2.4.6 Related work:

Chas Tomlin [8] He came up with a tool popularly named as Chas Tomlin's 'Litterbox' based on a real window system execution in which the host machine is compelled to reboot after 60 seconds of execution from a Linux image. This tool extricates the window registry, aggravates the windows severance and recrudesces the state of windows back to the inchoate clean stipulation. The predominant centralization of the tool is about the network activity, therefore while carrying out the analysis all the incoming IRC connection requests are given a positive acknowledgement as a virtual connection of the host machine is made with the currently operated IRC server. A huge success rate of the IRC servers proves to be a favored circumstance for CW sandbox unlike the C&C severs which is heretofore extenuated. However Litterbox is unable to watchdog the dynamic and behavioral consequences of a malware.

Truman: The Reusable Unknown Malware Analysis Net follows the same pathway as Chas Tomlin's Litterbox.

Galen Hunt and Doug Brubacher [9] both of them projected a framework which was more of an athenaeum than a tool for the contrivance of discretionary and random window functions, named Detours. Though this library provisions the automated deputation of malware analysis but renders a weak pliability and extensibility over the ministered functions as well as a diminishing jurisdiction over the same.

Stephanie Forrest and Steven Hofmeyer [11] outlined a rubric or a schema for the IDS system with the help of which the transgression of the system could be espied, discerned and revealed even at the elite and confidential tier of processing. Another platform called Systracer was brought in by **Provos** which terrorizes and environs the processes being run in the system and expedited by the monitoring of system call sequences.

Z.Salehi et. Al [7] outlined a procedure to detect and malwares which was based upon API calls and their expostulation. The feature vector comprises of some API calls and its arguments to examine and effectuate the analysis process. The best case of the classification which utilized RF i.e. rain forest algorithm constituted a success rate of 98.4%. According to the persistence and recurrence of a threshold value as

well as an API **R. Tian et. Al** projected an analysis scheme to categorize and demarcate congenial and malicious files by taking into consideration the memoir and evidences of API calls. A precision of 97% was achieved by this method in which a the feature vector which is a binary value indeed is marked as 0 if not present and 1 if the feature is contemporaneous.

Another proposition was laid down which comprises of an amalgamation of both static as well dynamic feature sets. **Rafiqul Islam et. Al** accumulated the information regarding the printable strings and symbolized them as features for every FLF i.e. the function length frequency of the transacted file. This boundarized the static feature set, whereas the dynamic set of attributes of specification, criterion as well as the function names of the API. The verdict and conclusion of this indoctrinated approach provided a accuracy rate of 97.05%.

Younghee Park et. al [14] developed a scheme to catalog and diagnosticate the behavior of malicious samples by clustering the graphs. From the information gathered regarding the decorum and demeanor of the malwares a KOBG is fabricated. KOBG stands for Kernel Object Behavioral Graph. A graph excavating technique referred to as WCBG [Weighted Common Behavioral Graph] generates a bourgeois graph for the entire cluster when a the KOBG's belong to the same group of malware samplings and precedents. Any unconventional and bizarre modifications of the deleterious samples could be determined by examining the eventuating WCBG.

A schematic approached proposed by **Chatchai Liangboonprakong et. Al.** stakes a precision and correctness of 96.6%. He referred to a technique in which static N-grams are taken into consideration for sorting and collocating the progeny of various malignant samples. This modus operandi is planted upon the feature set of succedent patterns of N-grams. The hexadecimal strings which are annihilated and extricated from the binary executables catheterize the N-grams, whose template is ordained and commanded to generate a feature vector and systematize it by unearthing the periodically and intermittently occurring sequences.

2.5 Tools of Dynamic Malware Analysis

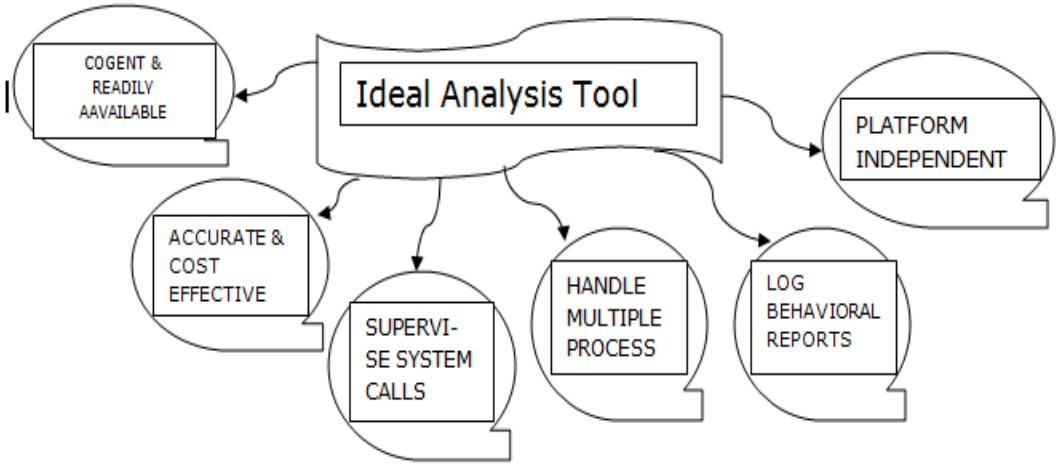


Fig.2.8 Parameters to choose an archetypal analysis tool

In defiance of some homogeneity extant in the previous tools available and the approach discussed in this work, CW sandbox withstands the gratification to proffer the automation, meticulous report generation and the entire analysis procedure at a surpassing magnitude. CW sandbox unlike Norman sandbox does not carry out the entire impersonation in a simulated environment with feasibility to accomplish the simulation of binary files even via live internet connection. In this scenario, a malware process is unable to meddle with the system or cause any defilement because such surroundings are mostly transparent to the malware and do not allow any kind of alteration to be caused by the malicious code in the current processes being run as in a simulated environment no operations are being transacted. As a result, most of the beneficial and important clues, instructions and tidings are left unnoticed and dilapidated. To dodge as well as elude such circumstances and not to jeopardize the security of a system, CW sandbox is underpinned with a real operating system and permits the deleterious code to meddle with the system by accessing only some constrained rampage being devised by the API hooking.

Another comparison of cwsandbox is drawn with TTAnalyse which is though laid upon the same foundation of API hooking, but dissents in the executive aspect. Cwsandbox makes use of a virtual machine to perform the analysis, whereas, a pc emulator called QEMU is endorsed for the examining via TTAnalyse tool.

2.5.1 Norman Sandbox: It is an example of a foregoing idea of malware detection which forbids any connection to the non-virtual internet and also contends as well as outvies the network surroundings may it be HTTP, FTP, SMTP, DNS, IRC and P2P. However this criterion of behavioral analysis is not able to manipulate with the remote host networks within a real cyber world as Norman sandbox constitutes of a method which customizes the conduct of transmitting the data as well as substantiates the emulation when communication takes place with a C&C server or any file server. The execution of a malware binary in a protected environment by this sandbox works by reinforcing the foundational windows system with the whole computer system connected to a single network. It takes generally 6-24 hours to analyze and report a new threat in the system. It makes use of API log view and summarizes the report on an account of the in-depth examination of the behavior of a particular malware. The information segment comprises of the examination and evaluation regarding:

- ✓ Network services
- ✓ Modifications in the registry as well the system
- ✓ Tampering with computer's file
- ✓ Information regarding the processor and OS
- ✓ Categories of malwares and the files i.e. W32/Downloader, W32/Worm, W32/Backdoor etc

System requirements for Norman sandbox encompass:

- ✓ Pentium PIII
- ✓ 512 MB RAM, 50 MB HD
- ✓ OS- XP, Windows 2000/2003
- ✓ For the PRO version of sandbox- free BSD Linux running on Intel

2.5.2 Ether: A transparent malware analyzer which deviates its procedure from stereotype approaches of virtualizing the API's or outveing a complete or fragmented system because every transparency necessity could be achieved by such methods. It cabinates its existence in the exterior of targeted operating system and extensify the analysis approach to hardware virtualization. The underlying requirement for Ether is software named Xen hypervisor version 3.1.0. The target for analysis is Windows XP and ether executes as a constituent of the hypervisor layer. Some attacks such as

memory attacks, timing attacks can defer the efficiency of ether but the changes it brings in the environment of the analysis are hardly perceptible.

Detection methods and alleviation:

- ✓ CPU registers
- ✓ In-memory presence
- ✓ Memory protection
- ✓ Privileged Instruction handling
- ✓ Emulating instructions

Hardware requirement for ether which is Intel VT is prone to some vulnerabilities which can create a huge trouble for ether and it may be traced. Ether forms a base for two another tools:

1. Ether Unpack
2. Ether Trace

Effectiveness of ether trace as well as ether unpack is supported by some packing tools like PEcompact, UPX, WinUpack, RCryptor, ASPack, MEW, Armadillo, Themida, Molebox etc. Ether could decrypt some hidden codes because it makes use of hardware instructions and not the software emulations for performing its investigation.

2.5.3 Tqana: It is a behavior-based analysis tool to detect activities of spywares. The system requirement for Tqana is:

- ✓ Intel x86 system
- ✓ QEMU x86 emulator
- ✓ Windows 2000 with no service packs [guest system]
- ✓ Internet Explorer
- ✓ Spyware [BHO]

This tool focuses upon the analysis by adopting the method of tainting. A taint tracking mechanism is installed in the guest machine based upon whose categorization techniques Tqana is able to designate the spyware files as congenial and vindictive. The intrinsic plug-ins to the Internet explorer i.e. the BHO's [Brower Helper Objects are targeted and carries an logged off investigation i.e. while the

network is in an offline state and manifests and renders a comprehensive examination of the spyware. This tainting system works efficiently for the arithmetic and logical operation to be carried out but lamentably, it is unable to thumbnail the prospects of an operand's address when a table or a set of array is taken into consideration. BHO is endowed with all the commands of the executing code and runs itself but it also sometimes makes a function call to carry out certain operations, so that case will not be ignored. The algorithm followed is:

- ✓ Relegate the consummation of codes from IE to the BHO
- ✓ Archive the valuation of prevalent stack pointer
- ✓ Code segment of BHO possesses the instruction set, hence every enactment and implementation of the same is cited
- ✓ Account for the position of the value of current stack pointer with the stored value in the BHO
- ✓ If the current value is less than the stored value, process is enforced
- ✓ Else again the beginning of the algorithm is recommenced

Tqana makes use of an emulator i.e. QEMU which possesses IFT i.e. information flow tracking competency to fecundate the amenable and confidential data through the system. While the scrutinization of the malicious sample is being carried out, a script called AutoIT or BENETT outvies and patternizies the reciprocal action taking place between the guest machine and the user.

The gimmicks of inter process communication taken into consideration could be the shared memory data or any other operations performed within the processes. As the installation and complete working of Tqana is rendered in the exterior of the system, that is why it enables the tool to effectuate a furtive and an enigmatic supervision of the spyware. Also in the setup of the guest system no peripheral of kernel nodes is imperative and mandatory. The framework and configuration of the system memory is essential to restore and redeem to procure and capture the information from the contended memory. However the steadiness, endurance and sustainability of such actions is not ascertained and affirmed across the service packs and various versions of the operating systems.

2.6 Comparative Analysis of Tools

TOOLS /STRATEGIES	USER-MODE	KERNEL-MODE	API HOOK-ING	SYSTEM-CALLS	IFT	TRACE	PACKET ANALYSER
CW-SANDBOX	△	△	△	△	⊗	△	⊗
ETHER	⊗	△	⊗	△	⊗	⊗	⊗
TQANA	⊗	△	△	△	⊗	⊗	⊗
NORMAN	△	△	△	△	△	⊗	⊗

Table 2.1: Comparative Analysis of Dynamic Analysis Tools

Table 2.1, above, describes the various methodologies of various tools and the strategies adopted by them in analyzing various malwares. Different parameters of the scrutiny are being taken into consideration for the comparability of various tools.

The interpretation and the assay of various tools and techniques available for the behavioral analysis of Malwares substantiates a disengage idea and an outlook of actualizing and executing these tools. It also postulates a convinced and definite idea to carry the maneuvering of parameters and techniques which a particular tool subsumes and consolidates. The user also gets benefitted as he can get knowledge regarding the stewardship and sustentation of various techniques used for analysis such as Function Call Monitoring, IFT analysis, PSI, API hooking etc.

- CW-Sandbox and Tqana focus upon the Kernel level implementation. The framework of data which is feasible and lies within the approachable range can be altered and tempered with by insinuating a loadable kernel module. Also the function pointers can be mutated by the same techniques. The attributes which a kernel consigns is also attainable. The module which is injected in the Kernel comprises of a permit to carry out certain contraband &

deleterious activities. This permit is nothing but an apprenticeship of data displacement and hauling which targets the memory heap. These malicious operations also focus upon manipulating the memory address by forging the allusion of Kernel symbols which are prohibited to access.

CHAPTER 3

PROBLEM STATEMENT AND OBJECTIVE

3.1 Problem Statement:

To skirmish the blatant and opprobrious malwares, signature based techniques had been adopted by security analysts as well as virus scanners. Slowly becoming unsusceptible and immune to such methods, worms and viruses etc relinquish from being discovered by disguising themselves or by exhibiting the surveillance clandestinely causing no influence to the vulnerable machines. With the application of reverse engineering and standard non-automated techniques to annihilate the malicious software as well as wrangling with the malwares proves to be futile and trifling.

For a timely prevention from the activities of malwares and the denouement they cause to the system on a whole, a propitious, promising, up-to-date and a favorable analysis tool has to be preferred and designated. Behavioral analysis is appointed over static because of its run-time analysis of pernicious codes. Similarly, an automated analysis tool which renders a persuasive, efficacious, accurate and an impeccable interpretation should be favored. To pick up an appropriate tool at the time of analysis is a real head scratching task. Because a single tool does not cover all the fields of investigation, hence we have to use multiple tools which involves a lot of time, money as well as a huge risk to the system in which a malware has invaded. Some parameters a tool analyzes, is not even necessary to take counter actions against it. Therefore the choice of a perfect tool is the act of smartness required by a good analyst.

3.2 Objective:

An ideal tool carries out an expeditious and meticulous report generation without any human intercession, examining and contemplating which an analyst triggers the process of automation which consists of refurbishing signatures of the intrusion detection system to protect the network. Our objective is to categorize various dynamic analysis tools available for the scrutinization of malware behaviors and results. Since an analyst blindly depends upon the information and results generated by the tool to apprise and evaluate the threat, hence a tool must be cogent, persuasive and germane enough to indite the significant avocations of malwares, refrain any false positives as well as to not neglect and overpass any of the accomplished range of capabilities. This dissertation combines the behavioral analysis scheme within CW sandbox in order to stalk and supervise all congruous system calls and establish a programmed machine-readable report. CW sandbox is a suffice platform to render cogency, validity, exactitude, precision, verity as well as automation while carrying out analysis on an operating system. It constitutes of dynamic analysis techniques like- API hooking, Function call monitoring as well information flow tracking so as to avoid being caught by the malwares while examining them. This is achieved by enforcing imperative and quintessential root functionalities. The novelty of these techniques is however denied as they already exist but their non-pareil and unparagoned congregation is an epitome of an entirely inked, assertably effective and a splendidly simple and easy to use analysis tool.

CHAPTER 4
TECHNIQUES AND RESULTS

4.1 Techniques and Results:

The modus operandi to supervise the programs by formulating the data and content of the docket i.e. the Information Flow Tracking is also named as Taint Analysis, has a set of indispensable apportions which ought to be considered while analysis :

A) Direct Tainted Data
B) Tainted Determinants and Terminals
C) Taint- Red Tape Tactics
D) Superfluous and Inadequate Tainting

Direct Tainted Data: The target and terminal values are impaired and infected inevitably and indubitably at each instance of the dynamism or mobility of the source value to a different operand. By adopting this approach, the operand is stamped and branded without any prevarication i.e. it is marked directly as the name adumbrates. Unambiguous marking of the operands can occur innumerably in the mandate of engrossment. The maneuver of tainted operands must be superintended fitly.

Tainted Determinants and Terminals: The tainted and marked data sources are incorporated in a tracking systemization which traces the systemization which traces the cascade of information and stipulates the precarious, tricky and predisposed data dossiers. Marked and conspicuous data is fed as an input to the system. If any constituent inside the tracking structure reciprocates and acknowledges the onset, it is alluded and exemplified as the tainted target. It is then pigeonholed and classified by the tainted sources which were fueled into the supervised contrivance of the information flow tracking system.

Taint Red-Tape Tactics: These are the stratagem or the policies which govern the execution and consummation of the taint analysis. This tactic comprises of three foundational and rudimentary peculiarities namely:

1) Initiation
2) Propagation
3) Investigation

This regimen is to abide by within the information flow tracking system. The initiation step comprises of commanding the method to introduce a taint into the system. The gradation to propagate the taint analysis refers to the stipulation and particularization of the stages of analysis and revising the stage in which a particular investigation is being carried out. The final step comprises of authenticating the taint status and steer a coterminous action against it.

Superfluous and Inadequate Tainting: A specific amount of precise and marked faux pas or we can say some absurdity could be bumped into while carrying out the taint examination. A situation of superfluous tainting eventualizes when a particular data which is not simulated from the tainted sources is misleadingly designated as a tainted data. On the contrary, when a tainted value is consigned to oblivion and is left unmarked is a predicament called as inadequate or under tainting.

The pre-requisite and a cardinal ingredient of an analysis tool is its dexterity to inspect and review multiple processes. The activation and elicitation of a malware is contingent on various extramural and speculative conditions such as Logic Bombs which are secretly infused into program to accomplish harmful implementation if environmental conditions are satisfied and appropriate.

4.2 Analysis of Results

By utilizing the characteristic of an analysis tool to carry out multiple processes at the same time, a comprehensive and a painstaking examination and breakdown study can performed by an analyst. Assorted degrees of the operating system could be exploited and manipulated to execute and materialize the structure as well as orderliness of the analysis. The computation of the layout of analysis technique is epitomized by the categories of malwares being examined as well as the evulsions' stratums of the information assessed. The two levels of analysis i.e. User level and Kernel level hold different perspectives of implementing the analysis. User level focuses upon the API calls which subsume a resplendent rundown of the functions which are called upon during the scrutiny process as well as the administration of API calls. Adversatively, inestimable erudition regarding intercede native API's and system calls is extricated via Kernel mode analysis. Reconnoitering a malware sample in a congenital environment in-transits the endangerment and precariousness of bugs and virus. However, this point in question could be subjugated if the analysis is performed in a well supervised and a custodial environment. To postulate the pursuit and performance of the gimmick as well as maneuvering of tools and techniques, the analysis is substantiated and ratified. It also focuses upon the stewardship stipulated for Function Call Monitoring and IFT analysis. Tools like VM-Ray, Tqana, CW-Sandbox and Hook-Finder pinpoint on the kernel level usage. API's are used by malicious codes to make common system calls to effectuate a elite admittance. The techniques though which system calls could be examined are incorporated in the tools used. A tool named Hook-finder uniquely and expressly endorses the IFT functioning to confront and grapple some of the impenetrable and esoteric hooks within a system. Vitiation or Tainting mechanism which is adopted to examine and audit the exudation of consequential as well as integral information is delineated by Panorama and Tqana. Since the most time consuming and tedious task is to make use of an appropriate tool which covers necessary attributes required for examination and investigation of the malware's behavior and changes caused in the system, this dissertation focuses upon the downright and paramount set of quirks as well as the idiosyncrasies which are obligatory to lay down a speculated but elaborated implication for the perfect comparison of the malware analysis tools.

CHAPTER 5
CONCLUSION AND FUTUER SCOPE

5.1 Conclusion:

This dissertation emblematizes the indoctrination i.e. combination of static and behavioral analysis. It is pulled off from this work that rather than using individual schemas for scrutining the malware, an amalgamated mechanism which incorporates the plus points of both, produces greater espial meticulousness. Cataloguing of binary files is established on the avenue of support vector doohickey of Machine Learning. This mechanism overpowers the Random Forest method and is highly improvised. It is visualized from the simulations that integrated approach caters an apprehension certainty or precision by 1.5%.

The security aspect against malwares can be contrived only after acumenning the behavior of a particular sample in an environment. Without a thorough knowledge regarding the literature of various tools and mechanisms to analyze a malware's actions, the eluding measures it takes into consideration, one cannot directly jump to the remedies and antivenins of the same. A consummate tool advocates an analyst to obtain a breakneck and detailed version of a malware's decorum. Customarily, the tools which belong to the genre of behavior analysis consider two aspects pay-rolled by the malware, i.e. API's and System Calls. Hence investigating the API's being importunate and the system calls being initiated is an expedient attribute in order to communicate with the domain of a detrimental sample. These tools also categorize the function calls in a syntactic way and are incorporated with wide range of components such as to let the analyst eye-ball and get hold of the mannerism in which a deleterious data is fecundates a system. All these dead-giveaways aid an analyst to diagnosticate the atrocious aspects of a malware sample. Behavioral analysis conjecture the information obtained from an analysis report into a conclusion about prioritizing the malware samples to undergo a scrutining process. This is done by aggregating the specimens into rational categories corresponding to the manner in which a malevolent data conducts itself. An analysis tool axiomatically contrives the discernible reports of upcoming fashionable impendence and compares the trends

with the categorized samples. Thousands of neoteric malignant samples are broadcasted by the hackers and other notorious communities of attackers which make it obligatory and elementary to itemize the results obtained from previous analysis. Hence a felicitous analysis tool caters a cloudless interpretation to an analyst concerning the tenue, demeanor and deportment of a particular case history of a malware. Also, it concocts the underpinning and groundwork to materialize and invoke the antivenins and correct counteragents with a judicious time slot and in a pertinent oddity.

5.2 Future Scope:

Various permutations and combinations can be applied on the feature extraction mechanisms, hence comparing the accuracy of malware detection with previous results. The scenarios for the future in malware analysis are expected to be vividly accentuated. With the invasion of unaccustomed and contradistinctive cyber-attacks on a daily basis, field of malware analysis is conventional to invent new features, techniques for scrutiny of malwares and other customized tools. The techniques used in this dissertation can be aggrandized for the augmentation of unmasking and ferreting out the green-eyed and desirous samples of malware in other applications like the Java Script code. The apperception and cataloging of various shell codes used in Java Script malwares can be bolstered up and remodeled. Also, the improvisation and rendition which is conversant by some sandbox environments can be utilized to materialize and execute a browser addendum for the fanatical snag of Drive-by-Download attacks.

Another upcoming and advancing era of digitalization is DNA data storage. Since the computer storage is day by day drowning into the whirlpool of malwares and gawkily affected by the attacks related to them, data can now be preserved in a single gram of a DNA strand. This research field popularly known as The Human Genome Projects can be furthered reconnoitered for the gospels that if data is hoarded in a DNA, could it be preyed upon, if yes then what would be its ramifications on the human body and how the information could be ensconced. This interdisciplinary field has extremely interesting research fields. One of such area being to observe the passing of stored data from a parent to offspring via the genes and the methods to constrain and quell the transfer. Thus future work could focus upon how to blend Data Security with Genetics.

REFERENCES

[1] William Enck, Peter Gilbert, Byung-Gon Chun, Landon P.Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N.Sheth. Taintdroid: an information flow tracking system for realtime privacy monitoring on smartphones. In Precedings of the 9th USENIX conference on Operating systems design and Implementation, ODSI'10, pages 1-6, Berkeley, CA, USA, 2010. USENIX Association.

[2] Malware Analyst's Cookbook by Michael Hale Igh, Stevgen Adair and Mathew Richard

[3] Practical Malware Analysis (The Hands-on Guide to Dissecting Malicious Software) by Michael Sikorkski and Andrew Honig

[4] Contagiodump.blogspot.in/2010/06/malware-analysis-and-forensics-tools.html

[5] Static and Dynamic analysis for android malware detection by Ankita Kapratwar.

[6] A Survey on supervised method for detection of malware by Nisha Badwaik and Vijay Bagadi

[7] Detecting malware and sandbox evasion techniques by Dilshan Keragala, dkeragala@att.net.

[8] Malware Detection Using Windows API Sequence and Machine Learning. *http://research.ijcaonline.org/volumw43/number17/pxc3878715.pdf*

[9] A New generic Taxonomy on Hybrid Malware Detection Technique

[10] A Survey of Malware Detection Techniques, http://www.serc.net/system/files/SERC-TR-286.pdf

[11] A Malware instruction set for behavior based analysis.

[12] Asia Slowinskaand Herbert Bos. Pointless tainting, evaluating the practicality of pointer tainting, 2009

[13] C. Willems, T. Holz, and F. Frieling, Toward automated dynamic malware analysis using cwsandbox. Security Privacy, IEEE, 5(2):32- 39,2007

[14] Norman Solutions. Norman Sandbox Whitepaper, 2003.

[15] Kannhavong, B., Nakayama, H., Nemoto, Y., Kato, N. and Jamalipour, A., 2007. A survey of routing attacks in mobile ad hoc networks. *Wireless communications, IEEE*, *14*(5), pp.85-91.

[16] Ameza, F., Assam, N. and Beghdad, R., 2010. Defending AODV routing protocol against the black hole attack. *International Journal of Computer Science and Information Security*, *8*(2).

[17] Li, J., Li, R. and Kato, J., 2008. Future trust management framework for mobile ad hoc networks. *Communications Magazine, IEEE*, *46*(4), pp.108-114.

[18] Val´erie Viet Triem Tong, Jean-Fran¸cois Lalande, Mourad Leslous 2016 Challenges in Android Malware Analysis.

[19] Rakesh Singh Kunwar and Priyanka Sharma 2016. Malware Analysis Tools and Techniques. *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies Article No. 144*

[20] Aditya K.Sood and Sherali Zedally 2016. Drive-By-Download Attacks: A Comparative Study

[21] Marco Cova, Christopher Kruegel and Giovanni Vigna- 2010. Detection and Analysis of Drive-By-Download Attacks and Malicious Java Script Code

[22] Joash W.J.Tan, Roland H.C. Yap, 2016. Detecting Malware through anti-analysis Signals- A Preliminary Study.

[23] Adaptive semantics-aware malware classification- 2016. Bojan Kolosnjaji, Appostolis Zarras, Tamas Lengyel, George Websert and Claudia Eckert.

[24] Evolution, Detection and Analysis of Malware for Smart Devices 2017. Guillermo Suarez-Tangil, Juan E. Tapiador, Pedro Periz-Lopez and Arturo Ribagorda.

[25] A Framework for Dynamic Malware Analysis Based on Behavior Artifacts 2017. T.G. Gregory Paul and T.Gireesh Kumar

ANNEXURE I

LIST OF PUBLICATIONS

- Chahak, Anil Kumar Verma, "**A Comparative Analysis of Dynamic Malware Analysis Tools**", in Proc. International Conference on Computer Communications and Networks (ICCCN 2017), pp 476-482, NITTTR CHD. [Accepted and Published] [Ref. No- ICCCN- 17-170]

- Chahak, "**Wireless Sensor Networks in view of Intrusion Detection Systems",** in 3rd DAV National Conference 2016, STEHM'16 DAVIET, pp 208-209. [Accepted and Published]

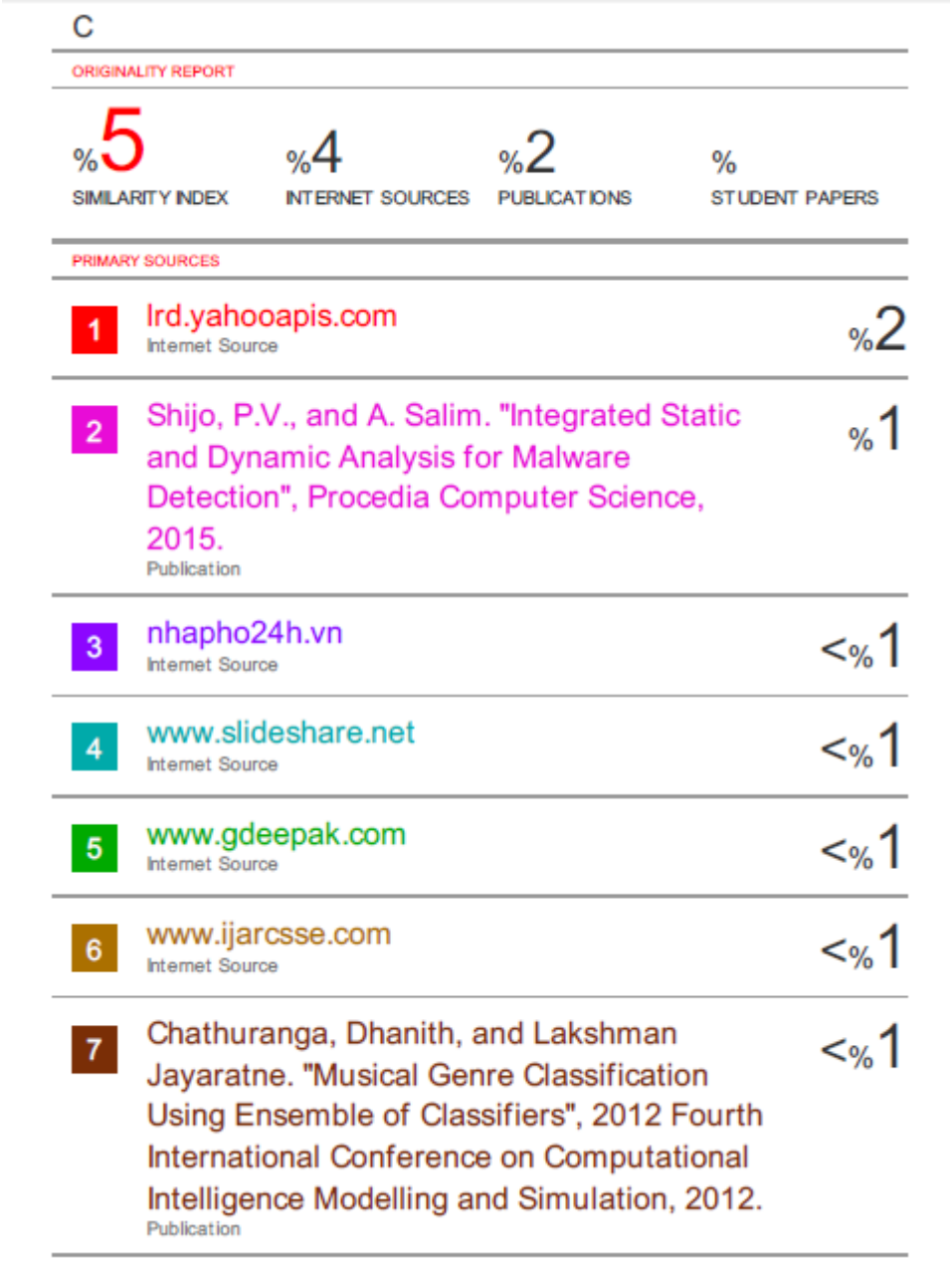
ANNEXURE II

VIDEO PRESENTATION

https://youtu.be/wh3cHsZzCmU

ANNEXURE III

PLAGIARISM REPORT



C

ORIGINALITY REPORT

%5	%4	%2	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	lrd.yahooapis.com Internet Source	%2
2	Shijo, P.V., and A. Salim. "Integrated Static and Dynamic Analysis for Malware Detection", Procedia Computer Science, 2015. Publication	%1
3	nhapho24h.vn Internet Source	<%1
4	www.slideshare.net Internet Source	<%1
5	www.gdeepak.com Internet Source	<%1
6	www.ijarcsse.com Internet Source	<%1
7	Chathuranga, Dhanith, and Lakshman Jayaratne. "Musical Genre Classification Using Ensemble of Classifiers", 2012 Fourth International Conference on Computational Intelligence Modelling and Simulation, 2012. Publication	<%1

8	Park, Younghee, Douglas S. Reeves, and Mark Stamp. "Deriving common malware behavior through graph clustering", Computers & Security, 2013. Publication	<%1
9	dspace.thapar.edu:8080 Internet Source	<%1
10	ahvaz.ist.unomaha.edu Internet Source	<%1
11	www.scribd.com Internet Source	<%1
12	shodhganga.inflibnet.ac.in Internet Source	<%1
13	Carsten Willems. "", IEEE Security and Privacy Magazine, 3/2007 Publication	<%1
14	insights.ubuntu.com Internet Source	<%1
15	qmro.qmul.ac.uk Internet Source	<%1
16	aut.researchgateway.ac.nz Internet Source	<%1
17	ArtemPaulMonirulWenke DinaburgRoyalSharifLee. "Ether", Proceedings of the 15th ACM conference on Computer and communications security - CCS 08 CCS 08, 2008	<%1

	Publication	
18	"Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Applications", Springer Nature, 2017 Publication	<%1
19	docplayer.net Internet Source	<%1
20	gdeepak.com Internet Source	<%1

EXCLUDE QUOTES	ON	EXCLUDE MATCHES	OFF
EXCLUDE BIBLIOGRAPHY	ON		