Analyzing the Makier Virus

Ritwik M[1] and Praveen K[2]

[1] TIFAC CORE in Cyber Security, Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India

[2] TIFAC CORE in Cyber Security, Amrita Vishwa Vidyapeetham, Coimbatore, Tamil Nadu, India

Abstract

Malware is most rampant in the modern era. Security professionals remain one step behind the attackers as they are reactive rather than proactive. This paper conducts an analysis of a recent malware strain henceforth called The Makier Virus.

Keywords: *Analysis, Computer Virus, Makier, Malware, Malware Analysis.*

1. Introduction

Shared resources, such as the Internet, have created a highly interconnected cyber infrastructure. Critical infrastructures in domains such as military, telecommunications, medical, power and finance are highly dependent on information systems. These two factors have exposed our critical infrastructures to malicious attacks and accidental failures. Disruption of services caused by such undesirable events can have catastrophic effects, including loss of human life, disruption of essential services, and huge financial losses. The arrival of the STUXNET worm [10] as well as its variant Duqu and the recently famous flame virus are perfect examples of cyber terrorism. Given the devastating effect malicious code can have on our cyber infrastructure, identifying and containing malicious programs is an important goal for malware researchers.

Malware is usually classified according to its propagation method [9] and goal as viruses, worms, Trojans, backdoors, spyware, droppers, etc. There exist several techniques for creation, detection and analysis of such a variety of malware. In this paper we will focus only on computer viruses.

2. Virus Analysis Techniques

Many techniques for detecting, neutralizing and analyzing viruses exist and are being used in practice. There are two main families of analysis techniques. These are Static and Dynamic analysis. Both methods provide their own set of strengths and weaknesses while their success hinges largely on the type of virus involved [2].

2.1 Static Analysis

Static malware analysis can be loosely defined as searching for malicious intent within a binary executable. This has been the tried and proven malware analysis and detection technique for many years. In this technique, the computer program is viewed as a series of instruction sets which can be seen from within the binary executable. Many malware detection programs and software were initially based on signatures or patterns which were provided by the malware analyst, to characterize malicious behavior. The limits of this approach as the sole analysis technique has been discussed in the paper by Kruegel, et al [3].

2.2 Dynamic Analysis

Dynamic techniques for malware detection and analysis attempt to solve the problem from a different perspective. This focuses on observing a program's behavior as it runs. Rather than attempt to detect malicious intent before run-time, dynamic analysis systems monitor running programs for malicious behavior. The benefit to this approach is that one doesn't have to guess whether the program will do something malicious or not, but can observe it.

In reality though, defining malicious behavior is itself not always so clear-cut. The resulting high number of false-positives over static methods are one of the issues researchers face when developing new dynamic analysis techniques.

Monitoring a process' environment for certain properties is the most common way dynamic analysis tools track process behavior. This can (and usually does) have an adverse effect on performance. A common approach to this type of monitoring is to run a program inside an emulated environment because of the extra monitoring capabilities available [11]. Emulation typically does not perform

particularly well though, especially while simultaneously handling and processing events occurring within the emulation platform.

3. Packing and Packers

Packing is a strategy used by malware writers to mask the meaning of their malicious software in an attempt to foil signature-based analysis. A packing operation compresses a portable executable (PE), masking its behavior and producing a packed version of the original executable [6]. This interferes with most static analysis of the resulting file. The majority of new malware variants utilize some form of packing to defend against detection.

A packer, on the other hand, is a packaging tool designed to perform compression on an executable program, with the added side-effect that much of the internal meaning of the program is hidden. A packed executable will typically look either normal, or entirely undecipherable normally, but at runtime, the executable will unpack itself and execute the packed segments of program code

4. Analysis of the Makier Virus

The Makier virus is a fairly recent virus strain that attacked the windows operating system and was first reported in September 2012. Initial scans with popular anti-viruses offered no suggestions other than the fact that it could be a variant of a worm, Trojan or W32 Cryptor virus [12]. The papers by R. Flores [1] and Kendall [7] gave us an understanding of the analysis goals and methodology. The analysis of this virus was performed with four goals in mind.

(i) Identify the extent of virus activity.
(ii) Does this virus act as a dropper for another malware?
(iii) Is the program packed? If so what was the packer used?
(iv) Identify and locate the various windows dll's used by the virus.

Viruses are created with only one major goal in mind. Namely, to stealthily destroy a computer. This of course implies that a dedicated and isolated environment is required to "analyze" malware. One solution to this problem is to set up a network of computers that are isolated from the rest of the world. These machines should have software that can be restored from an image after the malware has finished its evil work. However, it is much easier, but less safe to create a simulated environment by making use of a single machine that makes use of a virtualized environment to create a simulated lab environment. It is this second (much easier) method that we have followed. We have also made use of popular malware analysis tools in order to perform this analysis [4] [5].

4.1 Packer Detection

One of the major complicating factors in performing malware analysis is the proliferation of programs that modify an executable file to obfuscate its contents and hide the actual program logic from a reverse engineer performing static analysis. These "Packers" modify the executable so that the original program data is very hard to recover. Here we use PEiD [13] to identify the packer, if it is used. PEiD also provides other details (Fig. 1, Fig. 2) such as entry point, size of image, export table, etc.

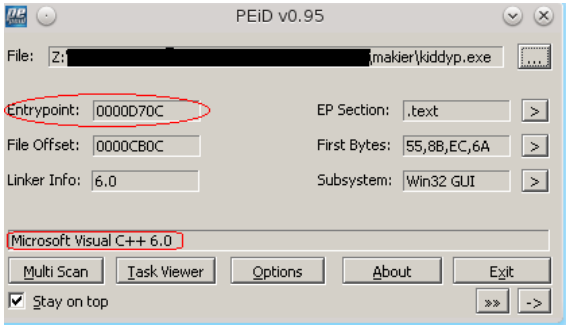

Fig.1 Packer Identification using PEiD

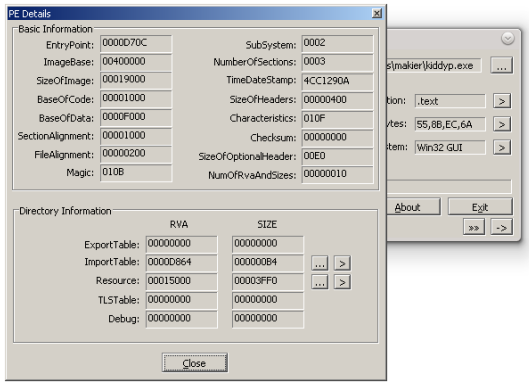

Fig.2 Entry Point Identification using PEiD

4.2 Program Static Analysis

To understand what the program does, it would be ideal if documentation was available. However, malicious programs come with no such manuals. The simplest way to

begin analysis, we found, was to analyze the strings of readable text that are embedded within the program. We used OllyDbg [14], a powerful windows based debugger. We found that it is easy to search for strings and their references to understand the logic of the programmer (Fig. 3). Of course, the more experienced the analyst, the more the information gleaned from stepping through the program instruction.

Fig.3 Analysis using OllyDbg

4.3 Program Dynamic Analysis

Process Monitor [15] is a SysInternals tool that allows users to monitor windows system activity. We used Process Monitor to view the activity (Fig: 4) of the virus in the host computer, the observations are mentioned in section 4.4 of this paper.

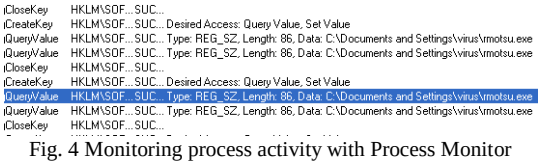

Fig. 4 Monitoring process activity with Process Monitor

Further, we made use of SysTracer [16], a system utility tool that can scan and analyze computer to find changed (added, modified or deleted) data into registry and files. Initially we created a base snapshot of the virus free environment. We termed this snapshot as the baseline. Then the virus was introduced into the system and then a second snapshot, which we called the malicious snapshot (Fig 5), was taken and we compared the results. This also we include in the observations section (Section 4.4).

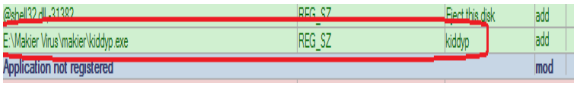

Fig. 5 Malicious snapshot

4.4 Observations

The Makier Virus is a PE32 executable for Windows operating systems. PEiD analysis revealed that this malware was packed using Microsoft VC++ 6.0, which is a very common packer for viruses. The propagation executable was identified as kiddyp.exe. This program creates and starts a hidden executable rmotsu.exe (C:/Documents and Settings/User_victim/ rmotsu.exe). A Microsoft Crypto API import was also detected and so we classify the program in the category of encrypted viruses. We noticed that various Microsoft dll files such as kernel32.dll, USER32.dll, gdi32.dll, camctl32.dll, etc. were also accessed by kiddyp.exe. From the www.virustotal.com database, we found that the first reported date of origin of this virus was September 2012. The propagation executable of the virus steals memory resources, creates illegal hidden processes, injects and executes a hidden windows executable that performs registry modifications.

During analysis, we noticed that the virus propagation occurs only through physical media, namely the USB device. A hidden directory named Makier is created on the external physical drive and within it, a hidden executable kiddyp.exe is placed. In fact, one of the main reasons for calling this malware as the Makier virus is due to the creation of this hidden folder. We could also say that the Makier virus is a combination of two programs, namely kiddyp.exe and rmotsu.exe as they are mutually dependent. The program is initialized and executed by a hidden autorun.inf program also injected into the infected USB device.

5. Conclusion

Computer viruses are becoming the real terrors of the computer world. With malware becoming increasingly complicated, analysis of these programs is also very time consuming. Malware authors use a range of evasion techniques to harden their creations against accurate analysis. The evasion techniques [8] aim to obfuscate code or even disrupt attempts of disassembly, debugging or analyze in a virtualized environment thus adding to the difficulty of malware analysis.

This paper provides a simple insight into the world of malware analysis. By analyzing a fairly recent virus strain using both static and dynamic analysis techniques, we were able to identify the extent of virus activity, its propagation technique, the various dll files used, as well as the packers employed. Additionally we also identified the original entry point of the virus code. Using this it is possible to create signatures for this specific virus strain thus helping the antivirus industry.

References

[1] R. Flores, "Malware reverse engineering part 1. Static Analysis", Official Malware Report, http://packetstormsecurity.org, 2012
[2] M. Christodorescu et al, Malware Detection, Springer, 2007.
[3] C. Kruegel, A. Moser and E. Kirda, "Limits of Static Analysis for Malware Detection", In 23rd Annual Computer Security Applications Conference (ACSAC), pp. 421430, 2007
[4] M.Egele, T. Scholte, E.Kirda, C. Kruegel, "A Survey on Automated Dynamic Malware-Analysis Techniques and Tools", ACM Computing Surveys (CSUR), Volume 44 Issue 2, February 2012
[5] L. Zeltser, "Reverse Engineering Malware", www.zeltser.com, 2001
[6] M. Vuksan and T. Pericin, "Constant insecurity: Things you didn't know about portable executable file format," BlackHat, 2011.
[7] K. Kendall, "Practical Malware Analysis," Mandiant-Intelligent Information Security, 2007.
[8] R. R. Branco, G. N. Barbosa and P. D. Neto, Scientific but Not Academical Overview of Malware Anti-Debugging, Anti-Disassembly and Anti-VM Technologies, BlackHat 2012
[9] A. Miraglia, "Analysing the spreading of computer worms and viruses: potentials and limits", Department of Computer Science, University of Zurich, 2011
[10]J .Larimer, "An inside look at Stuxnet", http://blogs.iss.net/archive/papers/ibm-xforce-an-inside-look-at-stuxnet.pdf, 2009
[11] C. A. Benninger, "Maitland: Analysis of Packed and Encrypted Malware via Para-virtualization Extensions", MS Thesis, University of Victoria, 2010.
[12] http://www.virustotal.com
[13] http://peid.com
[14] http://www.ollydbg.de
[15]http://technet.microsoft.com/en-in/sysinternals/bb896645.aspx
[16] http://www.blueproject.ro/systracer