On Malware Characterization and Attack Classification

Alan Lee Vijay Varadharajan Udaya Tupakula
Information and Networked Systems Security Research
Macquarie University, Australia
{vijay.varadharajan, udaya.tupakula}@mq.edu.au

Abstract

Malware is one of the significant problems in the current Internet. Often security tool vendors develop an attack signature to deal with the attacks. However attack techniques such as polymorphism and metamorphism can be used by the attacker to generate multiple variants of the malware and complicate the signature identification. In this paper we present our analysis on sample set of malware and then discuss how MAEC's taxonomy can help to address the malware problem.

Keywords: Malware, Software Security, MAEC framework.

1 Introduction

The size and complexity of the current operating systems and applications is continuously increasing and it is not an easy task to ensure the secure operation of such systems. Although there are several tools such as intrusion detection systems, honey pots, and antivirus the dynamic nature of the attacks makes it difficult to detect and prevent attacks. On the other hand, it is simple task for the attacker to compromise such systems and generate different types of attacks. As a result we are witnessing an increasing number of zero day attacks on a daily basis. Zero day attacks are the attacks which are previously not known. In order to deal with the malware the security tool vendors analyze the malware and develop attack signatures to deal with the malware. However, in most of the cases the analysis is done manually and this requires considerable time before a signature can be developed for malware. In addition to this, automated tools such as ADMmutate enable the attacker to automatically generate variations to the malware for each infection; making it extremely difficult for the security professionals to identify and develop suitable attack signatures.

Our analysis confirms there is no systematic approach to deal with the malware problem. Without a systematic approach, there will be an ever increasing gap with the malware attacks and the tools available to deal with the attacks. Recently Malware Attribute Enumeration and Characterization (MAEC) framework [Mitre, 2012] has been proposed for unified classification of the malware. In this paper we present our analysis on different types of malware and make use of MAEC framework for characterising and classifying malware. We have made several interesting observations and also provide some inputs for improving the MAEC framework.

The paper is organized as follows. Section II presents some of the challenges with the malware and an overview of the MAEC framework. Section III presents our analysis on different types of malware, use of MAEC framework for the malware and some inputs for improving the MAEC framework. Section IV concludes.

2 Background

In this section we will present some of the challenges related to the malware problem in the Internet and overview of the MAEC framework.

2.1 Malware Challenges

Massive Volumes of Malware Samples: Security tool vendors and malware researchers are constantly researching and analysing malware in order to create detection and remediation to protect their customers and their computers. In the days where malware growth has been pretty constant, malware researchers were able to analyse and produce signatures for malware detections for most of the malware samples. Due to the massive malware growth in the past few years, it has been increasingly difficult for malware researchers to analyse (be it dynamic or static) every malware samples that they encounter. Malware researcher needs a system of classifying malware so that malware researchers can rely on to help them tackle the massive volume of malware quickly.

Automated Analysis Tools: There has been a huge volume of malware that malware researchers need to analyse, automated dynamic analysis tools have been developed to assist malware researchers to speed up the process of malware analysis. Recently automated analysis using virtualisation is of considerable research interest. Also malware researchers occasionally submit malware samples to various automated analysis tools to obtain some information with regards to the malware sample. However, each of these automated analysis tools have their own way of formatting information and classification. There is no uniform way of presenting malware information as different automated analysis tools produces different kinds of reports. Therefore, this posts a problem for malware researchers as they have to sift through various reports to obtain useful malware information. Furthermore this results in duplication of research hindering progress on the malware analysis. For example, Vigilante [Costa, 2005] is a collaborative approach for the end to end detection/prevention of the worms. Each host runs specific software which captures the information regarding the exploit of the worm and distributes a Self Certifying Alert (SCA) to warn other

hosts regarding the spread of the worm. The end hosts use the information in the SCA to identify if it is vulnerable to the worm and apply a host based filter to prevent the worm. However note that in this case, the end hosts should be capable of identifying the malware from the SCA alerts. Otherwise it is not of much use for the end hosts.

Information Sharing: There are many security vendors and malware researchers in the IT security industry. Whenever there is a major outbreak of a particular malware, each security vendors and their malware researchers attempt to obtain malware samples and perform their own analysis. This occasionally results in duplication of malware research effort. There is also a lack of detailed information sharing amongst malware researchers. Most malware researchers are able to share bits and pieces of information as they go along discovering malware characteristics. Furthermore, malware researcher may refer malware characteristics differently as there has not been a widely accepted standard for unambiguously characterizing malware. Apart from the lack of standard for characterization, there is also no clear method of communicating malware information and their findings. As a result of this, there is non-interoperable and disparate malware reporting between security vendors.

Complexity of Malware: Malware infection has evolved overtime as malware writers are constantly adopting new concepts and ideas in making their malware more complex so that security products have problems detecting them and perform remediation on infected computers. Therefore, malware researchers require more time to analyse these malware. Nowadays, malware are usually not written by an individual. Instead, they are now written by a group of malware writers who usually collaborate through underground online forums, exchanging information and tips. Moreover, sophisticated malware often exhibits more than one characteristic. Therefore, it is not easy to classify malware into one single classification. Often, malware researchers may not agree on the classification of malware and this may lead to dispute among malware researchers, and also dispute between security vendors and companies or individual who has been alleged to creating malware.

2.2 MAEC Overview

Malware Attribute Enumeration and Characterization (MAEC) is an initiative developed by Mitre Corporation with the support of Department of Homeland Security (DHS) and National Institute of Science and Technology (NIST). One of the main aims of MAEC is to develop techniques to provide uniformity in malware reporting. This enables malware analysis results to be standardized and thus able to support the development of databases for storing malware characteristics. Also, one of MAEC's functions is to serve as a standard method of characterizing malware based on its behaviours, artifacts and attack patterns. As malware are evolving with obfuscation techniques such as polymorphism, encryption, packing etc. MAEC attempts to use malware characteristics to identify malware based on its distinct attributes instead of single signatures used in traditional security software.

Malware were usually classified in types (e.g. viruses, worms, backdoors etc.) based on its primary behaviour. However, recent malware outbreaks have exhibited multiple behaviours and malware researchers have been identifying the same malware as different types. This led to confusion about the true malware type and hinders accurate information being disseminated. Therefore, MAEC aims to permit accurate and unambiguous malware typing based on the malware attributes.

MAEC also aims to accurately identify new instances or variants of existing malware families using a minimum set of malware attributes that is sufficient of characterizing the malware family. For instance, given a particular malware family that has a particular set of malware attributes. If a new malware variant is discovered to exhibit the same set of malware attributes, we can accurately identify the new malware variant as belonging to the particular malware family by the means of MAEC characterization.

MAEC framework has three core components in its framework. This includes a MAEC enumeration (vocabulary), MAEC schema (grammar), and MAEC Cluster (standardized output). In the following subsections we will presents detail discussion on these components.

2.2.1 MAEC Enumeration

It comprise of the following three levels for describing malware attributes:

Low-level observables: these are the attributes and actions performed by the malware. These are usually system state changes made by malware on the system. Actions such as creating MUTEX in the Windows registry, and downloading a file from a remote server are recorded as low-level observables.

Mid-level behaviour: these are the consequences of the low level observables. The MAEC's language groups low-level observables into discrete clusters in order to define mid-level behaviours. It enables low-level observables to be clustered into a group. An action creating MUTEX will indicate a mid-level behaviour of preventing a second copy of the same malware from installing on the same machine

High-level taxonomy: this is the high level grouping of mid-level malware behaviours based on the underlying intention of malware writers. An example will be the behaviour of the malware being executed during every system start up will have a "Persistence" classification under the taxonomy. Other high level taxonomy described in MAEC includes Self Defense (malware's ability to protect against reverse engineering efforts) and Propagation (malware's capability to spread), to name a few.

2.2.2 MAEC Schema

It defines the syntax for the discrete MAEC language elements. It can be used as a baseline to create malware repositories or as a format for sharing information between repositories. The schema consists of the following three elements:-

Namespaces: Namespaces represents the grouping of malware behaviours, mechanism and other enumerated attributes into classes.

Properties: The schema will contain a set of properties that is applicable to the malware attributes and namespaces with specific properties for behaviours.

Relationships: MAEC schema includes an established set of files for defining the relationships among namespaces.

2.2.3 MAEC Cluster

Apart from MAEC's malware attribute enumeration and schema, MAEC will also define a standardized output for encompassing the attributes obtained from characterization of the malware. The MAEC cluster thus represents the standard output format of MAEC, encompassing any set of attributes and behaviours obtained from the characterization of malware instance. This will serve as a container for storing and sharing MAEC characterized information. Malware researchers can then share malware information in a standardized manner, therefore eliminating unnecessary confusion and speed up information sharing. Malware information can be output to a format like XML as it is human-readable as well as machine readable.

3 Malware Analysis and Characterisation

3.1.1 Approach

A size of about 5000 samples from different malware families have been gathered from online resources such as malwaredomainlist.com and one of the departments in university environment over a period of one month. After gathering the malware samples, each malware sample is executed in a controlled environment (e.g. virtual box and Xen). The aim of performing malware analysis on this set of different samples is to gather a small size of malware attributes and behaviour and establish the behaviours and attributes that are observed as belonging to malware and apply MAEC's taxonomy. The main focus of discussion in this paper is on the MAEC enumeration and MAEC schema.

The current attack detection and classification techniques can be classified as dynamic and static techniques. Our work makes use of both dynamic and static analysis techniques to detect the malware and analyse its behaviour. The dynamic analysis can be used to analyse the malicious behaviour during runtime and the static analysis techniques can be used to analyse behaviour without executing the code. Using system monitoring tools such as Installrite, regshot and InCtrl, logs are gathered and studied to derive low-level observables such as system state changes (registry keys and files), properties and propagation methods. In addition to the existing tools, we have developed some of the modules for analysing specific attacks. For example, since PDF documents have specific structure, we have developed the object extraction (ObEx) module to analyse the structure of the PDF documents and extract the information of the objects in the PDF document. The code extraction (CodEx) module is used to detect the code in the PDF document.

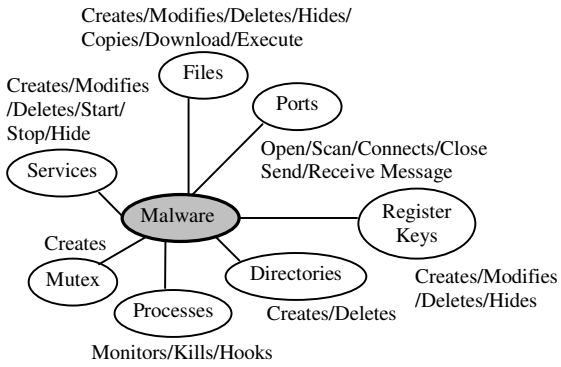

Figure 1: Common Malware Attributes

We have conducted a detail analysis and applied MAEC taxonomy on some of the attacks such as Koobface, malicious security tools and zbot. However due to space limitation we only present generic observations from our analysis. Figure 1 shows an overview of common malware actions and the objects that these actions are performed (attributes). The objects that malware performs actions upon are: file, registry keys, ports, directories, processes, mutex and services. The most common actions are creation, modification or deletion of files and registry keys. However, these actions are usually not malicious by itself. It is the intention behind these actions that determines the maliciousness of an action.

3.1.2 Malware Attributes

A typical malware when executed on a clean system will affect and perform some of the following actions (low-level observables) to some of following entities on the system at some point in time.

Malware are program that is capable of performing malicious actions. Therefore malware exists as a file - be it in a Portable Executable (PE) or a Portable Document Format (PDF) or any other format. Once executed, it is capable of affecting other files, registry keys, services, processes, ports, directories and memory.

Malware can create a new file on an infected system. This file varies in its purpose. Some files add more functionality to the malware whereas others may be dummy files created on the system to divert or thwart malware research efforts.

Registry on a Windows operating system is a database in which applications and system components store and retrieve configuration data. One of the most common registry key creations associated with malware is the run key(HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run). Malware will usually add a reference entry to this key so that the malware will be executed every time when Windows starts up. Although there are other methods to make sure that the malware is executed during Windows startup. Run key is one of the easiest and most popular methods used.

Another example of malware behaviour on registry keys is the modification of the Task Manager key (HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System\DisableTaskMgr).
Some malware tries to prevent infected victims from manually disabling the malware through Windows Task

Manager. It attempts to do so by modifying this key so that the Task Manager is disabled on the infected system.

Windows Services are used for creating long-running executable application that can be run in their own Windows sessions. Windows Services runs in the background and are usually used to providing support for other applications. Malware are known to deleting certain Windows services associated with security software so that the security software will no longer function properly.

Processes are executing programs. When a program is running, it creates a running process in memory. Some malware are capable of injecting malicious code in the running process so that the malware may monitor the process behaviour and trigger other malicious actions when necessary. This method is different from viruses whereby the malware made changes to the program directly. In processes injection, there is no actual code change done to the program. Therefore, if the process is terminated, the malicious behaviour stops as well.

Ports are used for originating connection to or receiving connection from a network entity. Malware can open ports to perform malicious actions. For example, malware uses port to connect to remote servers to download additional malware or sends stolen information to remote servers.

Directories are locations on the hard disk that allows users to group files into a common location. Malware sometimes create new directories as a holder to put all the files it needs.

MUTEX are usually created using the Windows Application Programming Interface (API) - CreateMUTEX. Their presence is usually to signal that the malware has been installed on the system. This can prevent another instance of the same software from executing if the software is designed to check for the existence of the MUTEX before commencing execution.

3.1.3 Malware Properties

In addition to the system changes discussed in the previous section, we have also identified some of the interesting properties of the malware. Listed below are some of the important properties of the malware:

Anti virtual machine: Malware sample that has anti virtual machine capabilities means that the malware sample will not execute and function as intended on a virtual machine environment. Most malware researchers perform malware analysis on a virtual machine environment due to its convenience and ability to mimic real machine behaviour. Some malware writer will therefore add anti virtual machine capabilities in the malware to prevent them being analyzed on virtual machines. There are many different ways in which this can be achieved. Some of the virtual machine detection methods include recognizing certain virtual machine processes or special values in the registers..

Anti debugging: Malware also uses anti debugging techniques to prevent malware researchers from tracing malware code through disassembly. The malware writer will include code that will detect if the malware sample is executed under a debugger. This can be achieved by using certain Windows API or detecting software breakpoints in the code etc.

Anti emulation: Code emulation is a technique used by security software to simulate CPU and memory management system to mimic code execution. The malware is simulated in a virtual machine of the security software and no actual code will be executed on the real system. Therefore code emulation is a very effective way of detecting malware. However, some malware writer use anti emulation techniques such as using structured exception handling or using long loops to prevent the malware from code emulation.

Encryption: Encryption is an obfuscation technique used by malware writers to prevent static analysis of malware by malware researchers. Encryption scrambles the malware code and any recognizable strings within it. Malware researchers will need to know the cipher key to decrypt the code to reveal the actual code for static analysis.

Packing: Packers are originally used by legitimate software vendors to encrypt and compress their software so as to reduce the size of the application and also protect their software from reverse engineers attempting to understand how the software works. However, malware writers adopted packers for the exact same purpose. Malware writers use packers to prevent malware researchers from static analysis and detection by security software. It also allows malware writers to create many different copies of malware samples as a small change in the malware source code will produce an entire different looking malware sample after going through a packer.

These properties are related to the malware sample rather than malware behaviours and characterization. Its intention is usually to thwart malware analysis attempts. Although these properties may not be malicious per se, it should somehow be used to add weight to the maliciousness of a malware sample.

3.1.4 Observations

Once the low-level observables research is completed, we began to associate these observables and determine with the respective mid-level behaviours. In this process, we were able to determine several common characteristics of malware behaviour that is seen in majority of malware.

Table 1 shows a list of common low-level observables and their associated mid-level behaviours. This is by no means an exhaustive list as it is not possible to list all malware attributes and behaviours.

Some of the attributes in Table 1, such as column 2, are common not just in malware but also legitimate software as this attribute exhibit the behaviour of automatic execution upon system startup. Some legitimate software such as security software exhibits this behaviour as well so that they can provide protection to the computer as soon as system starts up.

Other attributes such as column 1 has an associated malicious behaviour as its existence is to prevent execution of certain executable files.

Performing a study of common low-level observables and its associate malware behaviour helps us understand and establish behaviours and attributes that belong to malware.

	Low-Level observables	Mid-level behaviours
1	Creates key: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\avp.exe\Debugger = ""%UserProfile%\%UserName%1\winlogon.exe	Prevents file from execution
2	Adds key: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run] Bron-Spizaetus = ""%Windir%\ShellNew\bronstab.exe""	Enables automatic execution at System Startup
3	Modifies registry key: HKLM,"System\CurrentControlSet\Services\SharedAccess\Parameters\ FirewallPolicy\StandardProfile\AuthorizedApplications\List", "%startmenu%\Programs\Startup\ihaupd32.exe",0x00000000,"%startmenu%\Programs\Startup\ihaupd32.exe:*	Bypass firewall
4	Modifies registry key: HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Winlogon Value: Userinit = C:\Windows\System32\userinit.exe; C:\Windows\System32\sdra64.exe	Enable automatic execution at System Startup
5	Adds registry key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\bord_007	Enable automatic execution at System Startup (Windows Services)
6	Create folder: c:\directory\CyberGate	Instantiate malicious binary
7	Opens port: TCP8087	Establish connection
8	Modifies file: %System%\drivers\etc\hosts	Redirects websites
9	Downloads file: hxxp://vampizdecvsemnax.net/exe.exe	Download malicious binary from remote system
10	Creates file: C:\Windows\system\svchost.exe	Instantiate malicious binary
11	Creates MUTEX: @de#1?The Sh@de#1	Prevents 2nd installation of same malware
12	Deletes malware sample	Hiding trace

Table 1: Caption

3.1.5 Discussion

Our analysis of the malware examples and their characterization in terms of attributes and behaviours have resulted in several interesting observations. Listed below are some issues which were encountered during the process of mapping low-level observables to mid-level behaviours.

One of the issues during research and analysis was that some malware had undergone changes in terms of behaviour and the malware samples gathered may exhibit slight changes to its predecessors. Therefore, some of the MAEC characterization may not be a complete characterization of the malware.

There are some registry keys created by the malware that is classified as low-level observables. However, these registry keys do not reference to any binary files. Therefore, it may be a challenge to determine the real intention for creating the registry key and associate a mid-level behaviour to it. MAEC taxonomy does not explain how such cases should be handled.

Apart from the low-level observables like system state changes that a malware exhibits, there are properties such as encryption and packing that is associated to the malware samples. Although these properties are defined in MAEC schema, including properties information in mid-level behaviour could be helpful in identifying the maliciousness of the malware.

It can be foreseen that in future it is certain that new malware behaviours will surface and MAEC characterization must be able to handle and include these new malicious behaviours in their classification.

Malware has an execution flow that functions the way like normal software and some portion of code are run in sequence whereas others are run in parallels through multithreading. However, the behaviour structure does not exhibit any chronological characterization. Chronologically characterization may be useful in classifying malware as malware writers do not usually change their malware code and its execution flow for different malware variants. Therefore it will be good if MAEC taxonomy is able to incorporate chronological characterization in its output.

There are several instances where certain mid-level behaviour is achieved without any visible low-level observables like process injection etc. but such behaviour is visible in code or through 3rd party system monitoring tools. These code or observation may be suitable for inclusion in low-level observable as it adds uniqueness to malware identification. However, these do not appear to be used in MAEC characterization.

MAEC characterization does not seem to be able to determine which malware behaviour may be the dominant behaviour. This may cause issues in future whereby two or more malware researchers may disagree upon the primary malware behaviour. This may lead to different naming or malware typing and cause confusion to end users of security software.

4 Conclusion

In this paper we have conducted analysis on sample set of malware and applied MAEC taxonomy for malware. We have made several interesting observations and identified some inputs to improve the MAEC framework.

5 References

MAEC: Malware Attribute Enumeration and Characterization, Available at: http://maec.mitre.org/ Last Accessed 3 Nov 2012.

Manuel Costa, Jon Crowcroft, Miguel Castro, Antony Rowstron, Lidong Zhou, Lintao Zhang and Paul Barham, (2005): Vigilante: End-to-End containment of Internet Worms. Proceedings of the 20th ACM SOSP.