ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 4, April 2016

Efficient Detection of Malicious Worms with different Analysis methods and Techniques

B. Rajesh, Y.R. Janardhan Reddy, C.V. Chakradhar

***Abstract*— Malwares are harmful programmings. Malwares are intended to harm computer Networks without the information of the proprietor utilizing the Network. Programmings from trustworthy merchants similarly contain malicious code that influences the computer Network or releases data's to remote servers. Malware's includes computer viruses, spyware, dishonest ad-ware, rootkits, Trojans, dialers etc. The paper concentrates on various Malware detection techniques and analysis methods. Analysis methods include static analysis and dynamic analysis and Hybrid analysis and techniques like signature based detection and Behavior -based Detection technique. Dynamic Malware Analysis is preferable for Analyzing the Malware than Static Analysis this paper includes the Limitations of Static Malware Analysis and tools of Dynamic Malware Analysis and also about Hybrid Analysis.**

***Index Terms*— Static Analysis, Dynamic Analysis, Hybrid Analysis, Signature Based Detection, Behavior Based Detection.**

I. INTRODUCTION

Malware is malicious software which enters system without authorization of users. Malware is the term created from merging two words 'malicious' and 'software'. Malware is a very big threat in today's computing world.[1] It continues to grow in size and advances to complexity. Number of organizations tries to address the problem; the number of websites distributing the malware is increasing at an troubling rate and is getting out of control. Most of the malware comes into the system while downloading files over Internet. Once the malicious software finds its way into the system, it scans for vulnerabilities of operating system and perform unintentional actions on the system finally slowing down the performance of the system. Malware has capacity to infect other executable code, data/system files, boot partitions of drives, and generate excessive traffic on network leading to denial of service. When user runs the infected file it becomes resident in memory and infect any other file executed afterwards. If operating system has susceptibility, malware can also take control of system and infect other Systems on network. Such malicious programs are also known as parasites and destructively affect the performance of machine usually resulting in slow-down. Some malware are very easy to notice and remove by using antivirus software's. These antivirus software maintains a repository of virus signatures i.e., binary pattern characteristic of malicious code. Files suspected to be infected are verified for presence of any virus signatures. This method of detection works well until the malware writer started writing polymorphic and metamorphic malware. These variant of malware avoid detection through use of encryption techniques to prevent signature based detection. Security products such as virus scanners look for characteristics byte sequence (signature) to recognize malicious code. The eminence of the detector is decided by the techniques used for detection. A good malware detection technique must be able to identify malicious code that is hidden or embedded in the original program and should have some capability for detection of so far mysterious malwares. Commercial virus scanners have very low resistance to new attacks because malware writers continuously make use of new obfuscation methods so that the malware could evade detections.

Manuscript received on April, 2016.

***B.Rajesh**, Computer Science and Engineering, G. Pulla Reddy Engineering College, Kurnool, India, 9966383392.*

***Y.R.Janardhan Reddy**, Computer Science and Engineering, G. Pulla Reddy Engineering College, Kurnool, India, 9966545271.*

***C.V.Chakradhar**, Computer Science and Engineering, G. Pulla Reddy Engineering College, Kurnool, India, 9963039299.*

II. MALWARE DETECTION

A. Malware Detector

A Malware detector (D) is describe as a function whose domain and range are the set of executable program (P) and the set {malicious, benign} Malware detector can be described as

D(p) = { Malicious (if p contains malicious code),
Benign (otherwise)}

The detector scans the program 'p' ε P to check whether a program is benign program or malicious program. The main aim of testing is to find out false negative, false positive, hit ratio. Malware detector uncovers the malwares Based on their signatures the binary pattern of the machine code of a particular virus is called as signature. Antivirus programs evaluate their database of virus signatures with the files on the hard disk and detachable media (together with the boot sectors of the disks) as well as within RAM.

The antivirus dealer updates the signatures regularly and makes them accessible to customers by means of the Web.

1) False positive: false positive occurs while a virus scanner wrongly detects a 'virus' in a non-infected file. False positives result when the signature used to detect a particular

virus is not unique to the virus - i.e. the same signature appears illegal, non-infected software.

2) False negative: false negative occurs while a virus scanner fails to detect a virus in an infected file. The antivirus scanner may fail to detect the virus since the virus is new and no signature is yet accessible.

3) Hit ratio: hit ratio occurs when a malware detector detects the malware. This occurs because the signature of malware matches with the signatures stored in the signature databases.

III. Malware Detection Techniques

Malwares can be detected by using two approaches namely Signature based malware detection and Behavior based malware detection.

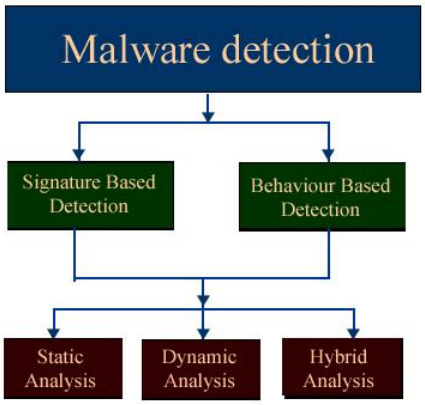

Fig: Malware detection and Analysis

A. *Signature-Based Malware Detection Technique*

Most of the industrial antivirus scanners check for signatures which are usually a series of bytes inside the malware code to state that the program scanned is malicious in nature. Fundamentally there are three kinds of malwares: basic, polymorphic, metamorphic malwares. In basic malware the program entry point is altered such that the control is transferred to malicious payload. Detection is relatively if the signature can be found for the viral code.

Fig: show basic malware.

Polymorphic viruses mutates as keeping the original code in one piece. A polymorphic malware consists of encrypted malicious code along with the decryption component. To enable the polymorphic virus the virus has got polymorphic engine someplace in the virus body. The polymorphic engine generates new mutants each time it is executed. Signature based detection for such a virus is complicated because for each variant new signature is generated which makes signatures based detection complex. Strong static analysis based on API sequencing is used for polymorphic virus detection.[9]

Fig: shows polymorphic malware's

Metamorphic malware be able to reprogram itself using certain obfuscation techniques so that the children never look like the parent.[10] Such malwares elude the detections from the malware detector since each new variant generated will have dissimilar signature, for this reason it is impossible to store the signatures of several variants of same malware sample. In order to prevent detection a metamorphic engine has to be implemented with some sort of disassemble in order to parse the input code. As soon as disassembly, the engine will transform the program code and will produce new code that will maintain its functionality and would still look different from the original code.

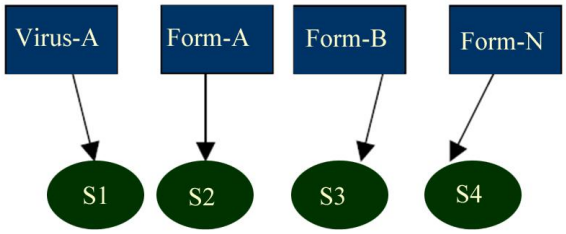

Fig: shows metamorphic malware and multiple signatures for multiple variants.

Let S1, S2, S3, S4 be the set of malware signatures. For the above figure, Si Є S are signatures of metamorphic variant belong to single metamorphic sample 'M'. The main problems with the signature based detection technique are as follows:

- Signature extraction and distribution is a complicated task.
- The signature generation involves manual involvement and requires rigorous code analysis.
- The signatures can be easily bypassed as and when new signatures are produced.
- The size of signature repository keeps on increasing at an alarming rate.

B. *Behavior -based Detection Technique*

Behavior based detection [3] differs from the exterior scanning method in that it identifies the action performed malware rather than the binary pattern. The programs with different syntax's but having same behavior are collected; as a result this single behavior signature can identify various samples of malware. This types of detection mechanisms helps in detecting the malwares which keeps on generating new mutants as they will continuously use the system resources and services in the similar way. The behavior detector [11][12]. Fundamentally consists of following components which are as follows:

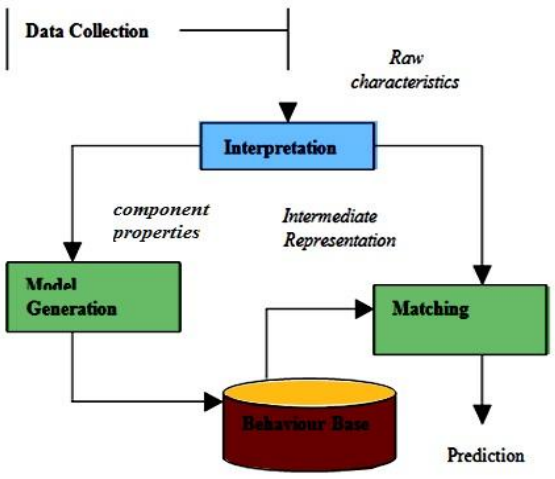

Fig: Malware behavior detector

- *Data Collection: Data collector* collects the dynamic / static information's are captured.
- *Interpretation: Interpretation* converts the raw information collected by data collection module into intermediate representations.
- *Matching Algorithm:* Matching Algorithm is used to compare the representation with the behavior signature.

IV. Malware Analysis Techniques

The detection process accompanied by signature based and behavior based techniques which further accomplished the whole process with static, dynamic and hybrid analysis.

Malware analysis: Malware analysis is the course of forming the reason and usefulness of a given malware samples, for example, an virus, worm, or Trojan horse. This procedure is an essential stride to have the capacity to create detection techniques for malicious code. Moreover, it is a critical essential for the improvement of evacuation devices that can completely erase malware from a contaminated machine. Generally, malware analysis has been a manual process that is dull and time-concentrated. Unfortunately, the quantity of tests that should be analyzed by security sellers every day is continually expanding. The procedure of be analyzing a given program during execution is called dynamic analysis; while static analysis alludes to all systems that investigate a project by assessing it. Static and Dynamic analysis is depicted in point of interest in next areas.

A. Static Malware Analysis

Static analysis techniques can be applied on different kinds of software's. Examining the software without running it is called static analysis. If source code is accessible the static analysis tools will assist you in finding memory corruption flaws and show the accuracy of the models for a given system.[1] We can also represent the software in binary format by using Static analysis tools. If we compile the source code by executing in binary format, there is a chance of losing some information. Which will cause additional risk in analyzing the code? Binary code is inspected manually without executing it The procedure of examining a given binary code without executing it is for the most part led physically. For instance, if the source code is accessible a few attractive data, for example, data structures, utilized capacities and call graphs can be extricated. This data gets lost once the source code has been compiled into a binary executable and accordingly obstructs further investigation. Inside of the malware area regularly the last is the situation, since the source code of a current malware double is commonly not accessible.[2] Different techniques are used for static malware analysis. Some of those are described below.

- *File fingerprinting:* By examining clear external features of the binary this includes operations on the file level such as computation of a cryptographic hash (example: massage digest5) of the binary in order to differentiate it from others and to confirm that it has not been modified.
- *Extraction of hard coded strings:* Software naturally prints output (e.g., status- or error-messages), which end up embedded in the compiled binary as understandable text. Examining these embedded strings often allows conclusions to be drawn on the subject of internals of the inspected binary.
- *File format:* through leveraging metadata of a given file format extra, useful information can be gathered. This includes the magic number on UNIX systems to decide the file type as well as dissecting information of the file format itself. For example from a Windows binary, which is typically in Portable Executable Format a batch of information can be extracted, such as compilation time, imported and exported functions as well as strings, menus and icons.
- *AV scanning:* If the examined binary is well-known malware it is highly likely to be detected by one or moreAV scanners. To use one or more AV scanner is time consuming but it becomes necessity from time to time.
- *Packer detection:* These days malware is typically scattered and harder to understand it is in an obfuscate form e.g., encrypted or compressed. This is achieved using a packer, whereas arbitrary algorithms can be used for adjustment. After packing the program looks very much different from a static analysis viewpoint and its logic as well as other metadata is thus hard to recover. While there are certain unpackers, such as PEiD2, there is accordingly no generic unpacker, building this a major challenge of static malware analysis.
- *Disassembly:* The major part of static analysis is typically the disassembly of a given binary. This is conducted utilizing tools, which are capable of reversing the machine code to assembly language, such as IDA Pro. Based on the reconstructed assembly code an analyst can then inspect the program logic and thus examine its intention. Usually this process is supported by debugging tools such as OllyDbg.

The main advantage of static malware analysis is that it allows a complete analysis of a given binary. That is, it can cover all possible execution paths of a malware sample. Additionally, static analysis is generally safer than dynamic analysis as the source code is not actually executed. However, it can be extremely time-consuming, cumbersome and thus requires expertise.

Static Malware Analysis Limitations

In general, the source code of malware samples is not quickly available. That reduces the valid static analysis techniques for malware analysis to those that retrieve the information from the binary representation of the malware. Analyzing binaries brings along complicated challenges. Consider, for instance, that most malware attacks hosts executing instructions in the IA32 instruction set. The disassembly of such programs may result in unclear results if the binary employs self modifying code techniques. Moreover, malware relying on values that cannot be statically determined (e.g., current system date, indirect jump instructions) exacerbate the application of static analysis techniques. The other is that malware authors know of the limitations of static analysis methods and hence, will likely create malware instances that make use of these techniques to put a stop to static analysis. Therefore, it is necessary to develop analysis techniques that are flexible to such modifications, and are able to reliably analyze malicious software.

B. Dynamic Malware Analysis

Executing a known malware sample within a controlled atmosphere and monitoring its actions in order to examine the malicious actions is called dynamic malware analysis. Dynamic Malware Analysis is done by inspecting and logging the behavior of the malware while running on the host. Virtual machines and Sandboxes are widely used for this type of analysis. The malware is debugged while running using a debugger such as GDB or WinDbg to watch the behavior of the malware step by step while its instructions are being processed by the processor and their live effects on RAM. Because Dynamic Malware Analysis is performed during runtime and malware unpacks itself, dynamic malware analysis avoids the restrictions of static analysis i.e., unpacking and obfuscation issues. Thus it is easy to see the real behavior of a program. Another key advantage is that it can be automated thus enabling analysis at a large scale basis. On the other hand, the main disadvantage is so-called dormant code: That is, unlike static analysis, dynamic analysis commonly monitors only one execution path and thus suffers from incomplete code coverage. Likewise there is the risk of hurting outsider frameworks, if the examination environment is not appropriately disengaged or limited separately. In addition, malware samples may change their behavior or stop executing at all once they detect to be executed within a controlled analysis environment.[2]

Mainly two basic approaches for dynamic malware analysis can be distinguished:

- *Analyzing the difference between defined points:* A given malware sample is executed for a definite period of time and then the modifications made to the system are analyzed by comparison to the early system state. In this approach, Comparison description states behaviour of malware.[2]
- *Observing runtime-behavior:* In this method, malicious activities launched by the malicious application are monitored throughout runtime using a dedicated tool.

An instance for the first approach is Truman (The Reusable Unidentified Malware Analysis Net). Thereby malware is executed on a real Windows environment rather than within a Virtual Machine. During runtime Truman provides a virtual Internet for the malware to interact with. After execution the host is restarted and boots a Linux image, which then mounts the earlier used Windows image in order to extract the related data, such as the Windows registry and a complete file list. Finally the Windows environment is reset to its original clean state. By using a native environment Truman is able to avoid possible anti-debugging measures of malware. However, since the result is only a snapshot of the infected system, information associated to dynamic activities such as spawned processes and temporarily produced files are lost. Therefore observing the runtime-behavior of an application is currently the most promising approach. It is mostly conducted utilizing sandboxing. A sandbox hereby refers to a constrained runtime environment which is partitioned from the rest of the system in order to separate the malicious process. This partitioning is naturally achieved using virtualization mechanisms on a definite level. While in principle existing tools, such as cheroot could be used to deploy such a controlled environment many sandbox environments devoted to malware analysis exist implementing specialized procedures.

C. Hybrid Analysis

The combinational approach of both static analysis and dynamic analysis is called Hybrid analysis. Hybrid analysis begins With analyzing signature specification of any malware program then connect it with the other behavioral parameters For development of complete analysis. With this approach we can overcome the limitations of both static and dynamic analysis..

D. Malware Analysis Tools

It is an outline of the already existing approaches and tools that make use of the accessible techniques to analyze Recognized and potentially malicious software. The analysis information generated by the tools in this section give analyst Valuable insights into actions performed by a sample. This knowledge is necessary for developing countermeasures in a timely manner.

- *Anubis:* The analysis of unidentified binaries project is based on TTAnalyze. Anubis executes the sample under analysis in an emulated atmosphere consisting of a Windows XP operating system running as the guest in

Qemu. The analysis is performed by examining the invocation of Windows API functions, as well as system service calls to the Windows Native API. In addition, the parameters passed to these functions are examined and tracked.

- *CWSandbox:* CWSandbox, created by Carsten Willems executes the sample under analysis either natively or else in virtual Windows environment. The analysis functionality is implemented by hook functions that perform the monitoring of API level. Additionally, monitoring of the system call interface is implemented. The system is intended to capture the behavior of malicious software samples with respect to file system and registry exploitation, network communication, and operating system interaction. The virtual system is a full installation of a Win32 class operating system under which the sample under analysis is executed together with the analysis components. [6]
- *JoeBox:* During the dynamic analysis of a potentially malicious sample, Joebox [Buehlmann and Liebchen] creates a log that contains high level information of the performed actions regarding file system, registry, and system activities. joebox is specially designed to run on real hardware, and not to rely on any virtualization technique. The system is designed as a client server model where a single controller instance can manage multiple clients that are responsible for performing the analysis. Thus, it is instantly forward to increase the throughput of the complete system by adding more analyzing clients to the system. All analysis data is collected by the controlling machine.[5] [8]
- *Norman Sandbox:* The Norman Sandbox is a dynamic malware analysis resolution which executes the sample in a tightly-controlled virtual environment that simulates a Windows operating system. This environment is used to simulate a host computer as well as an attached local area network and, to some level, Internet connectivity. The basic idea behind the Norman Sandbox is to restore all functionality that is required by an analyzed sample with a simulated version thereof. The simulated system thus has to provide support for operating system applicable mechanisms such as memory protection and multi-threading support. Moreover, all required APIs have to be present to give the sample the fake impression that it is running on a real system. Because the malware is executed in a simulated system, packed or obfuscated executables do not hinder the analysis itself. [7] Norman Sandbox focuses on the detection of worms that spread via email or P2P networks, as well as viruses that to duplicate over network shares. In addition, a generic malware detection technique tries to capture other malicious software. [7]

V. Conclusion

This paper explains the different techniques that are used for malware detection like signature based detection and behavior based detection and also the limitations associated with them. This paper also shows different malware analysis methods like static, dynamic and hybrid analysis methods for analyzing the malwares. By studying this paper we will identify which method is effective for malware detection and which analysis method is preferable.

REFERENCES

[1] https://en.wikipedia.org/wiki/Malware
[2] A Survey on Automated Dynamic Malware Analysis Techniques and tools, http://www.seclab.tuwien.ac.at/papers/malware_survey.pdf
[3] MasterThesis-MartinBrunner, Integrated Honeypot Based Malware Collection and Analysis http://www.martinbrunner.net/doc/MasterThesis-MartinBrunner.pdf
[4] A Malware Instruction Set for Behavior-Based Analysis,
[5] Anubis. Analysis of unknown binaries. http://anubis.iseclab.org
[6] Toward automated dynamic malware analysis using CWSandbox. http://dl.acm.org/citation.cfm?id=1262675
[7] Norman SandBox Whitepaper. http://download.norman.no/whitepapers/whitepaper Norman SandBox.pdf
[8] Joebox: a secure sandbox application for Windows to analyse the behaviour of malware. http://www.joebox.org/
[9] DataRescue,Inc. the ida pro disassemble www.datarescue.com/idabase,2006
[10] Arun Lakhotia , Aditya Kapoor , Eric Uday , "Are Metamorphic Viruses Really Invincible ? Part 2" , Virus Bulletin ,January 2005.
[11] Greoigre Jacob,Herve Debar,Eric Fillol,"Behavioral detection of malware:from a survey towards an established taxonomy",Springer-Verlag France 2008
[12] Gerard Wagener,Radu State,Alexandre Dulaunoy,"Malware Behavior Analysis",Springer-Verlag France 2007.

B. Rajesh received the M .Tech Degree from S.K. University, Ananthapuramu in 2012. Area of interest: Network security and Cloud computing.

Y.R. Janardhan Reddy received the M .Tech Degree from S.K. University, Ananthapuramu in 2012. Area of interest: Image Processing and Cloud computing.

C.V. Chakradhar received the M .Tech Degree from J.N.T University, Ananthapuramu in 2013. Area of interest: Artificial intelligence and Character recognition.