A Framework to Detect Novel Computer Viruses via System Calls

A. A. Abimbola, J. M. Munoz and W. J. Buchanan
School of Computing, Napier University, EH10 5DT, Scotland, UK
a.abimbola@napier.ac.uk

Abstract- **This paper describes a framework for detecting self-propagating email viruses based on deterministic system calls derived from associated email client's dynamic link libraries (DLLs). Our research approach is based on the principle that a key objective of an email virus attack is to eventually overwhelm a mail server and clients with large volume of email traffic. A virus achieves this by propagating to other email addresses in the infected email client inbox, alongside activating its payload. In doing this, the virus executes certain malicious processes, resulting in the creation of abnormal system calls via related DLLs. Our research effort advances Stephen Forrester earlier contribution that proved normal and abnormal system calls from a email client in a Unix platform could be differentiated, by describing a framework on how to monitor and detect abnormal system calls in real-time from an email application.**

1. INTRODUCTION

An email computer virus, or virus, is a piece of code with two structural subroutines [1]. One with the capability to reproduce and the other to transfer instances to other email clients via email addresses found in the initially infected email client. In addition, there is a payload or malicious act that may await a set of predetermined circumstances before being activated or triggered. Viruses remain a significant threat to modern networked computer systems. Despite the best efforts of those who developed anti-virus systems, new viruses, such as w32/sober.p@mm (McAfee adversary-www.mcafee.com) and others that implement hybrid exploitation techniques [2] are not dealt with by present anti-virus systems. In addition, the rate at which a virus can spread has risen dramatically with the increase in connectivity and also aided by the ease of accessing virus production toolkits [3].

Traditional anti-virus techniques typically focus on detecting static signatures of viruses. Whilst these techniques are effective, they do not address the dynamic nature of a virus infection within the context of an underlying system. For instance, polymorphic viruses [4] alter their instruction codes either by substitution or encryption methods to replicate a new viral instance.

There has been an increase in research efforts targeted towards using viruses as direct weapons of information warfare [5]. A key drawback to these research efforts is that viruses have mainly wandering properties. As a result, computer viruses do not discriminate between friend or foe in executing their payload. This non-discriminatory cruise-ability, makes viruses more suited as a terrorist weapon [6]. Our earlier research efforts [7], introduced a novel subroutine into a virus, enabling it to propagate from one network terrain to another, seeking a specific intended target. As a result, a discriminatory cruise-ability was achieved and enhanced our experimental virus for direct information warfare [7]. In another article [8], we proposed an approach to curbing cruising viruses and viruses as a whole using system call wrappers, while in the article "viral-net system via autonomous agent technology" we described an architecture using autonomous agents for similar reasons [8].

In this paper, we investigate further the design of a framework that employs the creation of a system call wrapper to isolate or confine the activities of an email client to innocuous trends only. Our research approach involves observing and logging system calls created when an email client executes in real-time malicious processes. These logged system calls are then compared to current system calls and any deviation could be perceived as abnormal.

11. BACKGROUND AND MOTIVATION

Viruses are programs that can "infect" other programs by modifying them to include a possible evolved version of it. With this infection property a virus can spread, from program –to program, host –to host and network –to network, corrupting the integrity of information as it spreads [7]. Given the widespread

ISBN: 1-9025-6013-9

use of file sharing on the Internet, the threat of a virus causing wide integrity corruption is significant [9]. Virus propagation can be divided into two phases, infection and payload activation. Some issues related to the infection phase are outlined below:

- Viruses can act as carriers for other information, and thus can be used to cause arbitrary effects [10]. They can carry other attacks along with them, and thus bypassing many of the protection mechanism that would otherwise be in place against the embedded attacks. Hence a virus can be used to introduce covert channel, modifying operation controls, or do almost any other damage [11],
- Infection may be performed on any information that is interpretational and may infect any other interpretable information including files-spreadsheet, database, compilation files, load libraries, macros and others.

Possible threats posed by malicious payload activation are data damage and harvesting, hardware damage and espionage as described in [12].

There have been several defenses implemented against viruses using systems such as pattern matching programs. These programs detect known attack patterns, and they are quite successful at this, but they present some problems such as:

- Firstly, pattern-matching technique only works against known attack patterns, and thus only detect viruses we already know about, or fail to detect evolving viruses [13],
- Secondly, we can detect larger classes of variations by having less accurate patterns (signatures), but this also increases the likelihood of false positives.

Other viral defenses implemented that exist in the research community include: - comparing operating system current parameters [13], proper protection state and real-time virus direction system using iNetmon Engine [14] just to name a few.

Recent advances in computer virus research have shown that viruses are able to target a specific host [7]. Thus, they propagate from host –to host, and only execute their payload at the target host. This type of viruses are known as cruising viruses [7] and are difficult to obtain intrusive signatures for, because they are practically invisible as they propagate and only execute their payload at the target host. In thwarting cruising viruses, a Viral-Net System was proposed in [8]. In broad terms, Viral-Net architecture consists of a detecting agent that discovers previously unknown viruses via system calls and a central analysis agent that automatically derives intrusive signatures and removal procedures for captured viruses. In this paper, we describe a framework on how the detecting agent highlighted above derives system calls for virus detection.

111. Related Work

M.Rhattacharya et al. developed the Malicious Email Tracking (MET) system to track the flow of malicious email such as self-replicating viruses through a network [15]. The novelty of their research was that sampling the entire email track exchanged between servers, in a wide area network, the tracking of the flow of malicious email traffic was not required. Nevertheless, its techniques for determining malicious email, such as the use of MD5 for identification of the propagation of the same virus, can be defected by polymorphic viruses.

M.Schultz et al.[16] attempt to use a data mining approach to detect malicious executables embedded within emails. Their attempt focuses on short sequences of machine instruction as a means to detect malicious executables. To determine if an attachment contains malicious code, a Naïve Bayes classifier was trained on a set of malicious and benign executables and then used as a determinant. A demerit of this approach is the assumption that there are similarities among the binary code of malicious executables. Whilst this may be true for current viruses, stealthy viruses may prove otherwise [17].

A complementary research approach to MET was introduced by S.Stolfo et al. [18], coined Email Mining Toolkit (EMT) and uses data mining to synthesize the behavior profile of users that is used by MET to detect malicious email. EMT incorporates modelling of normal email users behavior based on several categories, including patterns and frequency of correspondence, to detect malicious activities. As a result of this methodology, MET alerts at slight changes in communication pattern between email correspondences, which may lead to high false positive rate of attacks.

Static analysis techniques that verify program for compliance with security properties have also been proposed for the detection of malicious executables. Biship and Dilger showed how file access race conditions can be detected dynamically [19]. Tesauro et al. [20] used neural network to detect boot sector viruses, while Lo et al. proposed to use “tell-tale

signs" and " program slicing" for detecting malicious code [21]. LaBrea, an intrusion prevention tool, is able to trap known intruders by delaying their communication attempt [22]. The virus throttles research efforts carried out by Williamson et al. [23] who utilized the temporal locality found in normal email traffic and were able to slow down and identify massive mailing viruses as they made massive mailing connection attempts. Their success in both intrusion prevention and tolerance confirms the effectiveness of behavior modification based methods. These approaches depend on knowing, or having a prior sample of, an email virus for analysis before infection takes place. They also rely on the massive email traffic generated by an email virus during propagation. A cruising virus will prove difficult to detect using the above research approaches, since it is elusive and only propagates to selective addresses in the infected email client, as a result generating a low email traffic.

1V. An Overview of Our Virus Detection Framework

In this section, we will define the various components of our research framework. Our framework consist of the following stages: determine related application system calls, and a training period to create a malicious system call expert database system. These two stages are discussed in detail below.

To determine system call messages created by any application, we need to first locate the DLLs associated by that application. A possible approach in determining DLLs used by Microsoft based applications is to use Microsoft's Portable Executable (PE) [24] file format, see Figure 1 for PE structure. Microsoft introduced PE file format for backward compatibility between its various versions of operating system.

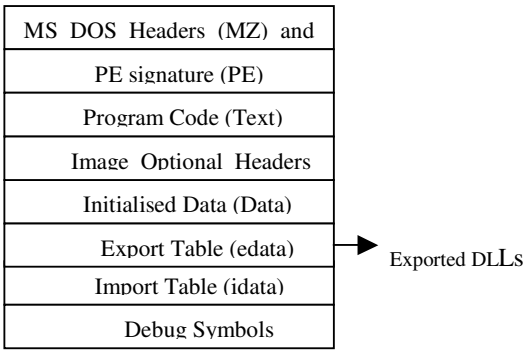

Fig.1. Portable Executable File Format Structure

A typical PE file consists of a list of export functions or static DLLs that are called by the PE during operations. These DLLs functions are within the win32 API architecture and produce system call messages.

They exist in the Internet downloadable programs, that can extract the list of export functions or static DLLs from any Microsoft based application [25]. Another approach would be to lookup Microsoft DLLHelp database [26], where a list of all associated DLL to any Microsoft based application can be found.

Determine Related Application System Calls- To monitor and log system call messages made by applications via associated DLLs, we need to inject a series of bogus DLLs with the same name as the Microsoft's supplied associated application DLLs. There are various techniques on how to inject DLLs [27], like system-wide hook technique, but we will focus on the most used technique, Registry modification. Microsoft's registry provides a registry key where a DLL path that would be loaded with all Microsoft based applications can be inserted. The actual loading occurs when user32.dll initialises and inputs the DLL path from registry, then calls loadlibrary() from its DLLMain code to load the related DLL from memory. The negative side effect to this technique is that the operating system needs to reboot before any changes can be effective. Also, the overhead induced by loading the injected DLL into every process of all Microsoft based applications, instead of just the application being monitored for intrusion, is high [24].

Training Phase Period- The training phase period, in intrusion detection, refers to the exposure of an intrusion detection design (IDD) to a series of malicious operations it is expected to detect. This is usually carried out after a risk assessment of the training phase period has taken place in a controlled environment. It is a period that enables the detection of malicious system call messages generated from a Microsoft supplied DLL owing to executing known and unknown (cruising) viruses. The reader is encouraged to use the Figure 2 to aid their understanding of the context in this section.

In order to fully appreciate our training phase period, we need to, first, define certain unique trends of a viral system call message generated from malicious email attachment. When a user receives a viral email attachment, executing this malicious attachment may take place with or without the user

clicking it. As a result of the attachment executing, the user's contact folders are likely searched and a series of listed address details are added to a new/forwarded message created by the virus that includes the malicious attachment before they are dispatched.

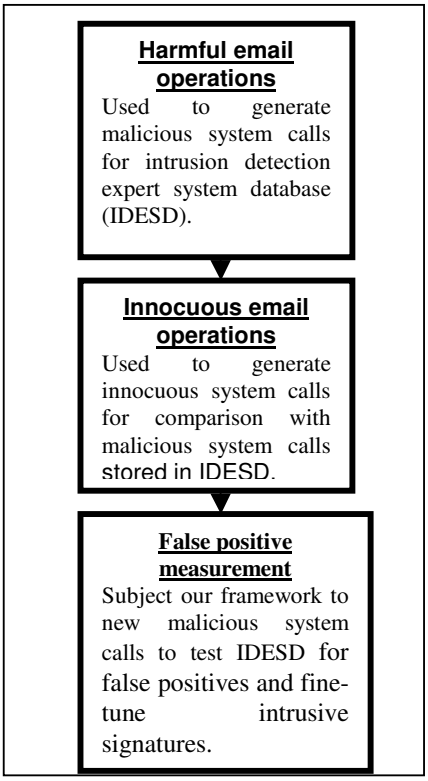

Fig.2. Operations of Our Training Period

In addition, the malicious attachment may execute a payload. The associated malicious system call messages that are generated, by executing a viral email attachment, reflect these minimal to none user interaction during their life span of a viral activity. For example the user does not carry out the following operations:

- click the forward/new bottom or use hot keys,
- attach dispatched sender address details to the "To" input-box either via the keyboard or contact folder,
- click the send button or use related hot keys.

This trend can be used as a means of detecting known and unknown viral email attachment.

The first phase of our training period is to subject our IDD to viral email attachment and then execute these attachments. While doing this, we log all malicious system call messages that are generated via our bogus injected DLLs. The duration of this initial phase usually depends on the number of viral email attachments. We recommend as many viral email attachments be used as possible, to provide as large as possible malicious system call message to train with. Also no other application should be running during this phase apart from the ones being used for the experiment, to avoid unwanted system call messages being generated.

The second phase of the training period involves performing a series of normal operations on Microsoft email client; these operations should include innocuous daily activities. The system call messages generated during this period are logged and compared with those of the first phase. In comparing, we seek only system call messages that are unique to the first phase of the training period, when viral email attachment are executed and not to the second phase when innocuous email client operations were carried out. These unique viral generated system call messages should then be stored in an Intrusion Detection Expert System Database (IDESD).

A third and final phase of our training period will be to subject the IDESD to a new fresh operation of normal Microsoft email client activities to measure its false positive rate of attacks. We recommend the reader adhere to DARPA 1999 suggestion of 10 false positives for every 100 transactions [28]. If an acceptable false positive rate is reached then the IDESD is ready to be installed online otherwise any of the phases could be repeated to fine tune the IDESD.

V. Conclusion

The negative effects of computer viruses have been a thorn to the research community working on intrusion detection for numerous years. Their elusive nature has bemused intrusion detection experts; these elusive techniques include: disabling an anti-viral software in the target host before activating a payload; monitoring and intercepting any interrupt request call made to the memory address it resides in; encrypting itself to prevent positive pattern matching, amongst others. As an attempt to thwart the negative effects of computer viruses, intrusion detection experts have emerged with several research approaches [12] that mainly depend on pattern matching methodology.

We follow through from our two earlier publications [7,8]. Where in [7] we describe a

cruising virus that only execute its malicious payload at a target host. And in [8], design a Viral-Net System that discovers previously unknown viruses via detecting agent and a central analysis agent.

In this paper, we underpin our earlier two publications by designing a framework that tackles how a detecting agent will isolate possible email viral samples. Our design framework consist of two components:- locating Microsoft email client associated DLLs and isolating malicious system call messages that these DLLs generated. We described the first component can be achieved by using Microsoft Portable Executable file format that exports static DLL called by any Microsoft based application. While for the second component, we introduce a training phase period that exposes our framework to malicious and innocuous email operations to isolate malicious system call messages only.

We conclude this paper by suggesting an alternative approach like system calls to detecting computer viruses, other than pattern matching. This approach in not vulnerable to cruising viruses and further research

References

[1] F. Cohen, "Computer Viruses", PhD Dissertation, University of Southern California, ASP Press (PO Box 81270, Pittsburgh, PA 15217 USA), 1986.

[2] "Response Strategies for Hybrid Threats", www.Itsecurity.com, 2005.

[3] "Building Anna Kournikova: An Analysis of the VBSWG Worm Kit", http/www.online. securityfcus .com /infocus /1287, 2005.

[4] Harley et al (2001), "Viruses Revealed", ASIN: 00721 30903.

[5] S.Hang, "Autonomous Mobile Cyber Weapon", http://www.tl.infi.net/~wtnewton /vinfo/bs3.html, 2005.

[6] S.Yang "Movement of Viruses", http://www .intergate.bc.ca/personal/yang /movement.htm, 2005.

[7] A.Abimbola, J.M. Munoz, W.J. Buchanan, "Analysis and Detection of Cruising Computer Viruses", ECIW 2004.

[8] A.Abimbola, J.M. Munoz, W.J. Buchanan,, "Viral-Net System Via Autonomous Agent Technology", BCS SGAI Symposium/Colloquium on Mobile Agent and Complex Agent System, Vol: 7, No:3, ISSN:1465-4091, 2005.

[9] R.R Linde, "Operating System Penetration", AIFIPS National Computer Conference, pp: 361-368, 1975.

[10] F. Cohen, "Computer Viruses- Theory and Experiments",Computers and Security, vol: 6, pp: 22-35, 1987.

[11] F. Cohen, "A Short Courses on Computer Viruses", ASP Press, (PO Box 81270, Pgh PA 15217, USA), 1990.

[12] B. Dwan, "The Computer Virus — From There to Here.: An Historical Perspective", Computer Fraud & Security, Vol: 2000, Iss: 12, pp: 13-16, 1 Dec 2000.

[13] J.C. Wierman and D. J. Marchette, "Modeling Computer Virus Prevalence with a Susceptible-Infected-Susceptible Model with Reintroduction", Computational Statistics & Data Analysis, Vol: 45, Iss: 1, pp: 3-23, 28 Feb 2004.

[14] N. Sathainanthan, N.C. Keong and C.S. Jong, "Real-Time Virus Detection Using iNetmon Engine", Network Research Group, School of Computer Science, University Science Malaysia, 2003.

[15] M. Bhattacharyyaet al., "MET: An Experimental System for Malicious Email Tracking", Proceedings of the 2002 Workshop on New Security Paradigms, pp:3-10, 2002.

[16] M.Schultz.M et al , "MEF: Malicious Email Filter A Unix Mail Filter that Detects malicious Windows Executables", USENIX Annual Technical Conference, pp: 245-252, 2001.

[17] S.Kumar and E.H. Spafford, A generic virus scanner in C++, In Proceedings of the 8th Computer Security Applications Conference, pp: 210--219, Los Alamitos CA, December 1992. ACM and IEEE, IEEE Press.

[18] J. Stolfo.J et al "A Behavior-Based Approach To Securing Email Systems", 1st International conference on Applied Cryptography and Network Security, 2003.

[19] M.Bishop and M.Dilger, "Checking for Race Conditions in File Access", Computing Systems 9, pp: 131-152, 1996.

[20] G.Tesauro, J.Kephart and G.Sorkin, "Neural Network for Computer Virus Recognition", IEEE Expert 11, pp: 5-6, 1996.

[21] R.W. Lo, K.N.Levitt and R.A.Olsson, "MCF: A Malicious Code Filter", Computer and Security, Vol: 14, pp:541-566, 1995.

[22] LaBrea Sentry IPS: Next Generation Intrusion Prevention System: http://www.labreatechnologies. com/ 2005.
[23] M.Williamson,“Throttling Viruses: Restriction Propagation to Defeat Malicious Mobile Code”, 18th Annual Computer Security Application Conference, pp: 61-68, 2002.
[24] M.Pietrek, “An In-Depth Look into the Win32 Portable Executable File Format”, MSDN Magazine, February 2002.
[25] ExeInfor Utilities, “ www.nirsoft.net”, 2005
[26] Microsoft DLLshelp Database, http://support. Microsoft.com/dllhelp/, 2005.
[27] API Hooking Revealed, www.codeproject. com, 2005.
[28] J.W.Haines, R.P.Lippmann, D.J.Fried, E.Tran, S.Boswell, M.A.Zissman, "1999 DARPA Intrusion Detection System Evaluation: Design and Procedures", MIT Lincoln Laboratory Technical Report.