

Symbian worm Yxes: Towards mobile botnets?

Axelle Aprville (Fortinet Technologies - France)

Abstract

In 2009, a new Symbian malware named SymbOS/Yxes was detected and quickly hit the headlines as one of the first malware for Symbian OS 9 and above all as the foretaste of a mobile botnet. Yet, the detailed analysis of the malware was still missing. This paper addresses this issue and details how the malware silently connects to the Internet, installs new malware or spreads to other victims.

Each of these points is illustrated with commented assembly code taken from the malware or re-generated Symbian API calls. Besides those implementation aspects, the paper also provides a global overview of Yxes's behaviour. It explains how malicious remote servers participate in the configuration and propagation of the malware, including Yxes's similarities with a botnet. It also tries to shed light on some incomplete or misleading statements in prior press articles. Those statements are corrected, based on the reverse engineering evidence previously.

Finally, the paper concludes on Yxes's importance and the lack of security on mobile phones. It also indicates several aspects future work should focus on such as communication decryption, tools to analyze embedded malware or cybercriminals motivations.

Parasitics. The Next Generation

Vitaly Zaytsev (McAfee Labs - USA), Josh Phillips (Kaspersky Labs - USA) and Abhishek Karnik (McAfee Labs - USA)

Abstract

Over the past decade parasitic viruses and anti-virus (AV) technologies have participated in an elaborate game of cat and mouse. The war against viruses continues to escalate. Advanced code obfuscation and mutation techniques have been employed to evade detection of virus defence systems. At the same time pattern-based virus scanners from the past have evolved greatly. Advanced engine designs, next generation battle-tested engine technologies, in-depth inspection techniques, behavioural monitoring have evolved to reflect the nature of today's complex threats. In this paper, an in-depth analysis of two of the most recent advanced and sophisticated viruses (W32/Xpaj, W32/Winemem) is presented, along with the new techniques they use to transform their code to avoid detection by AV scanners. We will discuss the novel usage of virtual machine (VM) based obfuscations employed by Win32/Xpaj, ways in which VM based obfuscators can be defeated and the novel ways Win32/Winemem and Win32/Induc infect their hosts.

BackDoor.Tdss (aka TDL3)

Alexey Tkachenko (Dr.Web - Russia)

Abstract

This rootkit is the most rapidly developing and technologically advanced malicious program at the moment. More than 30 modifications of it have been released since the end of September 2009. One can count vendors able to deal with an active rootkit on fingers of one hand. In this report we try to describe main difficulties in detecting the rootkit and curing the system, and follow up rootkit's development throughout different versions.

Entropy based detection of polymorphic malware

Zdenek Breitenbacher (AVG Technologies – Czech Republic)

Abstract

The last year brought a lot of news in the field of malware evolution. The bad news is that polymorphic malware now becomes the standard. There are many new threats, viruses as well as Trojans, which all utilize polymorphism, making analysis as well as detection more and more difficult.

Fortunately, there is also good news: Although each copy of polymorphic malware is totally different in a simple binary view, we still can find some characteristics, which remain always the same, or at least very similar. We only have to forget all previous methods of detection, especially those which were based on searching for some typical signatures. So, which characteristics can the malware analyst use? One, the most efficient of them, is entropy. But instead of the whole entropy, as one number describing the whole file, we need very detailed entropy map, which describes an entropy course throughout the file.

We are going to show that inspecting the entropy map a malware analyst can easily isolate different parts of the file, both innocent as well as suspicious ones. We will show that an entropy map of one polymorphic family often remains the same for all their copies. In fact, such entropy map can act as a special kind of signature, which can replace the classic one.

The entropy map can bring a new and unexpected view on malicious file and may help malware analysts in many different tasks. We will show the real entropy maps, which describe various binaries. Utilizing model samples, we will examine how to use the entropy map to detect polymorphic malware. We will also show that computing the entropy helps us to avoid false positives, as additional checking along with the traditional signature checking. We will also demonstrate that entropy map can unveil an obfuscated code and distinguish it from legal and straightforward code, as a strong heuristic indicator.

Finally, we are going to suggest some other areas of the anti-malware effort where the entropy based methods can be used effectively.

Windows 7 - Is it really more secure?

Itshak Carmona (HCL Technologies Ltd – Israel)

Abstract

Since October 2009, I have been running a pre-beta copy of Windows 7, the next OS from Microsoft. The Security Center, introduced in Windows XP SP2, is gone. Instead, there is an "Action Center" that incorporates alerts from 10 existing Windows features.

In both Vista and Windows 7, Microsoft has sought to improve application security as well as Windows' resiliency to application-specific vulnerabilities such as 'buffer overflow' exploits. However new code means new bugs and hacking opportunities for those out there. Add to that some of the old school 'backdoors' and security holes that still in place, and it seems there is still a long way to go...

In the paper I will explore in more technical way some of the new and old vulnerabilities and other security holes in Windows 7; hopefully by the time of the conference they will be irrelevant.

A single metric for evaluating a security product

Igor Muttik (McAfee Labs - United Kingdom)

Abstract

Historically, successes and failures of a security product were evaluated using binary logic ("detected" or "missed") and represented as a detection rate. Such rates are assumed to represent the quality (probability) of successful protection. But the timing of providing protection greatly affects this probability. We would demonstrate with examples that the "detection rate" which does not incorporate the timing element is not a valid metric.

We analyze the factors contributing to the probability of successful protection, present the mathematical approach to calculating this probability and discuss how this can be implemented in practice. We will show examples of how growing frequency of attacks dictate a statistical approach to how the quality of security software should be measured.

Obviously, for each attack, the "success of protection" is a function of time. For multiple attacks we will have a set of such functions. We would argue that a simple and meaningful numeric representation of this set of functions would be a probability calculated as an average of integrals of these functions over time.

In this model overall probability depends on the timeframes used to evaluate each attack. To be meaningful, the selection of these timeframes has to take into account users' exposure to the threat. But full knowledge about the exposure is only available after the attack so we have to deal with historical data. We investigate the effect of choosing the timeframe on to the calculation of the final probability.

An obvious optimization in calculating the integrals is to exclude timeframes when there were no updates to security software. Additional considerations include (a) different update cycles for security components, (b) generation of local knowledge (learning through artificial intelligence) and (c) cloud-based security where the update frequency is unknown. Moreover, for cloud-based security there could easily be a dependency on the location of the client and the connectivity conditions. We analyze the effect of these factors on our model.

In Combat against Rootkits

Robert Lipovsky (ESET – Slovakia)

Abstract

The days when malware used to display wicked popups and texts simply to irritate the infected user and stroke the ego of malware writers are long gone. Viruses, Trojans and worms can fulfill their purpose (robbing the user somehow) better when they remain stealthy or disguised.

Disguise is a great way to stay on the infected computer for as long as possible and is also the dictionary characteristic of Trojans. However, this relies merely on the ignorance of users and usually poses no technological challenge in detection. A very simple example is naming malware executables after system files.

A much more sophisticated way to accomplish stealth is to use rootkit techniques. These methods typically involve modifying the core of the operating system somehow or system data structures. Their complexity varies greatly - from simple DLL Import Address Table modification, through SSDT (System Service Dispatch Table) hooks to bypassing the Windows scheduler. Using such methods, malware is able to hide its presence – its processes, files, network connections, etc.

Obviously, in order to combat these threats, anti-malware technologies must implement certain methods of discovery. The action-and-reaction principle applies here as well, meaning that just as a new, more advanced way of detecting rootkits has been deployed, a way to circumvent it is being invented.

This paper describes a number of common methods used by current Windows-based rootkits and the approaches that can be used to thwart them. Some of these approaches are often known to be simple, yet efficient ways of detecting rootkits. But how well do they fare against current malware? That's the question our paper tries to answer.

In search of that answer a selection of participants from both sides are placed into opposition. After implementing a few detection methods, a test against a collection of rootkits is performed. The strengths and weaknesses of the analyzed hiding/detection methods are presented along with the explanation of their key characteristics.

A view into new testing techniques

Lysa Myers (West Coast Labs - United States) - Matt Garrad (West Coast Labs - United Kingdom)

Abstract

There has been much discussion, in the past couple years particularly, regarding the best way to go about testing anti-malware products. Being a testing organization, this is a subject about which we've given considerable thought. We've begun to implement many changes to the existing testing setups, in the efforts to shift our own testing paradigms. This paper looks to discuss the many things we've investigated and give a view into some of the results this has turned up.

In order to bring testing back in line with user experience, we've changed the timing and nature of the tests themselves. Not only does this mean a change in overall methodology of processing the samples, but a change in how results are analyzed as well. This has also required us to gather and verify our own samples to ensure a fresh and timely sample set, which has presented its own set of issues. Once all that is done, we must then find a meaningful way to present the data to users. As this is a process which does and should continue to change indefinitely, we'll present the most up-to-date report of our progress.

Real Performance?

Ján Vrabec (ESET – Slovakia) - David Harley (ESET - United Kingdom)

Abstract

The methodology and categories used in performance testing of Anti-malware products and their impact on the computer remains a contentious area. While there's plenty of information, some of it actually useful, on detection testing, there is very little on performance testing. Yet, while the issues are different, sound performance testing is at least as challenging, in its own way, as detection testing. Performance testing based on assumptions that 'one size [or methodology] fits all', or that reflects an incomplete understanding of the technicalities of performance evaluation, can be as misleading as a badly-implemented detection test. There are now several sources of guidelines on how to test detection, but no authoritative information on how to test performance in the context of anti-malware evaluation. Independent bodies are working on these right now but the current absence of such standards often results in the publication of inaccurate comparative test results. This is because they do not accurately reflect the real needs of the end-user and dwell on irrelevant indicators, resulting in potentially skewed product rankings and conclusions. Thus, the "winner" of these tests is not always the best choice for the user. For example a testing scenario created to evaluate performance of a consumer product, should not be used for benchmarking of server products.

There are, of course, examples of questionable results that have been published where the

testing body or tester seem to be unduly influenced by the functionality of a particular vendor. However, there is also scope, as with other forms of testing, to introduce inadvertent bias into a product performance test. There are several benchmarking tools that are intended to evaluate performance of hardware but for testing software as complex as antivirus solutions and their impact on the usability of a system, these simply aren't precise enough. This is especially likely to cause problems when a single benchmark is used in isolation, and looks at aspects of performance that may cause unfair advantage or disadvantage to specific products. This paper aims to objectively evaluate the most common performance testing models used in anti-malware testing, such as scanning speed, memory consumption and boot speed, and to help highlight the main potential pitfalls of these testing procedures. We present recommendations on how to test objectively and how to spot a potential bias. In addition, we propose some "best-fit" testing scenarios for determining the most suitable anti-malware product according to the specific type of end user and target audience.

Perception, Security, and Worms in the Apple

David Harley (ESET - United Kingdom) - Andrew Lee (K7 Computing – India) - Pierre-Marc Bureau (ESET – Canada)

Abstract

Apple's customer-base seems to be rejoining the rest of the user community on the firing line. In recent years, criminals have shown increasing interest in the potential of Mac users as a source of illicit income, using a wide range of malware types, while issues with jailbroken iPhones have highlighted weaknesses in Apple's reliance on a white-listing security model.

A recent survey carried out on behalf of the "Securing our eCity" community initiative, however, suggested that Mac (and, come to that, PC users) continue to see the Mac - or at any rate OS X - as a safe haven, while Apple seems wedded to the idea that it has no security problem.

However, analysis of hundreds of samples received by our virus labs tells a different story. While the general decline of old-school viral malware is reflected in the Macintosh statistics, we are seeing no shortage of other malicious code including rootkits such as WeaponX, fake codec Trojans, malicious code with Mac-specific DNS changing functionality, Trojan downloading and installation capability, server-side polymorphism, fake/rogue anti-malware, keyloggers, and adware (which is often regarded as a minor nuisance, but can sometimes have serious impact on affected systems).

Nor is this just a matter of Mach-O (Mach Object File) format binaries: scripts (bash, perl, AppleScript), disk image files, java bytecode, recompiled ELF and so on are also causes for concern. While neither the possibility nor the actual existence of a threat always equates to the probability of its having measurable impact, we take the position that the tiny proportion of compromised machines reflects, at least in part, the still limited market penetration of Apple products. The surprisingly swift escalation of exploits of a single iPhone vulnerability from PoC code to multi-platform hacker tool to functional botnet has perhaps been given more exposure

than its impact in terms of affected machines might deserve, yet it demonstrates how closely criminal elements are watching for any weakness that might be turned to advantage.

A security model based on white-listing and restricted privilege, implemented on the presumption of the user's conformance with licence agreements, can fail dramatically where there is an incentive to circumvent security for convenience or entertainment. Some types of attack (phishing is an obvious example) are completely platform agnostic because the "infected object" is the user rather than something on the system. Security reliant on the inability of a user to gain privileged access may lead to disaster if it fails to anticipate the ingenuity of hobby hackers and criminals alike, or the possibility of a conjunction of social engineering and technical vulnerability.

This paper will compare the view from Apple and the community as a whole with the view from the anti-virus labs of the actual threat landscape, examining: the ways in which the Apple-using community is receiving increasing attention as a potential source of illegitimate profit, reviewing the directions likely to be taken by malware over the next year or two, assessing the likely impact of attacks against Apple users.

The implications for business and for the security industry in an age of interconnectivity, interoperability, and the paradox of accelerated computing power on ever-shrinking devices.

CJ-Unpack: Efficient Runtime Unpacking System

Cristian Lungu (BitDefender – Romania) - Marius Botis (BitDefender – Romania)

Abstract

Antivirus signature based systems have become almost impractical due to the high polymorphism of malware. The most common way malware samples manage to achieve polymorphism is to have their own custom encoding methods (i.e. they are packed). Building a general unpacking framework is thus, as important for the antivirus software (AV) as the signature database itself because it allows a single signature to match the same malware sample, encrypted by different packers.

This paper presents CJ-Unpack, a simple method for generic unpacking that monitors carefully chosen patterns of API function calls as markers for unpacked code. Although this has been already attempted, our approach estimates where the malware code begins rather than where the packer code ends and monitors all the API functions in use rather than a special sub-set of API's that could heuristically mark the moment of unpack. This is useful because most of the malware is built with standard programming languages, libraries and compilers that can be easily recognized from the API flow. Expanding this idea, CJ-Unpack can recognize the compilers that were utilized to build the malware (C/C++, Delphi, Visual-Basic, .NET or others). The API patterns used for detection are generated by data mining algorithms applied on a large common-compiler API flow database. The methodology described could also be applied on

malware detection to generate behavioral signatures as API call patterns. CJ-Unpack implements a continuous monitoring approach, where the execution is observed in its entirety allowing for multilayer unpacking. The monitoring is done by a generic inline hooking mechanism done through an injected dynamically linked library.

Our technique is extremely efficient and is already used in our antivirus products. It can estimate the original entry point (OEP) dynamically on a live malware sample packed by one or more packers, without the need of an emulator or a virtual machine.

Is there a future for Crowdsourcing Security?

Methusela Cebrian Ferrer (HCL – CA – Australia)

Abstract

The world wide web has dramatically changed over the past years, from static pages to dynamic and even more interactive content. In a broader perspective, technologies that goes along with the internet has transformed the society we lived in. For most developing countries, online world is now considered as integral part of daily activity – we play online, we shop online, we work online, we learn online and we interact online.

In the same way, internet threats continuous to flourish each year, attackers are more than ever capable to build and deploy powerful attacks, often regarded for notoriety, political and financial gain. The unprecedented increase of malware reflects a perpetual arms race against cyber criminals.

Web 2.0 geared us into a participative open knowledge sharing culture, where platforms created and designed for individual to contribute ideas, share experiences, provide solutions and raise awareness. The richness of content and freely shared data in web-based communities such as social networks, forums, blogs and micro-blogs empowers internet crowd, for example security researchers responds to newly reported attack resulting to collaboration and timely response of security awareness and deliverables.

"Crowdsourcing is a neologism for the act of taking tasks traditionally performed by an employee or a contractor, and outsourcing them to a group (crowd) of people or community in the form of an open call."

This paper aims to examine the concepts of Crowdsourcing security, how it works, what are the benefits and ethical issues surrounding it. As web-based technologies moves towards interactive social media, real-time web, and capturing geo-specific content, it is important to understand whether Crowdsourcing security is a viable strategy for the security industry.

Typhoid Adware

Daniel Medeiros Nunes de Castro (University of Calgary – Canada) - Eric Lin (University of Calgary – Canada) - John Aycocock (University of Calgary – Canada)

Abstract

Typical strategy for adware authors is to install their software on as many machines as possible and, for each affected machine, display advertisements to the user(s) of that computer. In this paper we present a different model: typhoid adware. Typhoid adware is more covert, displaying advertisements on computers that do not have the adware installed. We prove that this is a viable adware model with three proof-of-concept implementations and discuss possible defenses, for which we have two proof-of-concept implementations.

Benchmarking Program Behaviour for Detecting Malware Infection

Narendra Kumar N. V. (Tata Institute of Fundamental Research – India) - Harshit Shah (Tata Institute of Fundamental Research – India) - Shyamasundar R. K. (Tata Institute of Fundamental Research – India)

Abstract

Malicious code is any code that has been modified with the intention of harming its usage or the user. Typical categories of malicious code include Trojan Horses, viruses, worms etc. With the growth of complexity of computing systems, detection of malicious code is becoming horrendously complex. For security of embedded devices it is important to ensure the integrity of software running in it. The general virus detection is undecidable. However, in the case of embedded systems the software and hardware configurations are known a priori. Taking this into consideration, we shall benchmark the behaviour with which we can compare. Major part of the current work on malware detection relies heavily on detection of syntactic patterns. Malware writers are resorting to simple syntactic transformations (which preserve the program semantics) such as various compiler optimizations and program obfuscation techniques to evade detection. In this paper we develop a model of behaviour of a program executing in an environment. Our approach to detect tampering is based on benchmarking the behaviour of a program executing in an environment, and then matching the observed behaviour of the program in a similar environment with the benchmark. Since execution behaviour remains the same in majority of obfuscations, our approach is resilient to such exploits. We have performed several experiments in this direction and obtained encouraging results. Differences between the benchmarked behaviour and the observed behaviour quantify the damage due to a virus. This enables us to arrive at refined notions of "harm" done by a virus and come up with measures for protection. Further, we also show how the usual language of system calls can emulate folder calculus.

Counting the Cost of University Internet Access: The Challenges of Balancing Security, Privacy and Forensic Computing

Vlasti Broucek (University of Tasmania – Australia)

Abstract

At present, Australian Universities are experiencing tight budgetary conditions and with the ever increasing reliance on Internet, flexible delivery and other use of on-line systems, the Internet traffic and the associated cost grow exponentially. Most universities rely on a range of ICT systems (new as well as legacy) to support a huge diversity of users and their knowledge as well as acceptance of appropriate on-line behaviours.

As a consequence, the universities are facing challenges of spiralling costs, inappropriate on-line behaviours, the need for protection against legal action, maintaining a balance of users' privacy. In an attempt to these challenges most universities have considered the introduction of Internet Management Systems (IMS), not much different from systems used by most Internet Services Providers (ISP).

On an example of the University of Tasmania, this paper analyses experience of University of Tasmania (UTAS) from the initial 'call for proposals' to deployment of the new system. While it would appear that there is a range of products on the market, none of them initially fully satisfied requirements of UTAS management and series of decisions led to selection of one particular product.

This paper highlights that any decision in this space involves a delicate balance across a range of competing interests. It highlights the changing nature of Internet Access and may be a precursor of further changes as IPV6, Google and Cloud computing change individual users relationships with 'their' information. Issues of access, ownership and control, privacy and freedom of speech are intimately related to these rather technical discussions of the management of Internet traffic.

This paper explores these relationships and highlights the need for continued vigilance on the part of users, network administrators, service providers and policy makers. On examples from two different areas of the university, it demonstrates, that if we are not to create an Internet of 'Big Brother Surveillance' and/or even worse an Internet of 'Self-censoring behaviours' and/or force mass adoption of encryption to ensure privacy and security of users from prying eyes, user education, change management and communication from very top to bottom of the organisation play vital role in successful adoption of such systems.

Bayesian Model-based detection and how to bypass it

Eric Filiol (ESIEA – France) – Sebastien Josse (ESIEA – France)

Abstract

Statistical methods have been used for a long time as a way to detect viral code. Such a detection method has been called spectral analysis, because it works with the instructions statistical distribution, namely the instructions frequencies spectrum.

More recently, data mining techniques, well known for their application in other research fields, such as genetic, speech recognition or text classification have been applied in several security research areas: cryptanalysis, spam filtering and viral detection.

First we will present an approach of viral detection by means of spectral analysis based on Bayesian networks, through two basic examples of such learning models: naive Bayes and hidden Markov models.

The main interest of this approach lies in the fact that we can define such a given model for any viral set. All those models are just derived from a unique general initial model with a suitable parametrization.

We will next present an overview of the criteria that are commonly used to measure the effectiveness of statistical information retrieval models and develop the issue of bypassing detection by statistical simulability which has been previously defined in (Filiol & Josse, 2007) by considering more sophisticated detection techniques and how to simulate them to avoid detection.

From the previous discussion, we will then present information theory based criteria to characterize the effectiveness of spectral analysis models and then discuss methods to bypass any such detection technique by means of testing simulability.

New Trends in Malware Sample-Independent AV Evaluation Techniques with Respect to Document Malware.

Jonathan Dechau (ESIEA Laval – France), Romain Grivaux (ESIEA Laval – France), Kenza Jaafar (ESIEA Laval – France) - Jean-Paul Fizaine (ESIEA Laval – France)

Abstract

Attack based on office documents exists since the 90's with the Concept virus. They exploit an office software functionality called macro which allows the execution of a event-oriented language which is natively embedded in document application. This concept is easy to put in action and had not changed until now. Nowadays it is even the easiest way to perform such attack with the new format of office documents (ODF or OpenXML). This fact has been recently demonstrated by malicious attack using office document (e.g in 2007 the German's chancellery computers were attacked by a Trojan introduced through Microsoft Office documents; this Trojan horse stole information for months). As the example of the German chancellery attack, that this

kind of attacks are very easy to carry out in addition that they are very powerful, thus making them extremely dangerous.

It is more than essential to evaluate the ability of antivirus to detect malware spreading through office documents. Until today, no one has a reproducible, open testing method to evaluate anti-virus product at his disposal. The AV vendors who share samples have jealously restricted the evaluation of their products to their own corporate realm only. It is then necessary for any one who wants to evaluate his anti-virus product to access free tools and techniques.

We have developed those news tools that apply techniques that we had developed especially for documents. As we will see, macro based attack are very easy to put in action with the use of the EICAR's test file.