An Infrastructure for Detecting Malware

Habeeb P
Sullamussalam Science College
Areacode

ABSTRACT

A malware is a program that has a malicious intent. Nowadays, attack from malwares is rising in alarming fashion and thousands of malwares are injected to the Internet. Malware authors use many techniques like obfuscation and packing to avoid detection. A number of techniques for malware detection are available and none of them able to detect all types of malwares. In this paper, a more efficient malware detection framework is presented. This framework utilizes the ability of sandbox to analyze files in an isolated environment. A group of sandbox is arranged parallel and process each incoming file from the Internet to internal network. A credit is assigned to each operation made by the file under inspection. Report generated by each sandbox is converted into a general intermediate format. Average credit of a specific file is calculated based on average credit from individual reports. Files are classified as malicious or benign based on this final average credit. This system increases the efficiency of malware detection by using multiple dynamic analysis technics.

General Terms

Security, malware detection, malware analysis, sandbox, advanced persistent threat.

Keywords

Malware detection and analysis, sandbox, apt malwares.

1. INTRODUCTION

Attacks from malwares are rising in alarming fashion. Each day thousands of malwares are injected into the network world. In early time malwares are written by hobbyist and programmers for fun but these days malware authors are mainly focused on profit. For example, an early malware named “brain”, tailored by two programmers Bazith and Amjad from Pakistan, just affect a boot sector of a floppy drive. However, recent disclosure of stuxnet malware that affects industrial control system shown us the ability of current malwares and changes in the domain of target. A new era of cyber-attacks is emerging: moving from the traditional viruses to sophisticated attacks like the advanced persistent threat (APT). Internet users need some techniques to protect from these threats.

Security vendors offer some tools that aim to detect and report malicious files. Most of the security tools use signature matching to detect malwares. These techniques required the vendor to provide a database of signatures. Signatures are compared against files under inspection in order to check whether malicious or not. Security vendors obtain malware samples by using techniques like honeypot. Once a sample of malware obtained to analysis, the first step is for a human analyst to determine whether this sample poses a threat to users by analyzing the sample. If the sample identified as the malicious, then analyst tries to create a pattern or signature that allows to identify this sample. The analysis of the file and generation of signature by human is very time consuming while compare to the malware generation rate. It furthermore, may be the generated signature is a false one due to human fault.

It is not extraordinary for an anti-virus vendor to receive thousands of unknown malware samples per day. Panda labs [2] report says that 15 million new malware samples were generated, at an average rate of 160,000 every day. This extensive quantity requires an automated approach to detect and analyze. Two ways to perform automated analysis: dynamic analysis and static analysis. Dynamic malware analysis log and watch malware while running on the system. Sandboxes and virtual machines are commonly used for dynamic malware analysis. Static malware analysis is done by extracting static details available in the file. Reverse engineering is commonly used to read code from an executable.

Most of the antivirus techniques are failed to detect targeted attacks and advanced persistent threat. A targeted attack is one that has been targeted a specific company or organization or a specific person. These attacks are not detectable by using traditional antimalware techniques. Advanced persistent threat has three steps; initially, the attacker thoroughly studies the target by using several techniques like social engineering. Once enough details are collected in reconnaissance phase, then the attacker tries to find out vulnerabilities to exploit. Exploitation of vulnerability is the third step. One of the main techniques used by attackers in order to get access to a particular organization is spear phishing, by which the attacker sent mail, with malicious attachment or links that moves target to vulnerable services or vulnerable websites. Compromised systems are used to get persistent access to network. Cost of data breach cause by this type of attack is thousands of dollars.

In this paper, I propose a framework for detection and analysis of malware. The proposed system uses a plurality of sandbox technologies. Motivation behind this system is that, a single antimalware technology is not enough to detect all malwares and if a group of antimalware techniques are used simultaneously, then the detection rate will increase dramatically. The proposed system uses both static and dynamic malware analysis techniques.

The rest of this paper is ordered as follows. Section II provides a brief description on what is a malware and types of malwares. An overview of sandbox technology is also given in second section. Section III discuss existing techniques for malware detection and analysis. Section IV provide detailed description of the proposed system. Section V concludes this paper.

2. MALWARE AND SANDBOX

Any software that does something that causes harm to a user, computer, or network can be considered malware [1], including spyware, viruses, worms, Trojan horses and rootkits. Malware analysis [1] is the process of dissecting malware to understand how it works, how to identify it, and how to defeat or eliminate it. The following section briefly explains different types of malware.

Spyware: Software that collects information from another computer and transfers this information to the attacker is known as spyware. Information that might be interesting for the attacker includes contents of documents and emails, history of visited web pages and bank account credentials. Some spywares are capable of record and transmit key strokes of a victim. The presence of malware is normally hidden from the victim and difficult to detect.

Virus: a virus is a piece a program that adds itself to other programs. A virus cannot run independently – it requires that its host program be run to activate it. Viruses perform some harmful activities on infected hosts.

Worms: a worm is a program that replicates itself in order to spread to other systems. A worm can run independently and can propagate a fully working version of itself to other computers. The Morris Worm is the first publicly known instance of a program that exposes worm like behavior on the Internet.

Trojan horse: a Trojan horse is a program that appears to be useful or safe, but in reality, it performs malicious actions in the background. While a Trojan horse can disguise itself as any legitimate program, usually they pretend to be useful browser plug-ins, screensavers, or downloadable games. Once installed, their malicious part might download additional malware, modify system settings or infect other files on the system. Trojan horses are one of the common methods criminal uses to infect target's computer and collect sensitive information.

Rootkit: rootkit is a type of malware that designed to hide the existence of certain processes or programs from normal methods of detection and enable continued privileged access to a system. Rootkits are normally activated before the actual completion of Operating system loading.

Sandbox is a security mechanism for separating running programs. Sandboxes run programs in an isolated environment and provide as many privileges it can. Sandboxes are normally used dynamic analysis of programs under consideration of bad behavior.

A sandbox is a tightly controlled environment where programs can be run. Sandboxes restrict what a piece of code can do, giving it just as many permissions as it needs without adding additional permissions that could be abused.

3. EXISTING TECHNIQUES

Malware detectors implement some malware detection techniques. The malware detection can be divided into two broad categories; signature based system, and behavior based systems. Malware detectors normally take two inputs; one input is the knowledge of the malicious behavior. If the malware detector works based on a signature based, then it`s knowledge base is a collection of signatures produced manually. Behavior based system has a rule set or some specification of what is malicious in-order to decide whether a program under inspection is valid or not. The other input malware detector takes as input is the file under assessment. Signature based systems try to create a signature of the file under inspection by using a pre-agreed algorithm. Generated signature compares with the existing database to determine whether the file under inspection is malicious or not. Behavior based system executes the program under inspection and monitor activities of the file. The activities are then compared with rule set to identify malicious behavior. The remainder of this section provides an overview of some available malware detection techniques.

Steven A. Hofmeyr et al. [3] introduced a method for detecting maliciousness of a program by monitoring system calls made by the file under inspection. A rule set of normal behaviors must need to develop before starting the actual inspection. Here normal behavior is specified in terms of system calls. Hamming distance is used to find out the deviation of a system call sequence with another. A threshold or critical value must be specified to determine whether a process is malicious or not. Processes with large Hamming distance are considered as malicious. Authors are able to detect intrusion attempts that exploits various UNIX programs like lpr, sendmail and ftpd.

N. L. Petroni [6] present a technique to detect kernel integrity violation by monitoring system with a software. The monitoring software run on a separate PCI card and take snapshots of the system memory periodically. The grabbed memory copy is analyzed to determine integrity violation. Later in [7] a specification based integrity checker is proposed. This system can examine the integrity of dynamic kernel data. These systems only grab snapshots of volatile memory.

Wei-Jen Li et al. [4] introduced Fileprints analysis for detecting malicious files. During the training phase, a specification or a set of specifications is derived based on the byte composition of the files. These specifications are used to categorize various file's types the system intends to handle. For example, benign .exe files have a unique distribution of bytes that is entirely different from the byte distribution of .doc or .pdf files. During the inspection files with large deviation from the given specification is considered as malicious. Author’s technique showed detection rates between 72.1 percent and 94.5 percent for PDF file that had embedded malware. But, the technique failed to detect some malicious file with embedded code and authors believe that 2-gram or 3-gram approaches may increase the detection rate.

Yi-Min Wang et al. [5] proposed a GhostBuster system that detects malicious files by the view comparison method. Their method compares two system scans- one internal scan and an external clean scan. The external scan is performed by restarting the system being examined with clean Operating System. The internal scan is otherwise, with execution of files under inspection. The clean boot or external scan destroys all non-persistent states, and it reduces the detection rate of the system.

In [8] Kreibich and Crowcroft introduced a honeycomb system. In honeycomb, honeypots are used to detect malware stemming from network traffic. Honeycomb generate signatures from detected malicious files. Honeycomb works on the assumption that traffic which stopped at a honeypot is suspicious. Each connection information store in a database and honeycomb use anomaly in the connection streams to create signatures.

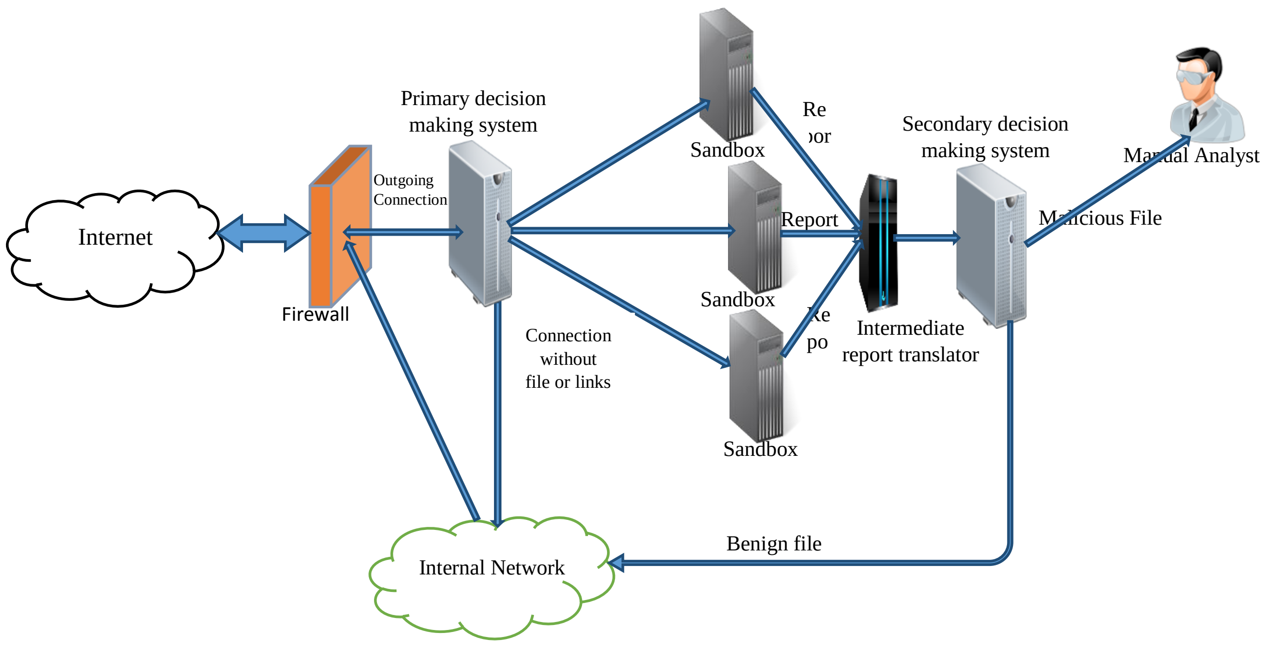

Fig 1: Overview of the proposed system

4. PROPOSED SYSTEM

Proposed system uses a group of sandboxes, and they are configured to process in parallel. Data from the Internet to internal network initially pass through a firewall for blocking unauthorized connections. Firewall direct successfully scanned data to an initial decision making server. Decision making server collect stream of data and analyze to determine whether it contains files or links. Connections without any files or links forward to internal network. Fig: 1 depicted an overview of the system.

Data for further processing are put into a waiting queue by the initial decision making server. A unique id is assigned to each file or link under consideration. Sandbox has also had a waiting queue. Initial server place a copy of file to waiting queue of sandbox. Sandbox executes files from the queue in a first-come-first-serve order and generates a report of processing. Reports are signed with unique id of the processed file and then forward to the intermediate report translator (IRT). IRT converts received reports into a general format and transfer to the secondary decision making system (DMS).

Secondary DMS finds out credit of each operation specified in the report from the rule set database. Rule set database contain tables with operations and corresponding credits. The values of credit assigned for operations are directly proportional to the importance of the operation. Once credit for each operation assigned then average credit of an entire report is calculated by using following equation.

$$Average\ Credit\ of\ a\ report\ (CRw) = \frac{Sum\ of\ the\ credit\ of\ individual\ operation}{Number\ of\ opeations}$$

$$CRw = \frac{\sum_{i=0}^{n} Cr(OP_i)}{n}$$

$wherer\ Cr(OP)\ is\ a\ function\ to\ get\ credit\ of\ operation$

$OP\ and\ \ n\ is\ the\ total\ number\ of\ operations$

Secondary DMS combines average credit from files with same signature id to find out the final credit of file under inspection. The below given equation is used to calculate average credit of a file.

$$Final\ Credit\ of\ the\ file\ (CRf) = \frac{Sum\ of\ the\ average\ credit\ of\ reports}{Number\ of\ reports}$$

$$CRf = \frac{\sum_{i=0}^{N} CRw_i}{N}$$

$wherer\ N\ is\ the\ total\ number\ of\ reports$

The final credit of the file (CRf) is compared with predefined critical credit (CRc). If the CRf is greater than CRc then the file is determined as malicious and selected for manual analysis. If the CRf is less than or equal to CRc then the file is benign and forward to internal network. Malicious files are forward to expert malware analyst for dissecting it. Fig: 2 depicts the data flow of in the proposed system.

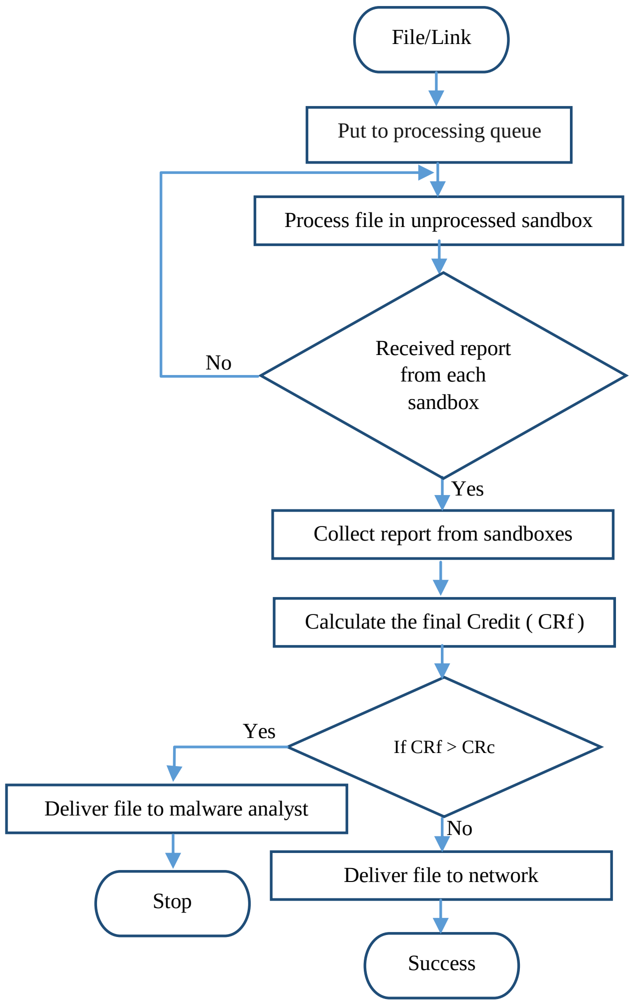
File/Link
Put to processing queue
Process file in unprocessed sandbox
No
Received report from each sandbox
Yes
Collect report from sandboxes
Calculate the final Credit (CRf)
Yes
If CRf > CRc
Deliver file to malware analyst
No
Deliver file to network
Stop
Success

Fig 1: Data flow diagram

Proposed sandbox based system uses both static malware analysis and dynamic analysis. Which improve the detection rate of the malicious file. In this system, each files are processed by each sandbox, and generate a unique report. This multiple scanning also contributes to increased efficiency of the system. Parallel running of sandboxes provides fast processing of files and report generation.

5. CONCLUSION

An efficient and effective malware analysis and detection system is still not available, and malware generation rate skyrocketed in last years. This paper proposed an overview of a malware detection framework by using a group of parallel working sandboxes. The detection rate of the malware definitely increased by this framework. Several features like parallel processing and intermediate report translation improved the quality and efficiency of the system. Actual implementation of this framework is kept as my future work.

6. REFERENCES

[1] Sikorski, Michael, and Andrew Honig. *Practical Malware Analysis: The Hands-On Guide to Dissecting* Malicious Software. No Starch Press, 2012.

[2] Quarterly report – Panda Security, http://mediacenter.pandasecurity.com/mediacenter/wp-content/uploads/2014/07/Quarterly-Report-Q2-2014.pdf, December 2014

[3] S. Hofmeyr, S. Forrest, and A. Somayaji. Intrusion detection using sequences of system calls. Journal of Computer Security, pages 151 – 180.

[4] W. Li, K.Wang, S. Stolfo, and B. Herzog. Fileprints: Identifying file types by n-gram analysis. 6th IEEE Information Assurance Workshop, June 2005.

[5] Y.-M. Wang, D. Beck, B. Vo, R. Roussev, and C. Verbowski. Detecting Stealth Software with Strider GhostBuster. Proc. of the 2005 International Conference on Dependable Systems and Networks, June 2005.

[6] N. L. Petroni, T. Fraser, J. Molina, and W. A. Arbaugh. Copilot - a Coprocessor-based Kernel Runtime Integrity Monitor. Proc. of the 13th USENIX Security Symposium, Aug. 2004.

[7] N. L. Petroni, T. Fraser, A. Walters, and W. Arbaugh. An Architecture for Specification-Based Detection of Semantic Integrity Violations in Kernel Dynamic Data. Proc. of the 15th USENIX Security Symposium, Aug. 2006.

[8] C. Kreibich and J. Crowcroft. Honeycomb – creating intrustion detection signatures using honeypots. In 2nd Workshop on Hot Topics in Network, 2003.