WHITE PAPER

Detecting Malware and Sandbox Evasion Techniques

Dilshan Keragala

Detecting Malware and Sandbox Evasion Techniques

GIAC (GSEC) Gold Certification

Author: Dilshan Keragala, dkeragala@att.net

Advisor: Chris Walker, C|CISO, CISSP, GSEC, GCUX, GCWN, GWEB

Accepted: January 16, 2016

Abstract

System integrity is a cardinal component of information security. It ensures that information systems operate within some desirable limits. Internet security threats such as malware and highly malicious programs are on the rise, resulting in the necessity for extensive research efforts to develop mechanisms that can counter various threats. Malware Sandbox analysis is an effective mechanism having received propositions as a potential solution. When using Malware Sandbox analysis, samples of malware are executed to determine their behaviors. The results of this action are then recorded for subsequent study. This paper first will explain the nature of malware, then discuss the available methods to detect and control various malware activities. Finally, it will examine the general Sandbox structure, with a major focus on a novel behavior based malware detection method leveraging Sandbox-evasion behaviors as an avenue to detecting, mitigating or totally evading the malware.

1. Introduction

The Internet has revolutionized the operations of businesses, the manner in which transactions are conducted, education programs are administered, and how research works are handled; these are a few of the benefits it has afforded society. However, there are also some inevitable downsides. Cybercrime issues, such as infringement on rights of ownership of intellectual property, phishing and monetary fraud, cyberspace theft of information and data among other cyber-related attacks, have also increased. Among the current threats to personal and business information, malware has proven to be the most significant one. Malware is a general term used for malicious software. There has been an immense investment in intense research efforts and resources towards attempting to understand and eliminate malware threats.

The proposition of Malware Sandbox analysis techniques is an effective remedy to the challenge posed by malware attacks. The concept behind a malware Sandbox analysis system is to capture the malicious program sample in a controlled testing environment (sandbox) where its behavior can be closely studied and analyzed (Messmer, 2013). The automated environment makes it easier to follow the progress of the malware and prepare a database that can be used to study future strains and categorize them into families. Notwithstanding these measures, cyber criminals are developing techniques with which to circumvent the security measures put in place by malware analysts leveraging detection-based sandboxes. They have been coming up with more sophisticated malware to evade the detection techniques of sandboxes (Singh, 2015). In the past, only a few viruses were able to evade detection; today, the number has taken a dramatic increase

Some of the techniques utilized include the creation of hidden executable files; executable files presented to be genuine system files; auto starting files during computer boot up; alteration of replacing their images with that of a different process; sandbox detection, sleeping, or refusing to debug (Kruegel, 2015). Previously, detection evasion in malware was overlooked with the rationale that they were so uncommon, and that use of such malware targeted attacks by very advanced hackers. Today, it has emerged that such malware is common and increasing at an alarming rate. What is most worrisome is that the current systems are established to detect and prevent such malware from orchestrating attacks are not up to par with the authors of the malware beating the techniques being used by system guardians. A report by Symantec shows that the number of new malware attacks had increased by 26%. The report also indicated that, due to the widespread usage of mobile devices, a majority of the malware developed in 2014 targeted smartphones (Park, 2015).

2. Malware Evasion Techniques

Malware leverages a range of techniques used to get around sandbox technology. Numerous countermeasures have been put in place by malware authors to overcome these safety precautions. The methods of yesterday are not completely effective against today's malware. The number of malware capable of evading detection has risen by a shocking 2000% since the year 2014, a very short duration indeed for such a spike (Kruegel, 2015). Using these evasive techniques, they have bypassed both effective firewalls and gateways, and they can also evade discovery by Sandboxes. According to Lakhani (2015), there are four fundamental evasion techniques used by malware. These techniques are categorized as follows: environment specific techniques, human interaction techniques, VMware-specific techniques, and configuration-specific techniques. Configuration-specific techniques include time triggers, extended sleep, process hiding and fast flux. Of these methods, extended sleep and fast flux are the most commonly used. The following subsections discuss these two methods, how they work, and possible countermeasures.

2.1 Extended Sleep

According to Lakhani (2015), extended sleep is one of the most common evasion techniques. Malware use extended sleep calls to wait out the anti-malware analysis system. The fact that it is computationally costly to run a sandbox environment makes this technique effective. "To achieve scalability, most sandbox systems only run a limited and finite range of virtual machines which implies that if malware does not have any reaction inside the sandbox, no malicious activity will be detected. Malware authors employ sleep calls to bypass the automated malware analysis sandbox systems" (Lakhani, 2015). This technique cannot be regarded as new since it has been around for some time; however, it had previously not seen much use. It is only today that malware authors have ventured into regularly using it to bypass sandboxes.

2.2 Fast Flux

"Another evasion technique is Fast Flux employed by malware in IP Fast flux and Domain Name Service (DNS)" (Kasama, 2014). This technique is used by botnets to conceal malware delivery and phishing sites behind a dynamically changing network of compromised host devices. Rapidly changing DNS names and IP addresses make it hard for organizations to stop malware activity by simply blocking IPs; similarly, the dynamic generation of malicious DNS domain names poses a challenge to organizations. Some advanced cyber security firms have used reverse-engineering techniques to impede the activity of these domain generation algorithms (DGAs). "The botnets use fluxing techniques to evade statically compiled and predefined black lists" (Kasama, 2014).

3. How Malware Detects Sandboxes

Malware can determine if a system is running on a virtual machine or sandbox as it scans hard drives and registries. There is detection in the scans of the installation of VMware tools, running processes and registry entries. Authors of malware programs are regularly working to reverse engineer built-in malware detection in Sandbox systems. This VMware-specific malware attack is becoming, even more, sophisticated. Through detection techniques, malware strings can now evade sandboxes and breach detection systems. (Mimoso, 2013).

3.1 User Interaction

The Trojan, Trojan.APT.BaneChan is an example of malware that leverages user interactions to evade detection. "It waits for a given number of mouse clicks before it starts executing to evade automated detection systems looking for human interaction movements and mouse motions before they start executing" (Raff, 2015). Malware writers study mouse and button replication implementation of sandbox systems and have come up with circumvention technology to evade detection. Handling of random user-like behavior by sandbox systems should improve so that malware finds it harder to bypass the established security measures. Current methods have been discovered to beat the commonly used detection techniques of sandboxes by simulating user-like mouse clicking and scrolling movements (Vashisht & Singh, 2014). A good example of malware that has evasive techniques that beat randomly generated user scrolls is one that was found to remain inactive until the reader of an RTF document scrolled to the second page of the document. The malware would remain dormant unless the user scrolled down past the first page and that is when it would execute. The malware used paragraph codes used in Microsoft documents to push itself down to the second page. When a user scrolled to that page, it would be brought to the active window (Vashisht & Singh, 2014). Generated scroll movements in the sandbox did not anticipate scroll to the second page; this is more like a human act than a simulated scroll. In the typical sandbox, a simulated scroll to the second page of a document is never loaded and thus the malicious code remains dormant. Therefore, the sandbox does not discover the virus (Vashisht & Singh, 2014).

Another example of malware using intelligent human interaction based evasion techniques is one that was discovered to check the speed of mouse movement. All it did was check whether mouse movement occurred at speeds that were suspiciously fast. If it does, the malware concludes it is the workings of a sandbox and not those of a real person. Consequently, the malware terminates or goes into a dormant state (Vashisht & Singh, 2014).

3.2 The Online Digital Signature

Surfing the internet allows access to information about one's computer configuration. The information from one's PC can lead to the creation of an online digital signature or fingerprint that uniquely identifies the specific host on a network (Raff, 2015). This identification is a vulnerability that can be exploited by malware authors to breach analysis systems. An automated sandbox system will not change, making it easy for the capitalization of environment-specific vulnerability (Raff, 2015).

4. Countering Malware Evasion

Lakhani (2015) discusses how to counter the extended sleep technique used by malware to evade sandbox analysis systems. He proposes two techniques; dynamically modifying the sleep duration, and increasing malware analysis period (Lakhani, 2015). The latter one entails doing a thorough analysis for a continued and lengthy period. Even though increasing the malware analysis period ties up sandbox resources, it can be used to counter the extended sleep technique. However, the high cost of running Sandbox malware analysis that long greatly diminishes its feasibility. Nonetheless, it has been found to be a very effective method to counter the extended sleep detection evasion technique. The former, highly considered as much more realistic, entails dynamic modification of sleep duration.

Thus, dynamically modifying the sleep duration is the most suitable option. The sandbox will detect that it has long or multiple sleep commands through error-checking or analysis of the malware. The sandbox will then dynamically alter its system clock to convince the malware that it ran for a longer period (Raff, 2015). This method aims at prompting the malware to wake up before its expected sleep duration elapses. This method targets specific malware that uses lengthy or multiple sleep commands. Manipulation of the system clock has the effect of deceiving the malware that it has run for a longer duration than it has run. Eventually, this makes the malware sleep for a shorter duration and activate just in time to get discovered by the sandbox. The advantage of this method is that it does not use up as many resources as the aforementioned one, and is considered the most feasible when dealing with malware that rely on extended sleep detection evasion techniques.

Due to the high visibility of dynamic examination frameworks, malware creators have reacted by concocting effective methods to guarantee that their projects do not reveal any undesirable actions during automated testing. Obviously, when malware does not do anything malicious while being tested, no discovery is conceivable. Basic evasion techniques have been in existence for a long time. For instance, malware may check in the vicinity of a virtual machine, or it may question well-understood Windows registry keys or documents that uncover a specific sandbox. Other malware creators educated their malware to rest for some time, trusting that the sandbox would time out the investigation before anything unusual occurs (Christodorescu, 2005).

Security merchants responded by including some counter-techniques of their own to their frameworks. They included snares that would distinguish situations where malware questions for understood keys and they would drive a system to wake up after sleep calls. This methodology worked sensibly well for some time, in spite of the fact that it is in a general sense reactive in nature. That is, the malware examination framework should be physically overhauled to handle each new, ingenious trap. Accordingly, malware creators who make zero day avoidances can sidestep being identified until the redesigning of the sandbox. Lamentably, malware creators have as of late presented newer techniques that can no longer be taken care of by current sandboxes (regardless of the fact that the trap is known).

The key issue and the explanation behind the major constraint of current sandboxes are their absence of perception into the execution of a malware program. A decent sandbox needs to accomplish two objectives: visibility and stealth. That is, a sandbox needs to see as much as could reasonably be the expectation of the execution of a system. Moreover, it needs to do so in a stealthy manner. Otherwise, it is simple for malware to distinguish the vicinity of the sandbox and adjust its conduct (Christodorescu, 2007).

There are limitations in the current signature-based malware detection solutions especially when it comes to polymorphic variations. Present day malware encompasses high intelligence to infiltrate systems and targeting systems inclusive of tools that generate polymorphic variants of a single threat with the objective of increasing the rate of infection in an automated manner (Chen, 2008). As such software analysis tools have been designed to identify and investigate the behavior of malicious codes without directly involving the user. A good example is Sandbox based automated analysis software applied in the detection of malware. It critically compares the pre-state of the image of the system and then executes samples and its post-state after scanning. Some new developing malware have means of avoiding Sandbox detection techniques with the potential to neutralize the defense mechanism effectively. Anti-malware programmers have therefore come up with methods to trigger unique behaviors of the suspicious software samples that avoid Sandbox execution.

Analyzing targeted program while the execution process in at hand is referred to as dynamic analysis with static analysis being on the opposite end, that encompasses all the techniques that analyze malware by inspecting the code, which is done before executing it (Messmer, 2013). Analysts execute function oriented analysis with the assembly of codes obtained from the binary executable files. It makes it possible to identify entities such as function routines that are yet to be executed while the malware is operating. Such an approach highlights every line of code inclusive of the packers that may exist and polymorphic mutants as well.

During dynamic analysis the processes in the operation of the testing software are included in the monitoring software intercept operation calls by encompassing regarding access to system files, network sockets, typical files, libraries, account information, and memory, making logs of all of them. Analytics favors sandbox based analysis tools during the detection process. It enables the identification of malware even when they are inoperable, encrypted, packed or compress as this method traces the access routes that may be used by the malware to infiltrate system entities (Raff, 2015).

Emulation based analysis and virtual machine-based detection methods have also been used to scan massive volumes of malware code. Emulation methods indirectly run software on the virtual space via emulator image files that were generated prior to the scan and were based on the composition of actual systems. The scan of code and executable files are detected by scanning the engine of a relative system through the virtual memory and CPU and generates reports of the results without putting the actual system at risk (Singh, 2015).

A sandbox is typically a virtual machine that creates independent execution areas without involving the rest of the system where sample software is actually run. Security tools in this entity are a dependable source to retrieve vital sources of data from, with respect to recorded API invocations, access to the registry, manipulation of operation process and tracing network socket activity. It provides a mechanism for security that manages rigid, controlled sets of resources for suspicious programs or untested code. It runs them using either scratch space on disk, constraints on the network. The CWSandbox uses a common virtual machine design that runs the executable file through threads via the injection of the CWMonitor.dll file to inspect the behavior of the suspicious program. It then tracks all the procedures and drafts a report in the main execution module.

4.1 Dynamic Change of Sleep Duration

As an example to explain dynamic manipulation of time to confuse the malware, Lakhani (2015) discusses the nature and resurrection of Khelios botnet, a malware deemed dead four years ago. “A new Khelios sample, TrojanNap calls the SleepEx () API resulting in a 10-minute time-out to evade detection within file-based sandboxes” (Lakhani, 2015). Based on the fact that most sandboxes are created to execute samples for seconds, TrojanNap delays malicious activity beyond the time the sandbox takes to detect the malware. “The Trojan called a Windows API routine, NtDelayExecution () to execute the extended sleep call” (Lakhani, 2015).

4.2 Simulated Human Interactions

Malware authors know the deception behind simulated human interactions generated in a sandbox (Constantin, 2015). It is no longer effective; therefore, relying on using simulated mouse clicks and scrolls as some advanced malware will know they are being fooled and take further evasion maneuvers (Constantin, 2015). They know how to digitally “perceive” a simulated click or scroll. They have integrated means of detecting that human interaction in computer interactions. Therefore, the simulation of clicks and mouse scrolls will have to be done more deliberately to be intelligent and “human” enough to beat malware authors at their own game.

5.0 Sandbox Evasion Techniques

5.1 Sandbox definition and discussion

It is a computing environment that is isolated so that programs or files run in it without having any effect on the application it executes. Sandboxes have traditionally been used to do several tasks. First, they are used to by programmers to test new codes, and second, they are used to analyze malware without putting an entire system at risk. Based on Messmer (2013), sandboxing technology is a security measure aimed at detecting malware programs by passing them through a computer-based system. The technology allows the malware to run while analyzing its traits and behaviors. It is an alternative to signature-based malware detection, a traditional approach, as it is more effective handling stealthy attacks and detecting zero-day malware (Messmer, 2013).

5.2 Current sandbox evasion techniques

Calhoun (2015) tries to explain the up-to-date sandbox evasion techniques. There is a constant battle between security researchers and malware developers with each trying to outwit the other.

Current Sandbox usage normally depends on a virtual domain that contains the visitor working framework. Occasionally, a sandbox runs the working framework specifically on a genuine machine. The malware project begins inside the visitor OS. To screen a program's action, a sandbox presents snares. These snares can be embedded straightforwardly into a system to get notices (callbacks) for capacity or library calls. The issue with direct snares is that the system code should be adjusted, and this can be recognized by malware or meddle with element code era (unloading). Most often, sandboxes snare framework calls to screen the association between a project and the working framework (Bailey 2007).

That is entirely stealthy, particularly for client mode malware. Besides, framework calls catch all collaborations between a system and its environment (e.g., pursuing of records, registry keys are composed, and deliverance of system activity). The key issue with trapping framework calls (or library capacities) is that the sandbox is incognizant in regards to everything that happens in the middle of calls; that is, a conventional sandbox cannot see any direction that the malware executes between calls. This is a noteworthy drawback that malware creators can target; and they do as such with slowing down code, which is code that keeps running between framework calls.

5.2.1 Delay onset

Onset delay is one of the techniques used by malware developers to evade detection in sandbox analysis. As explained by Lakhani (2015), this technique is the same extended sleep call. Attackers delay execution of malware programs, which making it harder for the sandbox to detect the malware's behaviors. However, some techniques can be used to force lackluster code into instantaneous execution. After the malware commences execution, Sandbox techniques can then be used to detect and analyze its activity. Some of the techniques used include static malware analysis, which involves analyzing the malware's binary file. On the other hand, dynamic malware analysis involves watching and keeping track of malware activity. That is in line with Lakhani's (2015) observation of sleep calls being used to evade malware detection capability of sandboxes.

5.2.2 Diagnosing of the sandbox

Another technique used is the diagnosing of the sandbox. "This is an ingenious technique whereby the malware scans a system and virtual machines characteristics to identify the sandbox environment" (Calhoun, 2015). Sandboxes have been known to use techniques to hide their environment to fool malware. Yet these workarounds are usually short-term. Static code analysis provides the most efficient detection capabilities. However, malware programs have resorted to dynamic behavior, which makes it hard for sandboxes using static analysis to detect them (Calhoun, 2015).

5.2.3 Checking for human pulse

Additionally, malware programs look for human interaction within the system before executing. "Checking for human pulse is a technique that identifies the users of the system as one of the frailest links in the chain of security using sandboxes in malware detection (Calhoun, 2015)." Human activities such as mouse motion and button clicking are known to pose a challenge of detection by the virtual environment because they are hard to replicate. Malware-detecting, an unusual behavior in the system, exits its execution to avoid being detected by the sandbox.

5.2.4 Stalling code

Christopher Kruegel, chief scientist at Lastline, explains why he thinks the sandbox cannot be considered a silver bullet in malware detection. The admonition basis is the fact that malware authors have devised ways to evade sandbox detection in many methods: stalling code – this is the same method seen from previous authors whereby malware delays their execution so that the sandbox times out (Mimoso, 2015). However, Messmer (2013) notes that the malware undertakes in some useless computation that gives the impression that an activity is executing. This vulnerability is a flaw with the sandbox; another technique that she discusses is a “blind spot in the sandbox implementation” (Messmer, 2013). A sandbox injects hooks into a program to get notifications for routine calls. The hooks introduce a vulnerability in that the program needs alteration, a weakness that can be detected by the malware or that tampers with unpacking or dynamic code generation. The sandbox will then fail to identify any instruction that the malware executes due to the hooks (Messmer, 2013). Stalling code capitalizes on the failure to run between system calls.

Slowing down code is executed before any malignant conduct – paying little respect to the execution environment. The motivation behind such shifty code is to postpone the execution of malevolent action sufficiently long so that computerized examination frameworks abandon a specimen, erroneously accepting that the project is non-useful, or does not execute any activity of hobby. It is critical to watch that the issue of slowing down code influences all investigation frameworks, even those that are completely straightforward. Additionally, slowing down code does not need to perform any checks.

Slowing down code misuses two regular properties of mechanized malware examination frameworks: First, the time that a framework can spend to execute a solitary specimen is restricted. Normally, a computerized malware examination framework will end the investigation of a specimen following a few minutes. That is so since the framework needs to make an exchange of the data obtained from a solitary specimen, and the aggregate number of tests that can be broken down each day. Second, malware creators can make their code so that the execution takes any longer inside the examination environment than on a real casualty host. Hence, despite the fact that a specimen may slow down and not execute any noxious action in an examination situation for quite a while (numerous minutes), the postponement seen on the casualty host is just a few moments. That is vital because malware creators consider postponements on a casualty's machine as unsafe. The reason is that the vindictive procedure will probably be identified or ended by hostile to infection programming, a mindful client, or a framework reboot.

5.2.5 Others

Bisson (2015) looks at four of the most common evasion techniques employed by malware. A typical malware sample assumes ten evasive behaviors from which four are common: environmental awareness, confusing automated tools, obfuscating internal data, and timing-based observation (Bisson, 2015). This argument shares the same sentiments portrayed by Lakhani (2015) and Calhoun (2015) in the sources previously studied. Confusing automated detection tools allow malware to evade detection by automated technologies, signature-based antivirus software being an example. "Dyreza/Dyre banking malware according to a study published by security analysts at Talos Group: Angel Villegas and Alex Chiu; older versions of the malware encrypted their URLs when connecting with their command and control servers. Its authors have been changing the virus' domain on a regular basis to avoid malware blacklists by using a domain generating algorithm (DGA)" (Raff, 2015).

DGA determines where a command and control server will be at a given period, thus making it difficult to block activities related to the malware. The environmental awareness behavior gives the malware the capability to fully study and understands the fundamental runtime environment of the sandbox system that it targets to attack. The malware can determine the differences between a bare-metal environment and a virtualized environment, as well as other components of the operating system and hard drives. "A research showing that 17% of Car Bank malware samples investigated by a security firm concentrating on real-time analysis of advanced malware, Last line, tried to identify a virtual environment of a sandbox before starting to execute. Timing-based evasion technique is employed by malware when trying to run in response to user actions" (Raff, 2015).

This finding is the same as the one discussed by Lakhani (2015) and Calhoun (2015), whereby a malware will detect human interaction motion before it starts executing. The malware may be scheduled to attack only at given dates or to activate only when the system reboots. The most pervasive kind of point-of-sale (POS) malware is Black POS. It adopts a time-based evasion to a degree that newer versions of the same checks up the system time on the beset device against the time they have hard coded into the executable; a behavior that allows it to remain dormant much of the time while executing in certain periods. Finally, malware can implement confusing data or internal data to execute code that cannot be detected by the sandbox analysis system. Bisson

(2015) explains ROM, a new version of the Backoff POS malware, replaces API names with encrypted values, disregarding certain processes from being construed by using a table of encrypted values while communicating with command and control server using port 443, which hashes the information passed effectively (Bisson, 2015). This poses difficulties for detection systems trying to identify the malicious nature of the ROM malware.

5.3 Countering sandbox evasions

Calhoun (2015) proposes some prevention and treatment tips. Behavioral monitoring of the suspicious malware in a controlled environment is an invaluable technique. However, it may not be sufficient to diagnose malware that alters its identity in real life to evade detection. The history and change of malware activity are crucial to security vendors. It can be used to categorize and track malware strains. Therefore, it forms an important foundation in the malware detection process. "Information about malicious email addresses, suspected websites, and IP addresses should be stored in a comprehensive database. It can be used for analysis and prevention of malware attacks" (Calhoun, 2015).

A suspected malware passing dynamic sandbox analysis should be subjected to full static code analysis, that running deeper in unpacking the code for testing and parsing to analyze all execution paths. This sophisticated disassembly ensures that the actual file is fully analyzed to detect any evasion. Hackers compress their codes and encrypt text strings, but full static code analysis is powerful in revealing hidden malicious IP addresses and websites. Unpacked and decrypted code can then be easily studied to find its family regarding behavior about known malware (Calhoun, 2015). The fingerprinting analysis used by researchers against evasive detection characteristics of malware can be a solution to aid their detection. According to Balzarotti et al. (2010) in a column for IBM Security Intelligence. Security analysts can look after environmental analysis by making environmental artifacts random so that malware will not find the vulnerability, which they typically have exploited. Automatic prevention of Sandbox timeout can occur through code execution, which helps identify when the sandbox stops or behaves unusually.

According to Singh (2015), malware analysts have leveraged sandboxes to automate the analysis of malware. Still, he argues the fact that most prevention sandboxes use only known information. Modern day virus authors seek to evade the widely used virtual machines. Examples include Qemu, CWSandbox, Anubis, VMWare Fusion, VMWare Workstation and Virtual Box. They have come up with anti-VM techniques. Process names – malware rely on process names to identify the presence of a virtual machine. Malware analysts know that processes such as vmusrvc.exe, vboxtray.exe, vmtoolsd.exe, df5serv.exe, and vboxservice.exe among others may be running in a sandbox based on their names" (Singh, 2015). Routines such as Process32First ()/ Process32next () are used to identify and enumerate the processes (Singh, 2015). Registry artifacts can be used to determine the presence of sandboxes the most common registry being HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\Disk\Enum. A sub key value Subkey: "0" is parsed to identify the existence of strings or substrings such as qemu, Xen, VMware and so on (Singh, 2015).

Malware attacks are known to use module names (Singh, 2015). Sandboxes introduce modules into a process to log the activities that the process executes. Module names can be used by malware to detect the presence of sandboxes. Targeted module names include sbiedll.dll and dbghelp.dll. A virus sends a routine call, GetModuleHandleA () on a process, a loaded module will return a certain value (Singh, 2015). The malware then unloads the routine using FreeLibrary () which impedes the sandbox from logging any activity associated with the malware (Singh, 2015).

Other techniques that can be leveraged to determine the presence of a sandbox include backdoor detection, long opcode instructions, the number of cores, device information, data structures, network adapter MAC address, file system artifacts and sensitive instructions (Singh, 2015).

5.4 The future of sandbox techniques and malware evasions

Raff (2015) looks at the artifices exhibited by Dyre malware in evading sandboxes. He maintains that the malware's success in evasion of sandboxes is a clear indication of the inability of sandboxing as a standalone mechanism, deeming it to be an incomplete security approach. "Dyre Wolf is a banking Trojan that stole $1million from corporate bank accounts by getting around a 2-factor authentication system" (Raff, 2015). Following a close examination of the Trojan, in their research lab, they noted some changes from the previous Tinba Trojan. "The version can avoid malware sandbox analysis by checking the number of cores in the machine" (Vijayan, 2015). Once the malware detects a single core, it stops executing. Many sandboxes, configured with a single processor having a single core, allow for the effectiveness of the mechanism employed by the malware in avoiding being analyzed by the sandbox system. The malware, unlike others, used only a single anti-sandboxing artifice, possibly because the technique or check was the solution to remaining undetected by sandboxing methods available. Despite maintaining its communication path, Dyre changed user agents to evade detection by signature-based detection systems. The author of the malware made some minimal alterations to the way the malware conducted itself once in the system to avoid eliminate detection. "Dyre combined with Upatre, a downloader malware that employed a very strange and distinctive user agent that had Mozilla/5.0 typo" (Vijayan, 2015).

However, the newer version has more generic user agents, which make it deceiving as it now easily evades signature-based detection. There has been an alteration of the Upatre downloader's communication path to one that does not follow any naming convention based on common features but employs an obscured one. That is enough proof of the inefficiency of sandboxing as standalone systems in security systems (Vijayan, 2015).

The ability to observe, analyze and detect evasive malware should thus incorporate non-deterministic machine learning techniques and the analysis of communication traffic regularly, (Vijayan, 2015). The longevity of malicious URLs is very brief during a Drive-by-Download (DBD) attack as attackers have adapted to changing their URLs continuously to avoid blacklist-based detection. Cloaking techniques avail contents based on the User-Agent HTTP header or IP addresses of the user requesting the contents of the page. Then follows identification of the accessed user as client/crawler honeypot of researchers or security organizations, the malicious servers deliver fake benignant contents to mislead the client/crawler honeypot to evade inspection (Balzarotti et al., 2010).

5.5 Recommended Techniques

5.5.1 Behavioral observation

Solutions to malware detection and prevention should combine behavioral observation with sophisticated code analysis to effectively uncover new malware threats and adaptive sandbox evasion techniques. For checking of Malware Sandbox evasion countermeasures, close attention has to be taken to determine any slight differences portrayed by suspicious programs against their benign software counterparts.

5.5.2 Treatment of environment sensitive programs as malware

Any environment-sensitive program should be treated as malware because malware identifies a particular malware sandbox and stops executing before detection. Despite it being a very crucial technique in the effective analysis of a large amount of malware, Malware Sandbox analysis has to address some grave issues to avoid vulnerabilities that have been exploited by malware authors.

5.5.3 Concentrating on servers issuing commands

The fact that malware samples have been changing their behaviors based on the information they receive from C&C servers calls for sandbox analysis to concentrate not just on the behaviors of the malware themselves, but also on the servers issuing commands with which the malware update themselves to confuse the detectors. “Sandbox analysis, which leverages a dummy client, is a novel malware analysis technique that utilizes a dummy client, which is an automatically produced script to interact with the remote servers instead of the real malware sample” (Messmer, 2013). The malware sample runs through the sandbox connected to an Internet emulator and the Internet. Monitoring of communication traffic in the sandbox facilitates in filtering out the high-risk ones. The dummy client then interacts with the C&C servers using the traffic during which it collects responses from the servers. “The responses are then injected into the Sandbox Internet Emulator, which utilizes them in improving observability in the malware sandbox analysis” (Messmer, 2013). By leveraging lightweight dummy clients instead of real malware samples, observability is enhanced by repeatedly observing numerous remote servers simultaneously without compromising effectiveness and efficiency.

Kasama (2014) has conducted extensive research on malware analysis leveraging sandbox behaviors. Attackers leverage various techniques to execute malware on a target system. According to Takahiro (2014), the techniques can be categorized into two: exploiting program vulnerability, and exploiting the human vulnerability. For exploitation of server apps and operating systems, malware actively scan and identify a vulnerable computer.

5.5.4 Use of honeypots

Malware samples may be collected using antivirus programs, security appliances, or using a honeypot, which acts as a decoy to lure malware into attacking them as vulnerable hosts. The honeypot traps and then exploits the codes. Malware analysis aims to provide exhaustive information about behaviors and operational mechanisms of malware for the development of effective countermeasures. “It can be sandbox (dynamic) analysis or static (white box) analysis. The latter is a traditional technique that analyzes malware binaries without execution” (Kasama, 2014). It works by unpacking the binaries and analyzing their behaviors and functionalities at the assembly level.

5.5.5 Unpacking malware before analysis

Malware authors have found a way to obfuscate malware by using packing tools such as the Ultimate Packer for eXecutables (UPX). Analysts are therefore expected to unpack the packed malware samples beforehand using methods such as memory dump (Singh 2015). Sandbox analysis executes the malware sample in a sandbox where the behaviors of the malware are analyzed and monitored. This method does not have any difficulty handling packing and code-obfuscation techniques. Implementation occurs in an automated fashion (Singh, 2015).

5.5.6 Analyzing malware while offline

The greatest pitfall of the sandbox analyzer is that it can only monitor and analyze behaviors of malware only when it is running. “Sandboxes can be totally isolated or those with an internet connection” (Singh, 2015). With the former, malware can be analyzed without any effect on the internet. Examples include Norman Sandbox and NICTER Microanalysis. However, as malware communicates with remote servers on a network such as a download and C&C servers, their behavior can dynamically change to make their observation more difficult. The latter connects the sandbox with the Internet. “Examples are Anubis and CWSandbox. The behaviors corresponding to actual remote servers on the internet can be observed” (Singh, 2015). But the sandbox risks inducing secondary infections in case their attack trickles out of the sandbox. Therefore, sandboxes should run evaluations when the computer is offline to prevent these.

5.5.7 Observing, containing and efficiently giving back results

Malware Sandbox analysis has three important properties: observability, containment, and efficiency. Observation involves analyzing malware behaviors in consideration. Containment suppresses leakage of important information and prevents an executed sample from infecting or attacking a remote host outside the sandbox environment. Finally, efficiency involves the provision of results with sufficient analysis within a reasonable period (Singh, 2015). Some countermeasures have been advanced to deal with malware attacks based on malware analysis.

"Generating malware signatures – open source software packages such as ClamAV and network-based IDs such as Snort analyze malware samples, extract characteristic strings while generating signatures for detection of the malicious codes" (Singh, 2015). Since malware authors typically use packing techniques and polymorphic shellcodes to avoid detection-based signatures, antivirus programs are leveraging behavior-based malware detection engines.

A Herder is a botnet controller that funnels the malicious activities of compromised hosts via C&C channels. Detection of computers infected with a bot is possible by detecting the C&C traffic as malware is known to communicate with remote host servers using C&C messages. Through blacklisting, determination of malicious DNS names, IP addresses, and URLs, occurs from analyzed samples, which form a pivotal basis as a blacklist for filtering access to harmful hosts and detecting infected ones. Botnet takedown – closes the C&C communication. Destruction of the botnet occurs, and potential victims are saved from being affected by cyber-threats. Consideration of technical and legal action against the botnets following collaboration between technology firms and law enforcement may be necessary (Singh, 2015).

Singh (2015) further looks at evading sandbox analysis. There are two categories of the evasion techniques. The first group involves making it difficult for the sandbox to decipher malware activity and the second one involves detecting sandboxes as the first step of evasion (Singh, 2015). After detection of the sandbox, malware can then suspend execution to avoid detection. One example is a Trojan horse, and time condition that activates a date otherwise called time bomb. It activates in some scheduled periods. A botnet changes its behaviors according to remote servers that direct it, thus performing malicious activities in line with C&C messages received from corresponding C&C servers (Singh, 2015). Then they download the additional binary from malware download servers and run them on the host machines. Some malware has randomized their behavior, which changes to evade analysis and detection an example being 'Conficker' which generates a DNS name by use of a pseudorandom number generating algorithm, making it almost impossible to list all accessed DNS names of malicious servers (Singh, 2015).

5.5.8 Using non-signature based detection

Many operating systems such as Windows have firewalls and antivirus software for detecting and neutralizing malware threats. As a result, some malware searches and kill these software processes in a bid to evade detection. This protection software uses signature-based detection engines to detect malware activity. Malware authors have devised ways of evading such detection techniques. "Some use a binary file as a signature trying to hide the characteristic binaries with compile time packing algorithms, and when leveraging a registry entry name or directory/file name as a signature, the malicious programs change their behaviors during compilation to avoid detection" (Balzarotti et al., 2010).

Polymorphic shellcode engines have also been utilized to evade signature-based detection in IDSs (Balzarotti et al., 2010). These engines produce various forms of the same initial shellcode by hashing the payload with diverse random IDs and by appending to them a self-decrypting function. "As mentioned from previous articles, fast flux, and domain generating algorithms are also used by malware to evade detection by adopting lists of malicious domains, IP addresses and URLs" (Balzarotti et al., 2010).

6.0 Conclusion

Malware has been the most critical culprit in cybercrime. With the evolving nature of malware, it has become almost impossible to contain or analyze all the types manually using reverse engineering methods available. Malware sandboxes have registered good results and potential in curbing the problem; however, authors of the malware have found ways of bypassing the analysis measures put in place. It is sound to say that sandboxes no longer provide or guarantee the needed protection that required to be afforded computing systems to prevent against malware attacks. It is imperative to at least know the appropriate sandbox analysis tools available in the market in order to make a wise selection on the vendors with effective sandbox based malware tools.

Virtualization over the years has been an effective tool used to observe all malware behavior without putting an actual system in jeopardy by constraining the effects it causes to a virtual world. Currently, attackers have advanced to detect when the malware they have authored is running on actual systems or in Sandboxes. A clear understanding of the malware programs, their behaviors, and the nature of sandbox analysis tools is important in understanding the measures needed to realize higher efficiency and better observability of the sandbox analysis method that makes it harder to be recognized by the malware.

Detection of the current malware threats demands more of a comprehensive approach, not just a unilateral push for the use of sandboxes. Virtual environments (sandboxes) should just be one of the many mechanisms deployed to protect systems against malware analyzing and prevention across multiple platforms. Attackers are merely picking up on any gaps in the defense strategies being used to counter them and using these gaps to slip through the complex mechanisms rendering them simply ineffective. It is, therefore, necessary that multiple security techniques be combined so that they work in a collaborative fashion. These combinations will then be able to detect malware whether in execution or a dormant state. That will help a great deal in the concealment of the gaps that malware authors painstakingly look for in all defense systems. Without any gaps to slip through, malware shall be easily prevented, detected and countered similar to the past, when only a few could evade detection. The concealing of the gaps must coincide with a deep analysis of the workings of any new malware to fortify the existent defenses (Singh & Bu, 2015).

Analysts have recognized that rootkit technologies are what present day malware coders' use due to its tenacity in avoiding detection from both antivirus programs and computer users. It was imperative that new antivirus methodologies were needed with several technological enterprises coming up with some dependable options. The general objective is to invest in solutions that have the ability to not only detect harmful or illegal malware in a computer network but also ensure that not illegal access occurs from an external host such as in the case of a hack. In addition to that, not only should these anti-virus systems protect the software in the computer system but hardware as well.

7.0 Appendix A: Recommendations

I. Target consumer: Huge Enterprises

The first recommended solution in the market is for huge enterprises that have volumes of technological solutions in its day to day entrepreneurial activities (FireEye, 2015). Companies involved in the mass manufacturing of a product, encompass factory-level infrastructure can invest in this option as it is well within their capacity.

Recommendation;

FireEye, a security software organization that manufactures a virtual machine based anti-virus platform. It has recently made headlines by collaborating with Parsons, a frontier company in the computer hardware industry for the past 70 years. Together, these to technological enterprises have come together to provide an infrastructural computer system design that comes with the FireEye security system platform factory installed. The system also provides industrial information management technologies that can be assimilated into an organization's entrepreneurial activities (Parsons, 2015). It is a secure system with the capacity to analyze and isolate any detected malware or cyber-attack in real time. Not only does it protect all intellectual property, information or physical damage but also facilitates the execution of services and alternate innovative delivery methods.

II. Target consumer; Small Scale Enterprises

There some small scale enterprises that may find investment to the infrastructural level not cost effective.

Recommendation;

BlueVector systems is an enterprise that noticed this niche in the market and have come up with a product friendly solution for small scale companies (Kuloor, 2015). Their security system is made up of a sensor and RFID network structure that not only provides security for data and software entities but automates the business environment. It has the capacity to analyze, detect and eliminate malware that may have evaded the conventional anti-virus software with its analysis of all the enterprise-scale information inclusive of inventory records, reorder entities and online communications.

III. Target consumer: Distributed Conglomerates

There are conglomerates that have and a lot of information flowing in its networks, permanently connected to the internet. As such, there are security software manufacturers that have products that apply cloud technology concept. These solutions come with two key advantages. Due to the colossal amount of information that may be flowing through an organization's networking system, cloud technology is a trending storage technology solution that comes highly recommended (Zcaler, 2015). The risk comes in trusting that the information stored in cloud solutions is secure from both illegal access and malware especially with its virtual existence being on the internet.

Recommendation;

Zcalar is a technological company that has come up with an anti-malware program that applies SSL-encrypted methodologies to sieve every single byte of information flowing in and out of an organization's computer network system and cloud storage solutions. It outshines the rival antivirus software solutions with its unique capacity to correlate and apply granular directives by file type, size, and authorized user. In the event malware is detected in the network, Zcaler isolates the intrusion in a cloud sandbox for deletion. AhnLab is another company that has incorporated cloud-based and local network analytics in it Malware Defense System. Their solution comes highly recommended with its ability to protect computer systems from malware, illegal access to secure files and optimize the business environment technology wise (AhnLab, 2015). In other terms, it is an information management system that monitors and protect not only the data flowing in the local network of an enterprise but that coming in and out of the internet and its cloud storage entities. It is a solution that protects technological hardware property such as servers, host computer, endpoints as well as software entities such as network and cloud resources. Any detected malware is guaranteed from the computer system in a virtual machine or sandbox depending on the scale of attack.

8.0 References

AhnLab. (2015). *AhnLab MDS; Multidimensional Analysis of Malware.* Pangyoyeok-ro, South Korea: AhnLab, Inc.

FireEye. (2015). *M-Trends 2015; A view from the frontlines.* California: MANDIANT.

Kuloor, R. (2015). *Blue Vector Systems; From edge to infrastructure.* Palo Alto: BlueVector Systems.

Parsons. (2015). *Parsons Corporate Report.* Pasadena: Parsons Corporation.

Zcaler. (2015). *Zcaler Web Security*. San Jose: Zcaler Inc.

Lakhani, A. (2015). Malware Sandbox and Breach Detection Evasion Techniques. Dr. Chaos. Retrieved 28 October 2015, from http://www.drchaos.com/malware-sandbox-and-breach-detection-evasion-techniques/.

Balzarotti, D., Marco, D.c Karlberger C., Krugel, C., Kirada, E. & Vigna, G. (2015). Efficient detection of split personalities in malware. University of California, Santa Barbara, California. (2010) Web.17August, 2015. Retrieved from https://cs.ucsb.edu/~vigna/publications/2010_balzarotti_cova_karlberger_kruegel_kirda_vigna_SplitPersonality.pdf

Bisson, David. (2015). The four most common evasive techniques used by malware. Tripwire, Inc., April 2015. Web. 18 August 2015. Retrieved from http://www.tripwire.com/state-of-security/security-data-protection/the-four-most-common-evasive-techniques-used-by-malware.

Calhoun, Pat. A glimpse at the latest sandbox evasion techniques. Security Week, January 2015. Web.18 August 2015. Retrieved from http://www.securityweek.com/glimpse-latest-sandbox-evasion-techniques.

Kasama, Takahiro. (2014). A study on malware analysis leveraging sandbox evasive behaviors. Retrieved from http://kamome.lib.ynu.ac.jp/dspace/bitstream/10131/8598/1/kasama_takahiro-thesis.pdf

Lakhani, Aamir. (2015) Malware Sandbox and breach detection evasion techniques. Dr. Chaos.Com, May 2015. Web. 17 August 2015. Retrieved from http://www.drchaos.com/malware-sandbox-and-breach-detection-evasion-techniques.

Messmer, Ellen. “Malware-detecting ‘sandboxing’ technology no silver bullet.” Networkworld, March 2013. Web. August 2015. Retrieved from http://www.networkworld.com/article/2164758/network-security/malware-detecting--sandboxing--technology-no-silver-bullet.html

Mimoso, Michael. (2015). Malware evasion techniques dissected at black hat. Threatpost. August 2015. Web. 18 August 2015. https://threatpost.com/malware-evasion-techniques-dissected-at-black-hat/101504

Raff, Aviv. (2015) New dyre version –yet another malware evading sandboxes. Seculert, 30 April 2015. Retrieved from http://www.seculert.com/blog/2015/04/new-dyre-version-evades-sandboxes.html >

Singh, Sudeep. (2015). Breaking the sandbox. Exploit-db.Com. Retrieved from https://www.exploit-db.com/docs/34591.pdf

Vijayan, Jai. (2015). Dyre Trojan adds new sandbox-evasion feature.” Dark reading, Retrieved from http://www.darkreading.com/vulnerabilities---threats/dyre-trojan-adds-new-sandbox-evasion-feature/d/d-id/1320244

Messmer, Ellen. “Malware-detecting ‘sandboxing’ technology no silver bullet.” Networkworld, March 2013. Web. August 2015. <http://www.networkworld.com/article/2164758/network-security/malware-detecting--sandboxing--technology-no-silver-bullet.html>

Kruegel, C. (2015). Evasive Malware Exposed and Deconstructed | USA 2015 RSA Conference. Rsaconference.com. Retrieved 6 November 2015, from https://www.rsaconference.com/events/us15/agenda/sessions/2022/evasive-malware-exposed-and-deconstructed.

Constantin, L. (2015). *New APT malware monitors mouse clicks to evade detection, researchers say*. *Computerworld*. Retrieved 6 November 2015, from http://www.computerworld.com/article/2496123/security0/new-apt-malware-monitors-mouse-clicks-to-evade-detection--researchers-say.html

M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant, “Semantics-Aware Malware Detection”, in Proc. of IEEE Symposium on Security and Privacy, 2005.

M. Christodorescu, S. Jha, and C. Kruegel, “Mining Specifications of malicious behavior”, in Proc. of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, 2007.

Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated Classification and Analysis of Internet Malware. In: Proceedings of the 10th International Symposium on Recent Advances in Intrusion Detection (RAID) (2007).

Chen, X., Andersen, J., Mao, Z.M., Bailey, M., Nazario, J.: Towards an Understanding of Anti-Virtualization and Anti-Debugging Behavior in Modern Malware. In: Proceedings of the 38th Annual IEEE International Conference on Dependable Systems and Networks (DSN) (2008).