Malware Analysis: An Introduction

Dennis Distler

Malware Analysis: An Introduction

GSEC Gold Certification

Author: Dennis Distler, distlerdennis@gmail.com

Adviser: Charles Hornat

Accepted: December 14, 2007

Table of Contents

2. Abstract

I am submitting this abstract to fulfill the technical paper requirements for the GSEC Gold Certification. The paper will be a detailed introduction of malware analysis for security professionals. This paper would be an excellent fit to the Security Essentials track by providing information to assist in the gap that exists in the field, as malware issues are common in computer security today.

The paper will begin with an introduction describing the various types of malware. Types of malware described include Virus, Worms, Trojans, Adware, Spyware, Backdoors and Rootkits that can disastrously affect a Microsoft Windows operating system.

The second section will discuss the basics of an incident response plan. A brief description of the steps of an incident response plan will be described. The role of malware analysis and what steps it pertains to in an incident response plan will be described.

The next section will discuss the goals to be accomplished by performing malware analysis. During this section, a fictitious worm will be described to provide examples of the goals behind malware analysis.

After a discussion of goals, this section will identify and discuss two basic types of malware analysis: code (static) and behavioral (dynamic) analysis. Basic static analysis

techniques will be discussed, such as scanning with anti-virus software, looking at the malware with a hex editor, unpacking the malware, performing a strings search and disassembling the malware. This section will also identify general behavioral analysis techniques such as network traffic analysis, file system, and other Windows features (services, processes, etc.).

The next discussion will be of the tools used to perform analysis. These tools will include VMware, tcpdump/windump, Sysinternal tools, disassembles, servers, netcat and others critical to the success of the analysis.

The following section will describe various types of malware acquisition. During this section, two common methods of how end user hosts become infected with malware will be discussed. Using honeynets to acquire malware will be the next topic discussed. Finally, the topic of using search engines to gather malware will be briefly discussed.

After discussing malware acquisition for malware analysis, a methodology will be presented for performing malware analysis. The steps will be as follows: how to build a sandbox environment, how to baseline the “victim” system, how to execute the malware, how to gather the data to be analyzed, and how to analyze the data.

The next step will be to perform an actual malware analysis. This will be a real world, practical example. Each step will be documented to identify how to analyze a piece of malware.

Following the analysis of the malware, a discussion of defenses that can be utilized to defend against this particular malware will be identified. The defenses for use against the malware will include firewall rules, Intrusion Detection Systems (IDS) rules, web filtering and host base intrusion prevention systems (HIPS).

Finally, the paper will conclude with credit given to those who have provided both help and support during the evolution of this paper.

3. Introduction

Walking into the office Monday morning after a long, three-day weekend the network administrator of GIAC Rock, Shannon, is immediately bombarded with complaints that one of the Windows server's is performing slowly. After a quick analysis of the server, she is perplexed as to why the server's performance is failing. Shannon begins performing a detailed analysis by eliminating hardware failures. Once hardware failures have been eliminated, Shannon begins to dig deeper into the operating system when she come across files that were not initially on the server when she left the office last week. Shannon now has

a sinking feeling when she realizes the server is compromised by some sort of unknown malicious software … better known as malware.

Knowing that malware has been detected on the system, a million thoughts race through Shannon's mind. However, the primary thought is that there is no way she can leave malware on the system. In a perfect scenario GIAC Rock would have an Incident Response and Disaster Recovery Plan in place that would allow her to return the server to pre-infected production condition in a controlled methodology.

The purpose of this paper is two fold: to help Information Security professionals, such as Shannon, perform malware analysis and to satisfy GSEC gold requirements. This paper will also serve as a guideline for the reader to perform malware analysis by providing definitions, tools to use, and real world examples to the reader with enough information to successfully perform malware analysis.

It should be noted: if an Incident Response plan is not already in place, do not attempt to create one during an infection. Rather, remove the infected server from the network. Create a plan to systematically return the infected server to its pre-infected production condition before beginning the recovery process. Incident response is not a responsibility that a single person can handle. Recovering a compromised server in a haphazardly fashion can create more system issues and do more damage then the initial compromise.

When discussing malware it is vital for the reader to have an understanding of cost of malware infections that occur in organizations. According to Computer Economics 2007 Malware Report, malware infections in 2006 cost $13.3 Billion dollars. Although the trend over the last two years is a down turn in the cost of malware infections, the cost of malware should concern companies of any size. The report states two factors for the reduction in malware infections cost, the wider spread deployment of Anti-Malware applications, and malware targeted at specific organizations and people (Computer Economics Online, 2007).

Before discussing malware analysis, it is important to identify key terminology that will be used through out this paper. Below is a list of terms and definitions the reader of this paper should be familiar with:

Viruses (Merriam-Webster Online, 2007) – a computer program that is usually hidden within another seemingly innocuous program and that produces copies of itself and inserts them into other programs and usually performs a malicious action (as destroying data)

Worms (Merriam-Webster Online, 2007) – a usually small self-contained and self-replicating computer program that invades computers on a network and usually performs a destructive action

Trojans Horse (Merriam-Webster Online, 2007) – a seemingly useful computer

program that contains concealed instructions which when activated perform an illicit or malicious action (as destroying data files)

Spyware (Merriam-Webster Online, 2007) – software that is installed in a computer without the user's knowledge and transmits information about the user's computer activities over the Internet

Adware – software installed that provides advertisers with information about the users browsing habits, thus allowing the advertiser to provide targeted ads.

Backdoors (Skoudis and Zeltser, 2003) – Bypasses normal security controls to give an attacker unauthorized access.

Rootkits (Skoudis and Zeltser, 2003) – Trojan horse backdoor tools that modify existing operating system software so that an attack can keep access to and hide on a machine.

Sniffers – an application used to monitor and analyze network traffic.

Reverse Code Engineering (Eilam, 2005) – the process of disassembling software to reveal how the software functions.

Disassemblers (Eilam, 2005) – programs that take a programs executable binary as input and generate textual files that contain the assembly language

code for the entire program or parts of it.

Debuggers (Eilam, 2005) – programs that allows software developers to observer their program while running it.

Decompiler (Eilam, 2005) – a program that take an executable binary file and attempts to produce readable high-level language code from it.

Overall, malware analysis is an interesting, exciting, and challenging field of computer security research. The complexity of malware analysis is only one area of the security profession that is constantly evolving.

4. Incident Response and Malware Analysis

According to techtarget.com, Incident Response is an organized approach to addressing and managing the aftermath of a security breach or attack (also known as an incident). The goal is to handle the situation in a way that limits damage and reduces recovery time and costs. (Techtarget Online, 2007).

As noted earlier, Incident Response Plans should not be created during a security incident nor should one person be assigned to develop an Incident Response Plan. Incident response should be the responsibility of different members from different groups in an organization. Management buy-in is essential for an Incident Response Plan to work and an

Incident Response team to be successful.

SANS created, through a consensus process a six step incident handling plan one needs to follow to prepare for and deal with a computer incident (SANS, 2007). The six steps of the incident response process follows:

Preparation

Identification

Containment

Eradication

Recovery

Lessons learned.

By following these six steps, an organization can recover from an incident with as little time and money lost to the business as possible, while also ensuring that the incident will not happen again.

During the preparation phase of the Incident Response Plan, there are several key items to be completed for an Incident Response Plan to be successful. The reader will see why Incident handling can not be the responsibility of one person (GIAC, 2007).

First, establish policies to identify who is responsible for responding to incidents. These policies should protect both the incident handler and the organization. Next, build relationships with *all* key players. These key players will be from Human Resources, Legal Consul, Information Technology, Security (both physical and computer), Public Relations and possibly law enforcement (GIAC, 2007).

Next, build an incident response *jump kit*. This jump kit will consist of hardware, software, call list, offices supplies, and possibly clothes. The jump kit should be well organized and available to the incident handlers at all times (GIAC, 2007).

The jump kit should also include incident checklist and incident communications plan. The incident checklist should be used to keep incident handlers on track as well as ensure nothing is missed when dealing with an incident. The communications plan is used to determine who and how the appointed personal will communicate with effected users, media, and law enforcement agents (GIAC, 2007).

At this point in the preparation phase, perform threat modeling to identify the types of incidents in which the incident response team will be responsible. If the organization is comprised of all Windows hosts, then threats to Linux hosts can not exist in the organization and should not be addressed. Be sure to include possible types of threats that may occur. For example, one of the authors' main data centers is in the flight path of one of the world's

busiest airports located in a major landlocked metropolitan city. While the possibility of an airliner crash is a viable threat to the building, a tsunami would not be. By using threat modeling, incidents that can occur are easily identified. This makes planning for incidents easier when real threats are known to the Incident Response Team.

After identifying the threats, the Incident Response team can be built. This team is responsible for all incidents an organization faces. Identification of the team members should be public so that the organization knows who to contact in the event of a suspected incident (GIAC, 2007).

The final part of the Preparation phase of an incident response plan, is to practice. The Incident Response team should continue to practice their skills so that improvements can be made to the Incident Response Plan (GIAC, 2007).

The next step in an Incident Response Plan is the Identification step. The identification step is when the Incident Response team must identify what is causing the incident. Although all steps are critical, this step is where the reader would perform malware analysis on an unknown piece of malware. All information learned from the malware analysis will be used in later steps in the Incident Response Plan.

During this step, it is critical that outside influences are not allowed to cloud the incident handler's judgment. People will supply the incident handler with all types of

conjecture as to what the person think happened, and sometimes it is easy to get sucked into their excitement. Don't do it. Gather all of the facts and make judgments based on those facts (GIAC, 2007).

While performing malware analysis during an incident, do not overlook important information that the malware analysis is providing. During an incident, panic will often set in. Do not let this happen. The information gathered during malware analysis is critical to protect the organization from more damage.

The containment step of the Incident Handling Plan is when the organization begins to deal with the incident. Information gathered during the malware analysis will be used in the containment step.

During the containment step of Incident Response, remove infected host or hosts off of the organizations network by unplugging the network cable, while leaving the system powered on. As infected hosts are been removed from the organizations network begin to protect the rest of the network. This would include WAN links and Internet connections by using access-list to deny traffic to and from infect host, subnets, or locations.

During the containment step, do not power off infected hosts. The incident handler should attempt to preserve evidence in case of legal action. When preserving evidence, make sure clean binaries are used and everything is documented. In some cases, it is inevitable

that a performed task will change something on the system. Be prepared to explain what changed and why that action was performed (GIAC, 2007).

Once the incident is contained, the next step in Incident Handling is to eradicate what is causing the incident. For this paper, it is malware. During this step, the clean up process will begin. When cleaning up from an incident, a critical few words of caution must be said.

In some cases, eradication of the attack is possible without having to rebuild the system. In most cases, though, especially with malware or rootkit attacks, the only way to truly be assured that eradication is successful is to perform a complete rebuild of the system. If this is the case, make sure that the media used for rebuilding or the backups being used for rebuilding of the infected system are not compromised as well. When Code Red attacked in June of 2000, many companies attempted to recover their systems from backups only to find that the backups of the systems were also infected with Code Red (GIAC, 2007).

After a system is considered cleaned per the organizations policies, perform a system vulnerability analysis. Once the system vulnerability analysis reports clean, it is ready for the next stage of the Incident Handling process: the recovery step.

With the systems cleaned from the incident, the next step is recovery. During this step, the system will be placed back in production and monitored for any signs of possible re-infection (GIAC, 2007).

During recovery, ensure that personal with proper authority authorize the system to be placed back into production. Also ensure that system administrators and end users of the infected system have tested the system and applications before placing the system back into production.

After the system has been placed back into production, continue to monitor for signs of possible re-infections. Use tools such as firewall logs and Intrusion Detection System to detect for signs of re-infection. Any sign of possible infections should be investigated immediately.

The final, most critical, and often overlooked step of incident response is the lessons learned step. All though this step is not exciting as the other five steps in incident response, more will be learned in this step then any other step.

When performing the lessons learned step, complete all incident documentation and present findings to management. Look for ways to improve both the technical and administrative process of incident handling. Finally, have a clearly defined plan to implement the lessons learn (GIAC, 2007).

For example, a host on the GIAC Rock network is compromised with a new, undetected, piece of malware. After a thorough analysis, it is discovered this malware, when executed, installs an SMTP engine that sends out Spam messages. This particular malware

was installed by an unsuspecting user who was victim of client-side attack. The name of the new malware is the *Loudpool* worm. Having an understanding of how the *Loudpool* malware functions is absolutely and essentially critical to the next goal; building defenses to protect the GIAC Rock network against *Loudpool.*

Following the GIAC Rock Incident Response Plan, the first phase during the *Loudpool* outbreak will be to deny all outbound SMTP traffic except for the identified SMTP servers that sends and receives external email, assuming that these servers are not compromised. By blocking at the firewall, containment of the *Loudpool* worm has begun. The next step during the containment phase will be to have all infected host or hosts network cable's removed, with the system left on. An attempt to identify the IP address of the server with the infected file is made so that the server can be blocked at the firewall to reduce the possibility of re-infection of the *Loudpool* worm. If organizational policy permits, attempt to contact the server administrator of the infected server. At this point, the *Loudpool* worm should be contained only to the infected host or hosts.

GIAC Rock can now start eradicating the *Loudpool* worm from all infected host. During the eradication phase of the worm GIAC Rock may have to delete files, registry keys or even possible rebuilding the infected host or hosts. During the eradication phase, GIAC Rock system administrators will also have to make a determination of when the malware infected

the compromised host or hosts, as well as if the host backups' are infected with *Loudpool*. Since *Loudpool* was not caused by any application vulnerability, patching is not required to protect against this piece of malware.

With the *Loudpool* worm eradicated, GIAC Rock can now move to the recovery phase of their Incident Response Plan. First, when the anti-virus vendor has released a signature for the *Loudpool* worm, the updated signature should be installed on all hosts.

According to GIAC Rock's Incident Response Plan, the CIO is responsible for determining when infected servers can be brought back on-line. During this phase, an IDS rule will be deployed that detects any host attempting to send SMTP traffic out of the GIAC Rock network. Firewall logs will also be closely monitored during the recovery phase for any unauthorized host sending SMTP traffic.

Although the *Loudpool* incident was fictitious, it is critical to remember that when building defenses against malware, or any type of attack, to use the Defense-In-Depth philosophy. In this example, the GIAC Rock Incident Response Plan was used for guidance during the incident. While the Incident Response Plan provided the plan for dealing with the *Loudpool* worm, several technologies such as access control list, anti-virus, IDS and log monitoring were used to contain, eradicate, and recover from the incident. In this example, it was not demonstrated but it is always important to remember regarding malware that

malware can and will morph to avoid defense mechanisms that are put in place.

When dealing with incidents, and if the organizations policies permit it, consider securely sending newly discovered malware along with all information detected during the incident to the organizations anti-virus vendor. Give great consideration to submitting the piece of malware to community resources such as Castle Cops, SANS Internet Storm Center, Bleeding Threats, and Shadow Server.

Now that there is a basic understanding of how the Incident handling process is outlined, and how malware analysis fits into the Incident Response Plan, the goals of malware analysis will be discussed.

5. Goals of Malware Analysis

Now that there is an understanding of how malware analysis fits into an organizations' Incident Response Plan the next step can be discussed. Before performing malware analysis, a goal of what is trying to be accomplished must be set.

When it comes to fighting malware, you may be asking as a security professional, "Why would I need to perform malware analysis? I don't work for an anti-virus vendor." If you are responsible for the security of a network, at some point in your career you will most likely have to perform malware analysis. With malware becoming target specific towards financial

gain over the last two years (Computer Economics Online, 2007) more malware is in the wild with less chance of Anti-Virus or Anti-Malware applications detecting the malware.

In fact, the author personally knows of an individual who was not in IT at the time who encountered this problem. This person is now a Network Security Manager for a very large international corporation. His career started by dissecting the "I Love You" virus and building defenses to eradicate that virus from the companies' global network.

The goal of malware analysis is to gain an understanding of how a specific piece of malware functions so that defenses can be built to protect an organization's network. There are two key questions that must be answered. The first: how did this machine become infected with this piece of malware? The second: what exactly does this malware do? After determining the specific type of malware, you will have to determine which question is more critical to your situation.

Now that we defined key terms and have determined our goals, it is time to discuss the two common types of malware analysis that are routinely performed.

6. Types of Malware Analysis

There are two types of malware analysis that security professionals perform: code (static) analysis or behavioral (dynamic) analysis. Although both types accomplish the same

goal of explaining how malware works, the tools, time and skills required to perform the analysis are very different. Code analysis is the actual viewing of code and walking trough it to get a better understanding of the malware and what it is doing (Malware Analysis, The Basics, 2007). Behavioral analysis is how the malware behaves when executed, who it talks to, what gets installed, and how it runs (Malware Analysis, The Basics, 2007). When performing Malware analysis, both static and dynamic analysis should be performed to gain a complete understanding on how that particular malware functions.

Knowing how malware functions allows for better defenses to protect the organization from this piece of malware, and possibly malware that attempt to infect a host using the same vulnerabilities are weaknesses.

With any discussion of reverse code engineering the reader *must* be aware of the laws in their country involved with reverse engineering software. Before undertaking reversing, check the local (country) laws about reverse code engineering. I do not encourage nor do I condone anyone to break laws.

Code analysis is performed by looking at the software code of the malware to gain a better understanding on how the malware functions. While performing code analysis, anti-virus software will run on the malware, string searches will be performed, and files such as shell scripts will be analyzed. Most likely, reverse engineering will have to be performed using

programs such as disassemblers, debuggers and decompilers.

After successfully reversing malware, the reader will be able to see how the “source” code of the malware functions. Seeing how the code functions allows the reader to build better defenses to protect their organization as well as serve as a sanity check on the completed behavioral analysis.

Once the malware code has been reversed, an understanding on how the malware infects the system will become clear. With malware today becoming more targeted, understanding how malware infects systems can reduce infections to an organization, thus reducing the overall cost.

Behavioral analysis is the “quick and dirty” way of malware analysis. When performing a behavioral analysis, look at how the malware behaves and what changes the malware makes on a base lined system. It should be noted, when performing behavioral analysis it is critical the malware lab in not connected to another network. For the best protection of production networks, the malware lab should never be connected to any network. If files must be transferred use a read only media such as CD-ROM.

When performing behavioral analysis, look for changes to the system as well as any unusual behavior on an infected system. Changes on the system that should raise a red flag include files that have been added and/or modified, new services that have been installed,

new processes that are running, any registry modifications noting which modifications took place, and finally, if any systems settings have been modified. This would include DNS server settings of the workstation which have been changed. Beside the behavior of the system itself, network traffic will also be examined.

Now that an understanding of what behavior the malware does to systems and networks, the reader may have the desire to understand how the malware actually performs these activities. The answers to that question require the reader to perform an analysis of the malware.

7. Tools of Malware Analysis

After covering the basics of malware analysis such as identifying key terms, goals of analysis, and types of analysis, it is critical to identify various tools that can be use to perform malware analysis. This is not a comprehensive list of tools that one must use, only the ones that the author has used.

Constructing a lab for malware analysis requires the reader to contemplate the pros and cons of using physical hardware or virtual machines (VM). The author recommend's leaning towards the use of virtual machines using the VMware Server product. VMware Server is a free download from the www.vmware.com website. VMware reduces the cost of

hardware to needing only one or two physical machines. VMware allows many types of OS, including Windows and Linux, to be installed. Once of the best features of VMware is the snapshot. Before performing any type of analysis, taking a snapshot will save lots of time down the road. Another nice feature is the host only networking, which means the lab will only see itself. Also one should utilize the ability to disable VMware's access to the network interface card. Remember, when using VMware a large amount of RAM is needed. For Windows based and Linux systems that need a GUI, a minimum 512 MB of RAM should be used. For text based Linux boxes a minimum 256 MB of RAM should be used.

Although virtualization of the malware lab is great for cost reduction, there are issues with using virtualization software. Some of the more sophisticated malware today will attempt to detect a VM. If the malware detects it is being run on a VM, it will not execute.

After building the virtual machines, the operating systems installed in the malware lab will depend on the malware being analyzed and the operating systems used in the organization. The author normally will have a Windows XP Professional machine and a Linux machine loaded. Depending on the malware being analyzed, load a Windows Server (either 2000 or 2003) with all appropriate applications, such as IIS. Use a Windows XP machine as the malware victim, and either the Linux or Windows server to host such services such as WWW, FTP, DNS, and SMTP. No matter what operating system is used, make sure that

installed services or listeners are running appropriately to act as the “compromised” server.

After installing the operating system, utilize VMware’s snapshot feature and take snapshots of the VM’s. Once the base OS snapshot is finished, install service packs, patches, and hot fixes deployed in the organization. Upon completion of the OS and all patches, load the tools needed for analysis. After loading the tools, record MD5 hashes of all tools used to ensure that the malware does not install a root kit. After obtaining the MD5 hashes, take one final snapshot before beginning the analysis.

The tools to be installed will depend on the type of analysis is to be performed. Although there is no hard and fast rule about which type to perform, the authors experience leads to performing the static analysis first, followed by the dynamic analysis. The information gathered during the static analysis will usually provide information needed for the dynamic analysis.

After selecting the tools, building the lab, and taking final snapshots, burn all of the tools onto a CD before beginning the analysis. The CD tools will be used to verify the tools installed on the system are reporting correctly. After completing the malware analysis, run an MD5 check again on the tools on the system that were used for the initial analysis. By comparing MD5 hashes, one can be assured that the malware did not modify the tools that were used to perform the analysis.

The tools that will be used for behavioral analysis are listed below along with a brief description of what the tool does and a the website were you can download the tool

BgInfo - small application providing import system information such as hostname, IP address, OS version, etc.

http://www.microsoft.com/technet/sysinternals/Utilities/BgInfo.mspx

Process Explorer – small application that find out what files, registry keys and other objects have open, which DLL's they have loaded.

http://www.microsoft.com/technet/sysinternals/Utilities/ProcessExplorer.mspx

Process Monitor – small application used to monitor file system, registry, process, thread and DLL activity in real-time.

http://www.microsoft.com/technet/sysinternals/Utilities/processmonitor.mspx

PSfile - application that shows a list of files on a system that are opened remotely.

http://www.microsoft.com/technet/sysinternals/Utilities/PsFile.mspx

RootkitRevealer – application that scans system for known rootkit-based malware.

http://www.microsoft.com/technet/sysinternals/Utilities/RootkitRevealer.mspx

Streams – application that reveals NTFS alternate streams.

http://www.microsoft.com/technet/sysinternals/Utilities/Streams.mspx

Strings – application that searches for ANSI and UNICODE strings in binary images.

http://www.microsoft.com/technet/sysinternals/Utilities/Strings.mspx

TCPView – application providing information about TCP and UDP connections, including the local and remote address and TCP connection state.

http://www.microsoft.com/technet/sysinternals/Utilities/TcpView.mspx

Windump – Windows version of the powerful and flexible tcpdump sniffer.

http://www.winpcap.org/windump

nmap – world's best port scanner.

http://www.insecure.org/nmap/download.html

Fport – Identifies unknown ports and their associate applications.

http://www.foundstone.com/us/resources-free-tools.asp

Hfind (Part of the Forensic Toolkit) – application that will scan for the disk for hidden files.

http://www.foundstone.com/us/resources-free-tools.asp

Vision – reports all open TCP and UDP ports and maps them to the owning process or application.

http://www.foundstone.com/us/resources-free-tools.asp

Filewatch – a file change monitor.

http://www.foundstone.com/us/resources-free-tools.asp

Attacker – a TCP/UDP port listener.

http://www.foundstone.com/us/resources-free-tools.asp

MD5sums – Generates MD5 hashes for file integrity verification.

http://www.pc-tools.net/win32

Winalysis – monitors for changes to files, the registry, users, groups, security policies, services, shares, scheduled jobs, the system environment and more.

http://www.winalysis.com/download.htm

WinHex – Hex editor, you may choose any hex editor that you like.

http://www.x-ways.net/winhex/

Some tools used for behavioral analysis will be also be used for code analysis. The

tools used for code analysis are listed below, along with a brief explanation.

IDA Pro – popular interactive, programmable, extendible, multi-processor debugger and disassembler.

http://www.datarescue.com/idabase/idadown.htm

Reverse Engineering Compiler – popular decompiler.

www.backerstreet.com/rec/rec.htm

ProcDump 32 – unpacker application.

http://www.fortunecity.com/millenium/firemansam/962/html/procdump.html

PE Explorer -- provides tools for disassembly and inspection of unknown binaries.

http://www.heaventools.com/download-pe-explorer.htm

Windbg – windows debugging applications.

http://www.microsoft.com/whdc/devtools/debugging/installx86.mspx

Livekd – application that allows Windbg debugger to run locally on a live system.

http://www.microsoft.com/technet/sysinternals/SystemInformation/LiveKd.mspx

Debugview – an application that monitors debug output on your local or a remote system.

http://www.microsoft.com/technet/sysinternals/utilities/debugview.mspx

Now that we have identified a list of critical tools that can be used to perform a malware analysis, we will briefly discuss an area of malware that could and should warrant its own paper. That is some common components of malware.

8. Components of Malware Analysis

When malware authors create malware, different components are used to create malware that meets the malware author's goal. The components written into the code varies depending on the purpose and goal of the malware as well as the experience and skill of the author. However, there is one very common component used today; a run-time packer. By 2004 90% of the 32-bit computer viruses use a run-time packer such as UPX or ASPACK (Szor, 2005).

Run-time packers, also known as packers, are applications that perform compression of code, much like other common compression applications such as WinZip or RAR. However, unlike WinZip or RAR, when an application is decompressed with a packer, the application will be decompressed into the system memory (Szor, 2005) rather through the

files being decompressed onto a file system. By unpacking the malware into memory, the detection and removal of code may be more complicated, requiring a more advanced skill set most end users do not posses.

There are many advantages to using packers with malware code which allow the malware code an increased chance of avoiding detection during infection, installation, execution, and propagation of the code itself. As stated earlier, the majority of malware today uses packer applications which warrant reasons for using packer applications.

The first reason for compressing the malware is to make the malware code smaller and less detectable. Many organizations today have bandwidth limits on how much traffic that a user can use, especially with email. Also, smaller code will be harder to detect since most networks traffic consist of numerous small sessions instead of one big session.

Another important reason for packing malware is the fact that if the code is packed anti-virus (AV) software may not be able to detect the malware code. In the book The Art of Computer Virus Research and Defense, the author states that there are over 500 various packer applications in the wild (Szor, 2005). In short, AV software must be able to detect over 500 different types of packers simply to look at the contents of whatever is in the packer. Although AV software may be able to detect that the file is compressed, it may not have the ability to unpack the file and perform a scan on the files contents.

Another anti-detection method that may be utilized would be to encrypt the malware code in the packer so even if the AV software is able to unpack the code, it will only scan an encrypted version. This allows the malware author who uses strong encryption a solid way to ensure that their malware will be able to bypass AV software.

Although packers are being discussed, it is important to remember malware will contain any type of component the author feels is needed for their code to be successful. This means the malware author can use applications, scripts, and/or remote networks to further the success of their malware.

Some malware code may contain different types of applications or code to install various types of application. These applications could be backdoor applications such as Netcat, VNC or configure Microsoft Remote Desktop Connection to desired configuration. Two great examples of installing or shoveling Netcat and VNC as backdoor applications can be found in Malware Fighting Malicious Code (Skoudis and Zeltser, 2003). Perhaps the malware will be a spam-bot and install its own SMTP engine or scan the network for one to use. A third piece of malware might contain its own web server to host a phishing scam.

Malware may contain scripts which change the system settings. For example, the malware will disable the windows firewall, change the DNS settings or possibly patch the system with the latest updates. You might ask “Why would an attacker write script to update

a vulnerable system?" The reason attackers are patching systems has nothing to do with being good "Net neighbors", but rather preventing another attacker from gaining access to a system that they "own".

Obviously, malware can contain a variety of different components, and any component will be used to ensure the malware will function as designed. When performing malware analysis, nothing should be taking for granted or trusted. Malware authors have repeatedly shown in the past that any thing will be used to ensure that their malware will succeed.

9. Malware Acquisition

In order to perform malware analysis, malware must be available to analyze. There are many methods to malware acquisition but three will be discussed in this section. The "old-school" method of a user "obtaining" the malware will be the first method described. Specialized honeynets that are used to capture malware will be discussed next. Finally, using a search engine as a method to acquire malware will be discussed.

End users from time to time will acquire malware for analysis. When this happens the organizations Incident Response Plan should be followed. There are countless ways end users acquire malware. Two of the most common ways users acquire malware is unknowingly visiting malicious web servers and email attachments.

One common way end users become infected with malware is when the user unknowingly visits a compromised server. According to the Know Your Enemy Whitepaper, when a user visits a compromised server a client-side attack will take place. Client-side attacks are attacks that target vulnerabilities in client applications that interact with a malicious server or process malicious data. Here, the client initiates the connection that could result in an attack. If a client does not interact with a server, it is not at risk, because it doesn't process any potentially harmful data sent from the server (Honeynet, 2007).

Most organizations today block most types of attachments. However, many types of attachments are still permitted through. Using social engineering, attackers can overcome the blocking of certain file types.

When end users become infected with malware the organizations' Incident Response Plan must become active. Unlike other methods of malware acquisition discussed later in this section, when an end user host becomes infected, this is an incident.

Low interaction honeynets such as Nepenthes (http://nepenthes.mwcollect.org/) is another method to gather malware. This type of malware acquisition is used for research purposes. When deploying honeynets, captured malware is not considered an incident. If captured malware is inadvertently release on the organizations production network, the Incident Handling Plan must be followed.

The third method is to use a search engine to search for malware binaries. HD Moore, author of Metasploit, released code to use the Google search engine to find live active malware. Details and use of this method to collection malware are described in the referenced article (eWeek, 2007).

It is important to remember when end users hosts that become infected with malware, the organizations Incident Response plan must be followed. Now that there is malware to analyze, a discussion of malware analysis methodology will follow.

10. Methodology of Malware Analysis

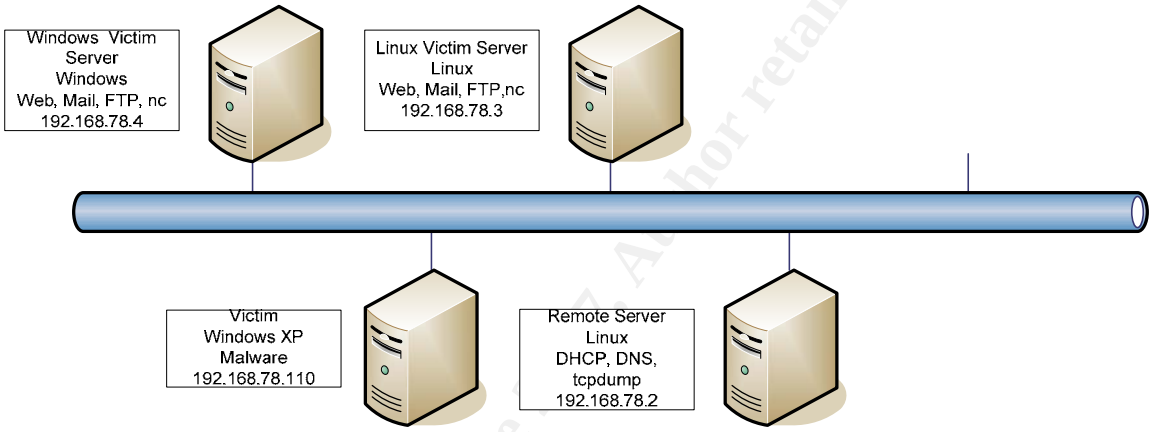
Up to this point, definitions, terms, incident handling, goals, types, tools, components of malware analysis, and malware acquisition which are critical to a true understanding of malware analysis have been discussed. Next, malware analysis methodology will be discussed.

When performing an analysis, do not be attached to the network except for the malware lab network. Accidentally launching a malware outbreak on the corporate network will not please your organizations management.

When performing malware analysis, the first step would be to have malware. For the purpose of this paper, Matt Jonkman at www.bleedingthreats.net provided access to their

sandnet so that I could analyze one of the pieces of malware previously obtained.

After obtaining malware, build the malware lab where the analysis will be performed. This malware lab will consist of four virtual systems using VMware server as the virtualization software. First, create the virtual machines (VM). Next, install the required operating systems in their respective VM. This lab will contain one Windows XP VM, one Windows 2003 Standard Server VM and two Linux VM. Below is a diagram of how the Malware lab will look.



After building the lab, copy the required tools to the various machines. Once the tools are copied, extract and if required, install the tools. Upon completion of installation of tools, take MD5 hashes of all tools that will be used in the malware.

Next, a base line of the system will be taken before running the malware. Multiple tools will be used for the base line. After executing the malware, the same tools will be run again to be used to compare the new infected system against the base line. Finally, take a VM snap

shot and ensure that *Host-Only* networking is selected.

The first step of malware analysis is to run the organizations Anti-Virus (AV) software against the malware. It is also wise to run AV software from multiple vendors. The author runs AV software from three different vendors. The three vendors are the AV application the author manages for his job, the AV application that the author runs on his systems at home, and a third AV application that the author changes from time to time. Test to see if the AV software detects the malware, making notes which AV applications detect, and what the applications detect the malware as.

After scanning the malware with AV software, open the Malware up in a hex editor to see what type of file the malware may be. While examining the file with the hex editor, attempt to determine if the malware is using a packer application such as UPX.

Some packers, such as UPX, will allow for the decompression of the file. If the malware is packed with one these packers, attempt to unpack the application. By unpacking or decompressing the malware, other tools can be run against the malware. Before, attempting to decompress the malware, make a copy of the malware. The malware may not function if the repacking of the malware is done incorrectly.

One of the most useful tools is Strings. Strings, is an application that searches for ASCII, Unicode, or both types in a file. The strings search can provide insightful information

such as protocols, ports, files, IP addresses, and even information about the malware's. By looking over the information generated by running strings more insight into the inner workings of malware might be gained.

After completing the strings search, it is time to disassemble the malware. The results from disassembly will vary from malware to malware. For the purpose of this paper, reviewing the disassembled code will focus on the calls the malware code will make to various DLL's, as well as system changes the malware will make.

Disassembled and reverse engineered malware can provide a great deal of information on how malware functions. Although these advanced malware analysis techniques are outside the scope of this paper, it should be noted that they are, in fact, performed.

Having completed the static piece malware analysis, it is time to move on to dynamic malware analysis. The dynamic analysis is where the malware is executed and the system is observed for any changes that may happen.

Before beginning dynamic analysis, there are a few steps that should be taken to protect your systems. Again, make sure all service packs, patches, hot fixes and any applications needed are installed on the VM before executing the malware. Next, make sure that VM networking is set to *Host-Only* networking. If the VM network settings are bridged or

Network Address Translation there is a *great* risk of introducing the malware to any connect networks.

Upon completion of the code analysis, perform one last baseline by taking a system snap shot using Winanalysis. Usually the author has the following four tools running on a windows machine when executing malware:

Process Explorer

TCPView

Windump

Explorer

Process Explorer shows any processes that are started during the execution of the malware. TCPView shows attempted or completed network connections the malware makes. Windump is used to record all network traffic. Explorer is used to browse to the location of the malware so that it may be executed.

Execute the malware while watching Process Explorer and TCPView for changes in the system status. Be sure to note changes that occur after executing the malware. Allow the malware some time to run. Fifteen minutes is usually sufficient.

Using Winanalysis, take another system snap shot. Upon completion, compare this

snap shot to the original snap shot. Run Process Explorer and TCPView again, looking for changes from the baseline taking earlier. Make notes of changes that were made. Finally, examine the network traffic captured with windump.

While reviewing Process Explorer, observe new process that may be running and note where they are located. These new process may be loaded into the registry which could allow the malware to be loaded every time at boot. Any new process installed by malware must be thoroughly investigated.

Use TCPView to look for new listeners, which might receive instructions from Command and Control servers, installed on the system by the malware. If a new listener is installed on the system, investigate the listener. Attempt to connect to it with various tools such as telnet, netcat, or a web browser. Trace the listener back to the process that spawned the listener and investigate thoroughly.

While examining network traffic, observe how the malware functions. Make notes on what the traffic looks like. This will be used to assist with writing access list and IDS rules. For example, if the malware installs a backdoor, set up a server so that it can download a backdoor. By allowing it to download a back door, you can now connect to the system and see what the malware author sees.

Although there is a methodology used when performing malware analysis, the reader

must be reminded that malware analysis is not an exact science. Changes to this methodology will occur because malware is constantly evolving. Malware analysis is both an art and a science, the reader must be flexible when performing malware analysis. It is now time to perform malware analysis on a live piece of malware using some of the tools outlined in this paper and the methodology as outlined above.

11. Malware Analysis

For this paper, the malware to be analyzed was provided by Matt Jonkman at bleedingthreats.net. This piece of malware was captured using their sandnet. The malware will be examining is called one-time.exe.

Now that the malware is acquired, create the malware analysis lab. For the purpose of this discussion, a virtual lab using VMware Server will be created. Begin by installing VMware which may be downloaded at no cost from www.vmware.com.

Next create the “computers” in malware virtual lab. This malware lab will consist of four systems, a victim workstation, a Linux server, a windows server and a remote “management” server. The victim machine will be the machine where the malware will be ran, and the remote server will be used to provide services such as DNS and DHCP. The Windows and Linux server will have netcat installed on to act as listeners.

Build the victim machine with Windows XP installed. Start the VM build by clicking on the “New Virtual Machine” icon, and then click “Next” on the welcome screen. Select “Typical” for the virtual machine configuration and click “Next”. For the guest Operating System, choose Microsoft Windows, and in the drop down select Windows XP. When asked for the virtual machine name type in the name VICTIM. Make sure when asked for network type you select *Host-Only* networking. Select the disk size that meets our needs, 2.0 GB and click “Finish”.

Build the Windows Victim server with Windows 2003 Server installed. Start the VM build by clicking on the “New Virtual Machine” icon, and then click “Next” on the welcome screen. Select “Typical” for the virtual machine configuration and click “Next”. For the guest Operating System, choose Microsoft Windows, and in the drop down select Windows 2003. When asked for the virtual machine name type in the name WINDOWS VICTIM SERVER. Make sure when asked for network type you select *Host-Only* networking. Select the disk size that meets our needs, 4.0 GB and click “Finish”.

Build the remote server using a Linux operating system. In this case, Mandriva will be the Linux flavor used in the malware lab. Start the VM build by clicking on the” New Virtual Machine” icon, and then click “Next” on the welcome screen. Select “Typical” for the virtual machine configuration and click “Next”. For the guest Operating System, choose Linux, and

in the drop down select Mandriva Linux. When asked for the virtual machine name type in the name REMOTE SERVER. Make sure for network type you select *Host-Only* networking. Select the disk size that meets our needs, 2.0 GB and click “Finish”.

The last step will be to build the Linux Victim server. Start the VM build by clicking on the “New Virtual Machine” icon, and then click “Next” on the welcome screen. Select “Typical” for the virtual machine configuration and click “Next”. For the guest Operating System, choose Linux, and in the drop down select Mandriva Linux. When asked for the virtual machine name type in the name LINUX VICTIM SERVER. Make sure for network type you select *Host-Only* networking. Select the disk size that meets our needs, 2.0 GB and click “Finish”.

Now the lab machines are built, install the OS onto each machine. On the remote server be sure to include the following applications as well:

Tcpdump

Bind

Apache

Dhcp

Iptables

SSH

Having completed installation of the operating systems on the virtual machines, install the tools required for analysis. On the Victim machine, create a folder called c:\tools. This folder will store all tools that will be used during our analysis. In the tools directory, copy the following tools into the directory:

MD5sums

PE Explorer

Process Explorer

Strings

Tcpview

UPX

Windump

Winhex

Winanalysis

WinPcap

It may be necessary to install some of the tools for the malware analysis. Complete the

installation and extraction of the above listed tools into the c:\tools directory.

Now that all of the tools needed are installed, take MD5 sum hashes of the tools by using the MD5sums.exe application. Below are some commands used to generated hashes and record the hash on the F drive, which is a USB key. Here is the syntax:

```
C:\tools\md5sums.exe file >> f:\ md5sums.txt
```

Using the file attacker.exe as an example, the syntax is:

```
C:\tools\md5sums.exe attacker.exe >> f:\ md5sums.txt
```

To perform hashes with multiple files with the same file extension, use this syntax:

```
C:\tools\md5sums.exe *.exe c:\tools\vision\*.exe c:\tools\winhex\*.exe
c:\tools\upx\*.exe >> f:\md5sums.txt
```

Once the tools MD5 sums are taken, it is time to begin to base line the system for comparison after the system becomes infected with the malware. Multiple tools will be run with the output being saved a USB drive. The methodology used is as follows:

A.) Reboot the system to ensure that no programs are running that should not be.
B.) Open up process explorer and save to f:\procexpbaseline.txt, close process explorer
C.) Open up tcpview and save to f:\tcpviewbaseline.txt

D.) Open up winalysis and take a snapshot, be sure to select all options when taking the snap shot. Save the snapshot as f:\winalysisbaseline.txt and say yes to include change details in the print details.

E.) Ensure that HOST-ONLY networking is selected in the VM

F.) Reboot

G.) Take a VM snapshot

After completing the base line system on the Victim desktop, complete the exact system base line on the OS victim server next. This ensures if the Victim server becomes compromised by the malware, an analysis can be performed as needed.

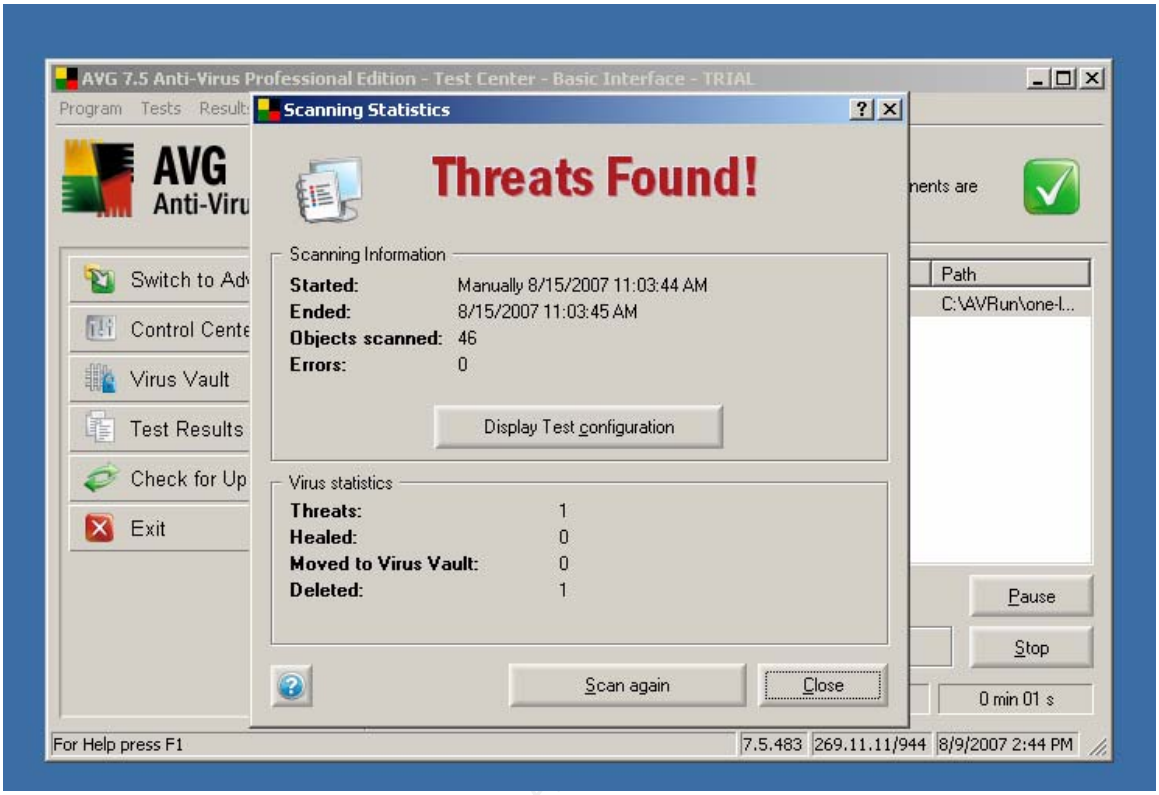
Now that all tools are installed and the system baseline is complete, begin the actual malware analysis. The first step in the analysis is to run AV software against the malware. The three vendors used for this piece of malware:

AVG

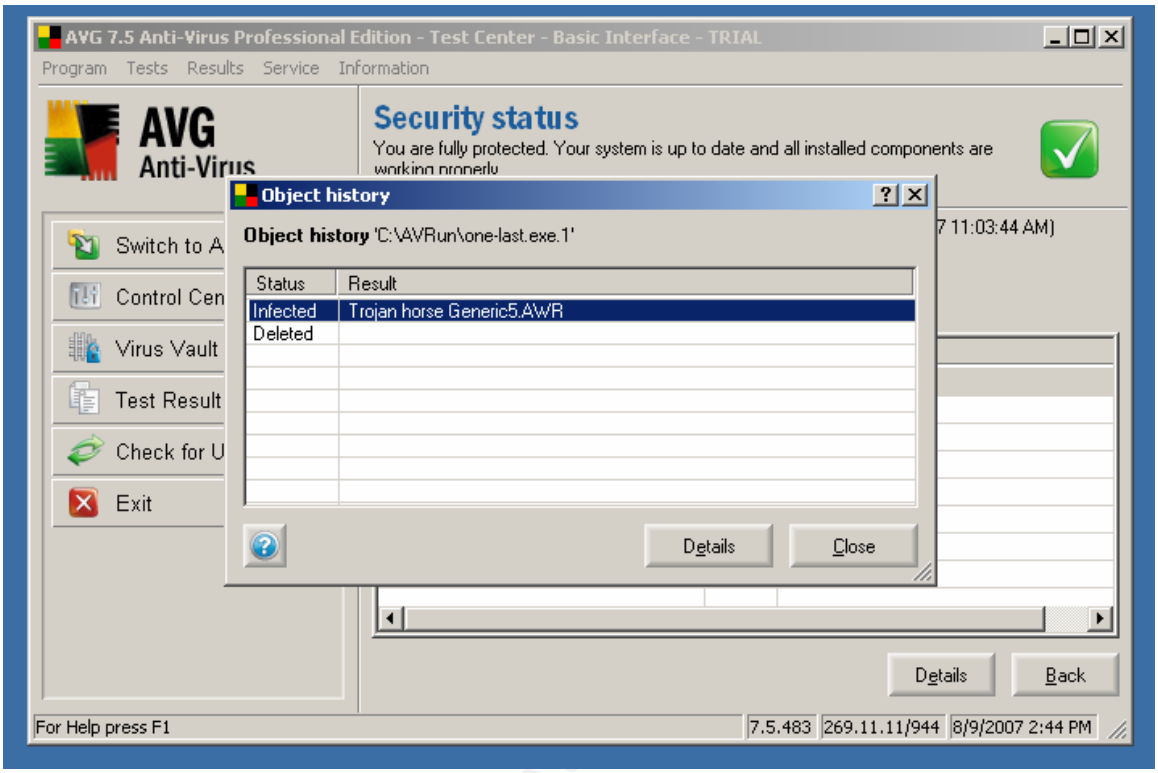
Kaspersky

McAfee

Selecting these three AV products is not an endorsement of these vendors or a negative endorsement of other AV vendors. The first AV software used to scan for the malware is AVG. Here is a screen shot of the results:



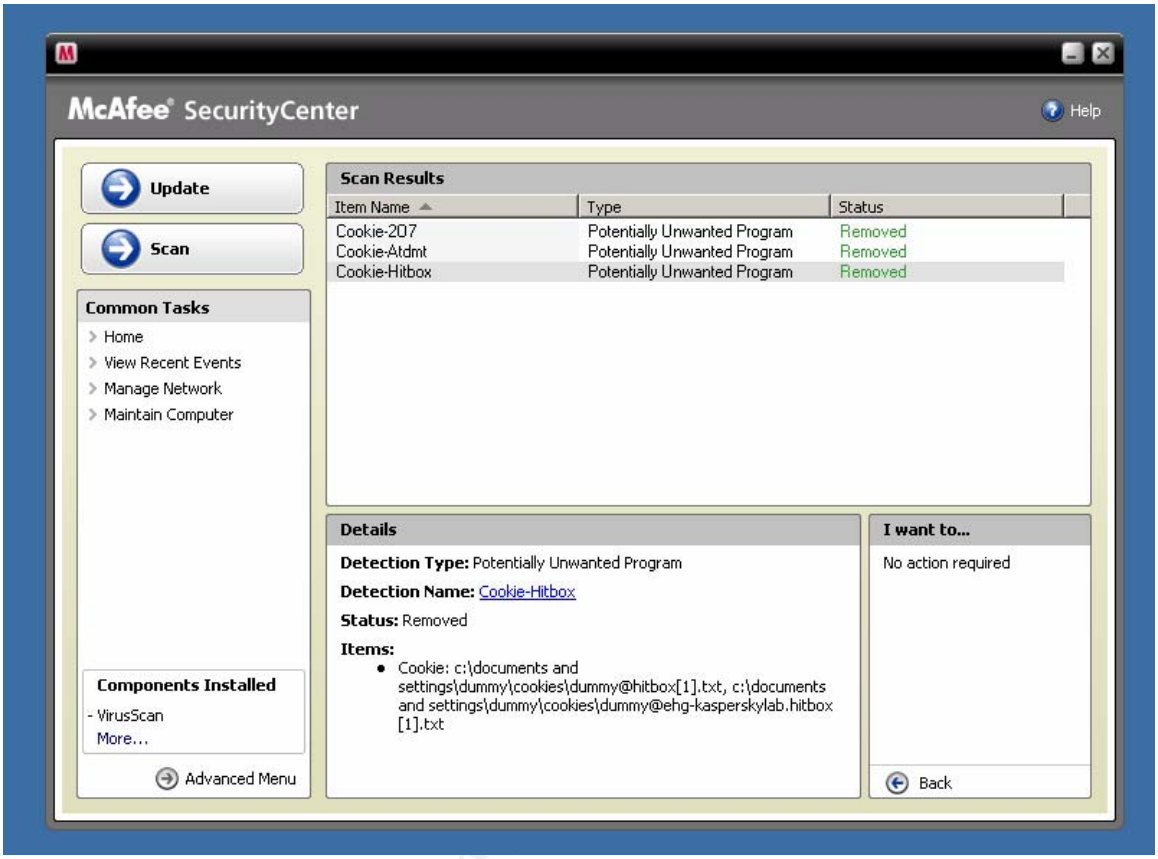
By reviewing the information from AVG, the object history informs us that the file is a Trojan horse downloader. Here is the screen shot that provides the information:



Next the system is scanned with the Kaspersky AV product. The results of the scan are seen in the screen shot below:

By reviewing the information from Kaspersky, the results are the file is a Trojan.Win32.Agent.ago downloader.

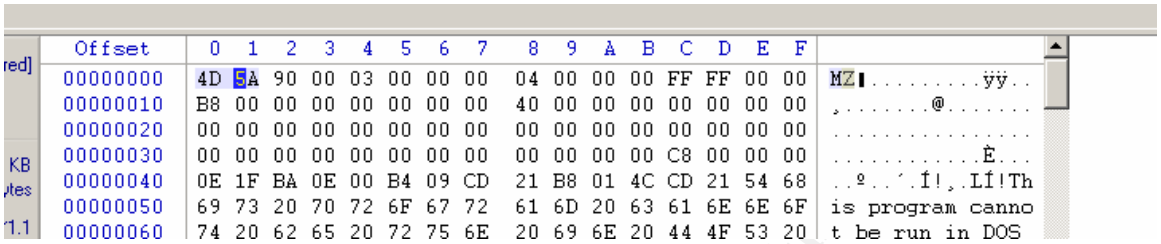
The final step is to scan the system with the McAfee AV product. The results of the scan are seen in the screen shot below:



After reviewing the results of all three vendors, note that the McAfee product did not detect the malware. This is good example why running multiple AV products when performing malware analysis is done. It should be noted that because McAfee did not detect this particular piece of malware, does not make McAfee a poor choice of AV product. There is other malware that McAfee will detect that Kaspersky and AVG will not. This is a great example of using defense in depth with AV software.

Next, review the malware with a hex editor. The first thing to note are the first two bytes from offset 0000 that have the hex of 0x4D 0x5A which translates to MZ. MZ is a

Portable Executable (PE) type file. PE files can run win32 applications which call functions in the Win32 API set (Szor, 2005). An example is shown in the screen shot below:



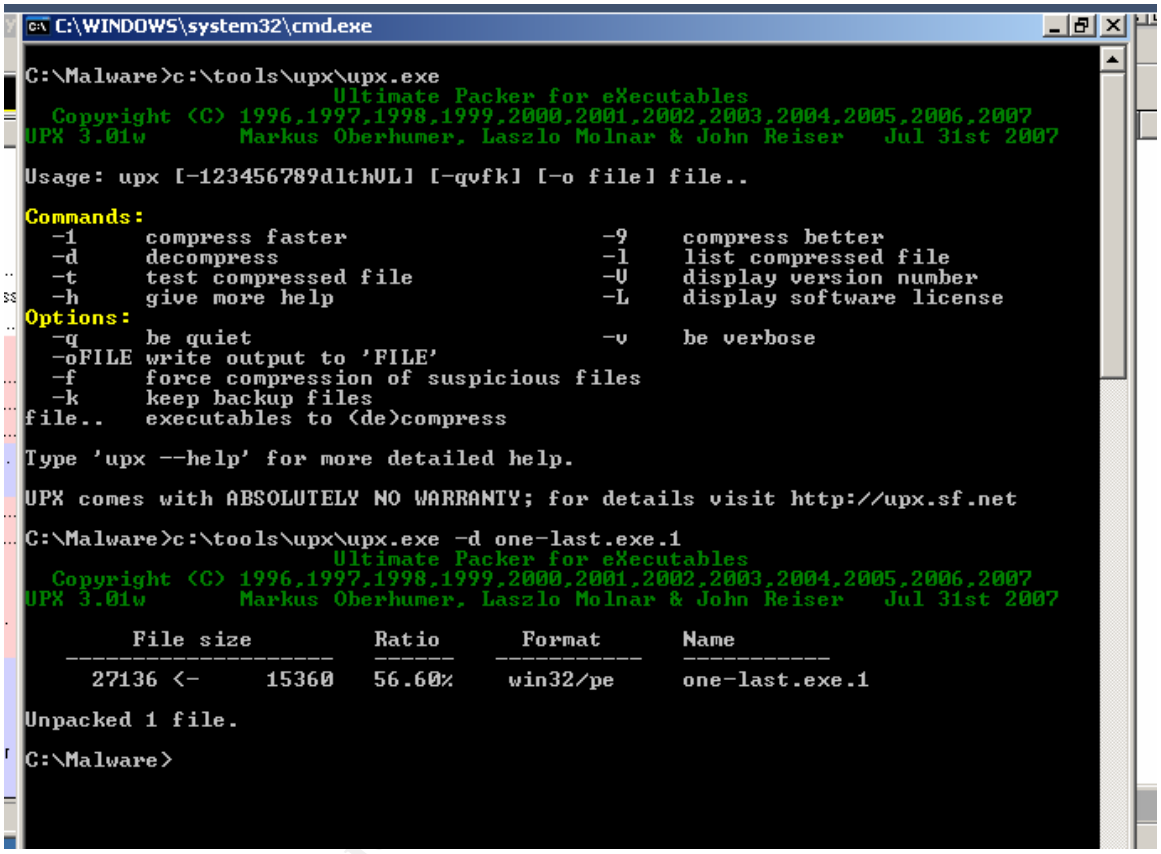
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ▮.....ÿÿ..
00000010	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	¸.......@.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000030	00	00	00	00	00	00	00	00	00	00	00	00	C8	00	00	00	.....È...
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	..º..´.Í!¸.LÍ!Th
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS

The next item of interest is looking at offset 01C0 we see the letters UPX. UPX is one of the most common packers used by malware authors today. All though none of the AV products earlier detected the packer, sometimes AV software will detect the packer. An example is shown in the screen shot below:

000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001C0	55	50	58	30	00	00	00	00	00	60	00	00	00	10	00	00	UPX0.....`.....
000001D0	00	00	00	00	00	04	00	00	00	00	00	00	00	00	00	00	.....
000001E0	00	00	00	00	80	00	00	E0	55	50	58	31	00	00	00	00	....▮..àUPX1....
000001F0	00	40	00	00	00	70	00	00	00	32	00	00	00	04	00	00	.@...p...2.....
00000200	00	00	00	00	00	00	00	00	00	00	00	00	40	00	00	E0	.....
00000210	2E	72	73	72	63	00	00	00	00	10	00	00	00	B0	00	00	.rsrc.....°..
00000220	00	06	00	00	00	36	00	00	00	00	00	00	00	00	00	00	.....6.....
00000230	00	00	00	00	40	00	00	C0	00	00	00	00	00	00	00	00	....@..À.....
00000240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

By examining the malware with a hex editor it is clear that the malware is of the Portable Executable format that is packed with UPX. By typing out the upx.exe command, a list of options to use with the UPX packer is provided. In order to decompress the application, use the following syntax to decompress the malware using the upx.exe –d command as shown in the screen shot below. Once the unpacking process is complete a message as

shown below in this screen shot will provide some details about the compression ratio and the file format that was decompressed:



Now that the file is decompressed, look through the executable file for characters that might provide more information about the malware. Use the application called Strings.exe that will perform the search. Strings, by default, will search the file for both Unicode and ASCII characters. Type in the following command:

strings.exe c:\malware\one-last.exe.1 > strings.txt.

The output from the strings command is sent to a text file to allow us to go back to the file as needed during our analysis. Search through the strings file and look for interesting information. Notice registry keys are defined as well as key values. Next, notice the comments on default network monitoring tool as seen in this screen shot:

```
Regmon Helper
SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\{75F5DD2A-58E3-48a6-AB94-53632157E17E}
SOFTWARE\Microsoft\Windows\CurrentVersion\Run
SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows
Network Core ID
Watcher Name
AppInit_DLLs
Run Value
Installed Subsystem ID
{38B63F16-A14D-4ba5-99BF-637FDC6346D7}
Global\{97A8068C-2262-4df7-8CBB-8898494C89F0}
Global\{7CE5AE4D-F896-489e-98A0-151D5CBDC8A8}
{D211556C-8977-4eb7-9BD3-79C5C70EF598}
{79D30EC4-E2CD-40d8-B277-80803A88A1EF}
cryptdll.dll
BIN
jjj
{38B63F16-A14D-4ba5-99BF-637FDC6346D7}
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ
cryptdll.dll
VS_VERSION_INFO
StringFileInfo
000004b0
Comments
Default network monitoring tool
CompanyName
Microsoft Corporation
FileVersion
3, 1, 0, 1
LegalCopyright
 Microsoft Corporation. All rights reserved.
ProductName
Default network monitoring tool
ProductVersion
2, 1, 0, 0
VarFileInfo
```

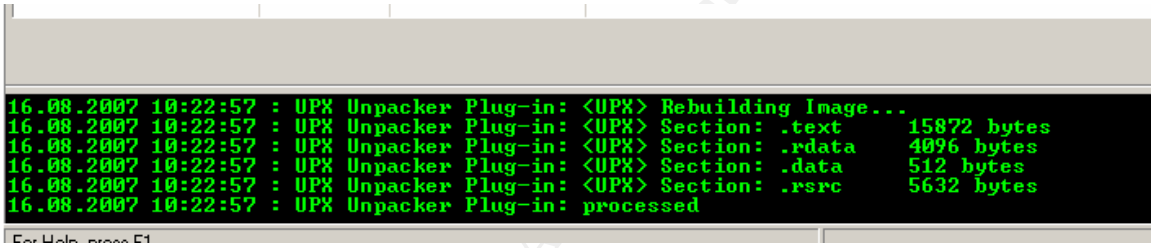
Further down in the strings output is a domain name as see in this screen shot:

```
%l`@
 P@
%p`@
%4Q@
www.jorbanblack.com
(1@
,1@
f1@
k1@
```

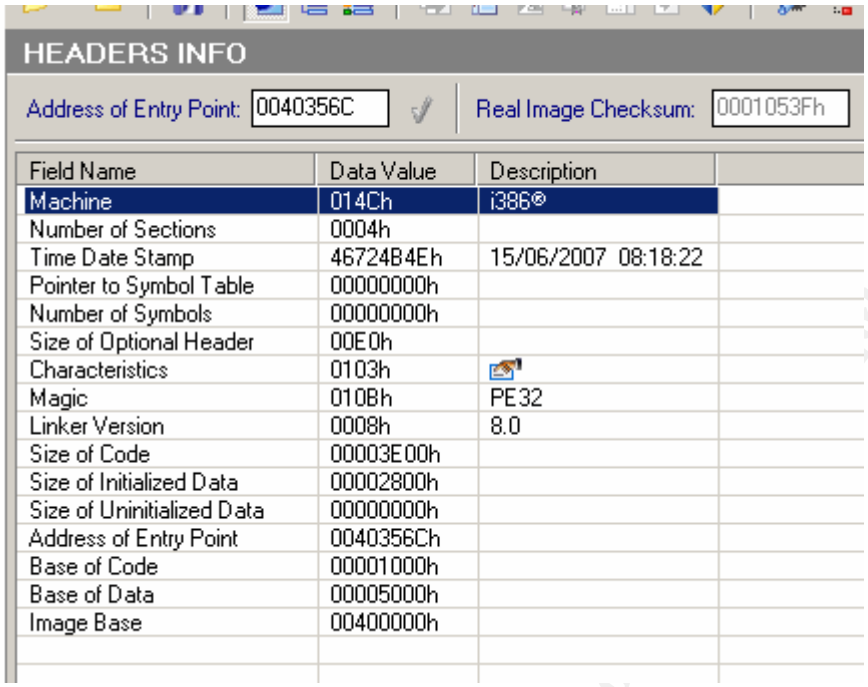
With the web site name of www.jorbanblack.com use a WHOIS database to conduct a search to see what information can be found about the web site. Always consider the idea of

contacting the web site administrator, if with 100% certainty, the reader *knows* the web site is hosting malware.

Now that the strings have been analyzed, the malware will be disassembled using PE Explorer. Remember to have an understanding for the laws that pertain to performing reverse engineering. Opening up the malware using PE explorer, note that PE explorer unpacks the malware as seen in this screen shot:



Other interesting information is provided by the header information as seen in this screen shot:



HEADERS INFO

Address of Entry Point: 0040356C | Real Image Checksum: 0001053Fh

Field Name	Data Value	Description
Machine	014Ch	i386®
Number of Sections	0004h	
Time Date Stamp	46724B4Eh	15/06/2007 08:18:22
Pointer to Symbol Table	00000000h	
Number of Symbols	00000000h	
Size of Optional Header	00E0h	
Characteristics	0103h	
Magic	010Bh	PE32
Linker Version	0008h	8.0
Size of Code	00003E00h	
Size of Initialized Data	00002800h	
Size of Uninitialized Data	00000000h	
Address of Entry Point	0040356Ch	
Base of Code	00001000h	
Base of Data	00005000h	
Image Base	00400000h	

Using PE Explorer, the malware will be disassembled. Again for this paper take notice of system changes and DLL calls in the disassembled code. Going through the code the first DLL call of interest is the KERNEL32.DLL!Delete file as seen in this screen shot:

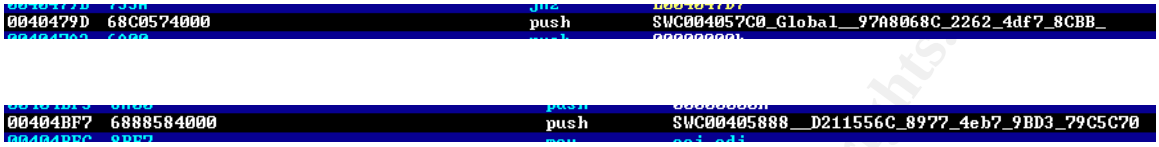
```
00403664  FF742414                        push    [esp+14h]
00403668  FF1578504000                    call    [KERNEL32.DLL!DeleteFileW]
0040366E                          L0040366E:
```

It is now known the malware will delete the kernel32.dll. Continue looking for more interesting DLL calls. The next DLL call of interest is the KERNEL32.DLL!CreateProcess call as seen in this screen shot:

```
00403772  50                              push    eax
00403773  FF1568504000                    call    [KERNEL32.DLL!CreateProcessW]
```

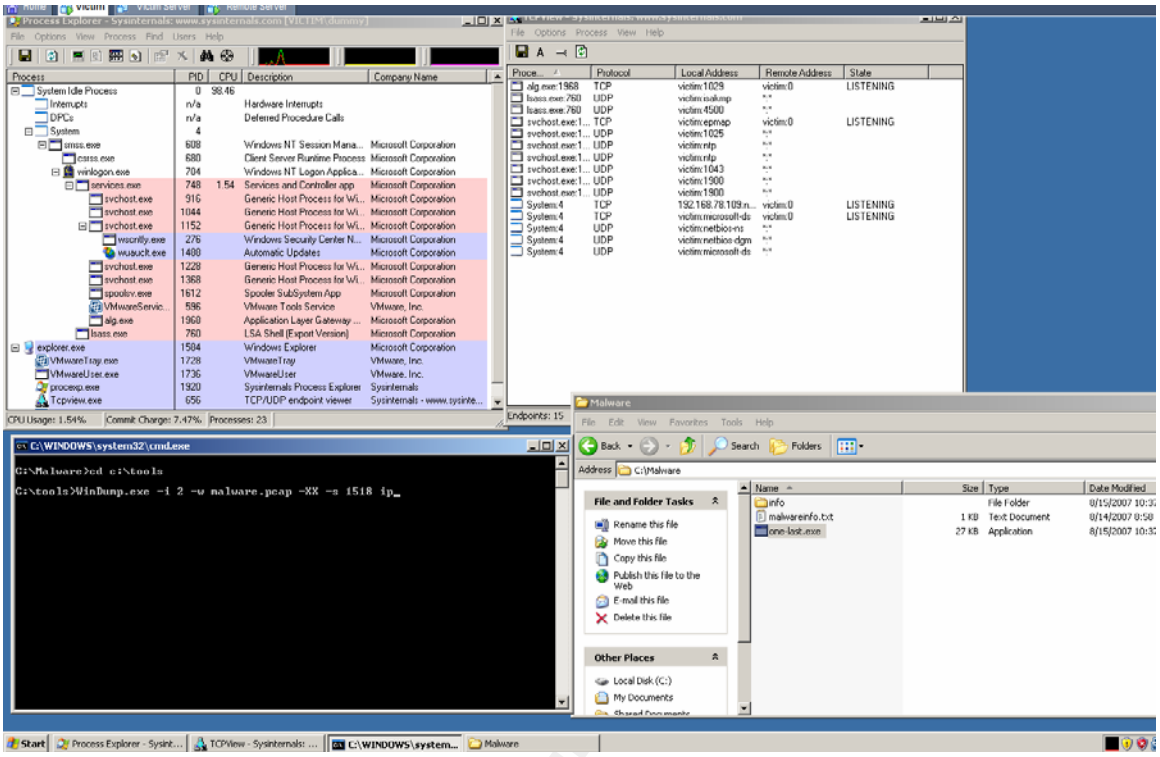
There is going to be a new process created by the malware. Continue through the

disassembled code looking for more interesting information, notice that two registry keys are being pushed as seen in this screen shot:



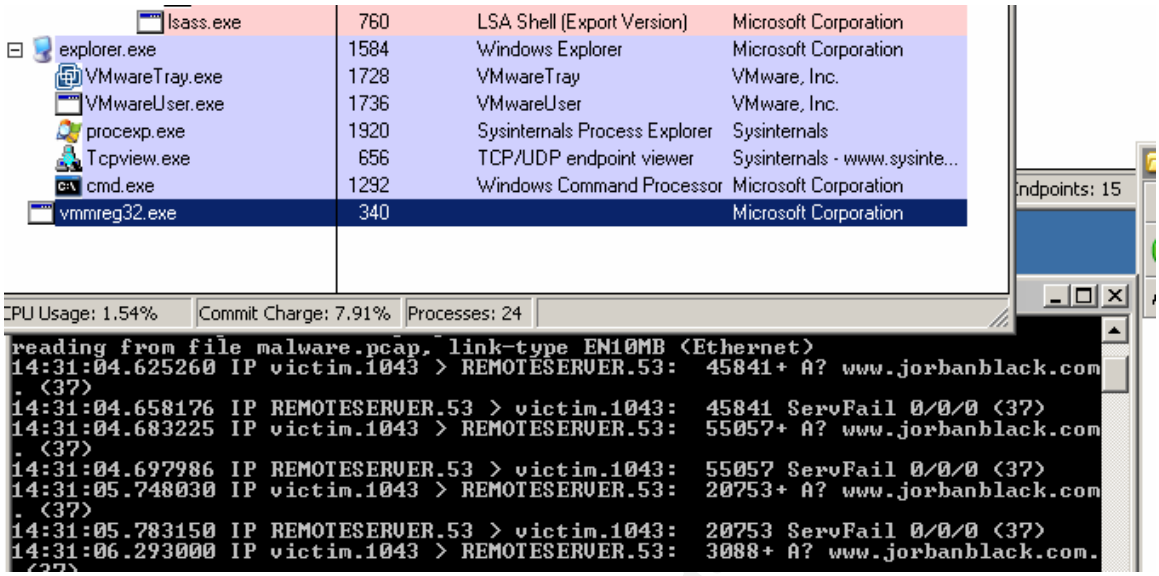
By reviewing the disassembled malware, notice the changes the malware will make to the system. For this piece of malware, a file will be deleted, a new process created, and at least two new registry keys installed.

After reviewing the static analysis, it is time to perform the dynamic analysis. One last time, ensure that *Host-Only* networking is being used. Next, open up and arrange Process Explorer, TCPView and Windump, so that all windows can be seen at the same time as seen in the screen shot below. Take a final snapshot before executing the malware.

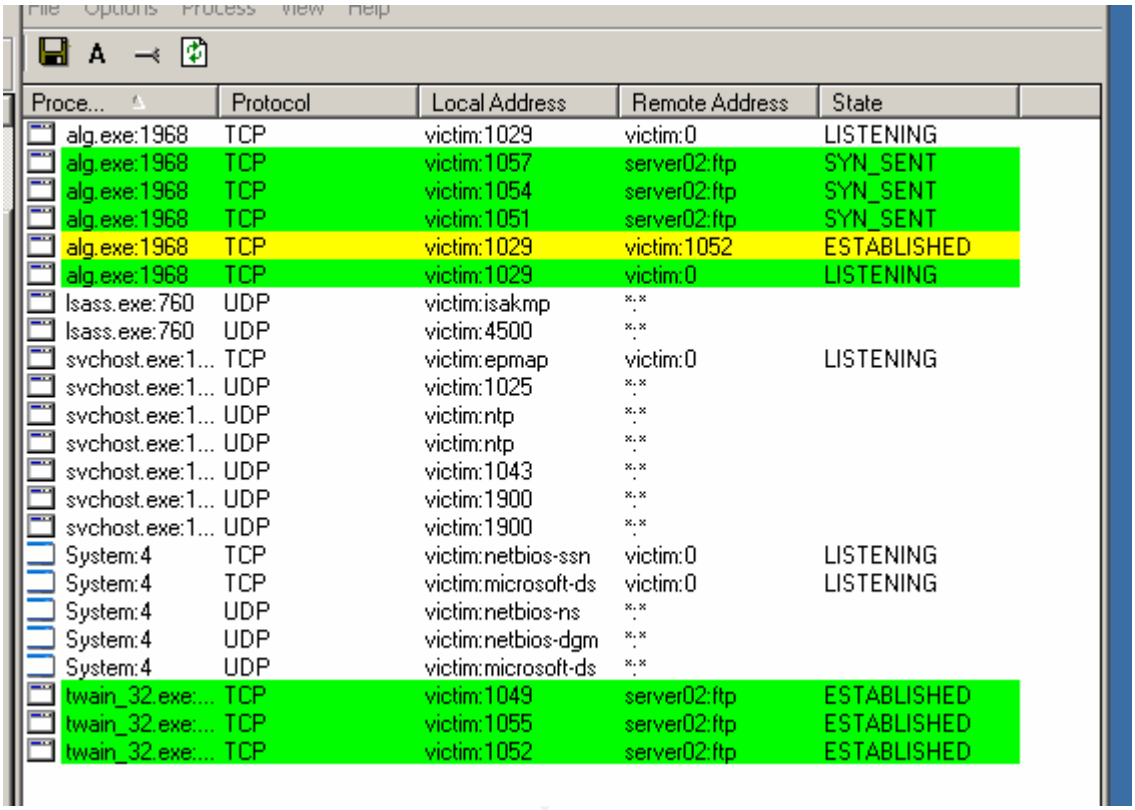


Now that the VICTIM host is base lined and having performed the static analysis, it is time to perform the behavioral analysis. Begin by executing the malware, watching the different applications for any changes that take place to the system.

The first thing to notice is a new process named vmreg32.exe is now running. Another item of interest is that the malware is attempting to connect the web site www.jorbanblack.com. Both pieces of information are shown in the screen shot below:



Create a DNS entry for www.jorbanblack.com to point to the Windows 2003 web server. Now that the malware can perform the DNS lookup, begin the malware again, and be sure that the monitoring tools are still running. Notice running the malware a second time the new process name is twain_32.exe this time. Watching TCPView the malware is attempting to make connections for common services such as WWW, SMTP, DNS, IMAP, FTP as seen in this screen shot:



Proce...	Protocol	Local Address	Remote Address	State
alg.exe:1968	TCP	victim:1029	victim:0	LISTENING
alg.exe:1968	TCP	victim:1057	server02:ftp	SYN_SENT
alg.exe:1968	TCP	victim:1054	server02:ftp	SYN_SENT
alg.exe:1968	TCP	victim:1051	server02:ftp	SYN_SENT
alg.exe:1968	TCP	victim:1029	victim:1052	ESTABLISHED
alg.exe:1968	TCP	victim:1029	victim:0	LISTENING
lsass.exe:760	UDP	victim:isakmp	*:*	
lsass.exe:760	UDP	victim:4500	*:*	
svchost.exe:1...	TCP	victim:epmap	victim:0	LISTENING
svchost.exe:1...	UDP	victim:1025	*:*	
svchost.exe:1...	UDP	victim:ntp	*:*	
svchost.exe:1...	UDP	victim:ntp	*:*	
svchost.exe:1...	UDP	victim:1043	*:*	
svchost.exe:1...	UDP	victim:1900	*:*	
svchost.exe:1...	UDP	victim:1900	*:*	
System:4	TCP	victim:netbios-ssn	victim:0	LISTENING
System:4	TCP	victim:microsoft-ds	victim:0	LISTENING
System:4	UDP	victim:netbios-ns	*:*	
System:4	UDP	victim:netbios-dgm	*:*	
System:4	UDP	victim:microsoft-ds	*:*	
twain_32.exe:...	TCP	victim:1049	server02:ftp	ESTABLISHED
twain_32.exe:...	TCP	victim:1055	server02:ftp	ESTABLISHED
twain_32.exe:...	TCP	victim:1052	server02:ftp	ESTABLISHED

After allowing the malware run for a few minutes, stop the network traffic capture. Using Windump review the traffic to see what ports that this malware is attempting to connect to. Create a listener on the victim server using Netcat on 5190 using the following syntax:

Nc –l –p 5190 –o c:\malware.log

Now that the malware can make connections with various “servers” listening on the www.jorbanblack.com website, run the malware again to see what traffic the malware sends to the “server”. After making connections to the remote servers and failing after a while the malware will encounter a problem and quit.

After reviewing the logs that Netcat generated, notice that the same data is being sent to various ports from the Victim system. This will led to, in my opinion, one of the most frustrating aspects of performing malware analysis: getting malware to run successfully as it is suppose to in the wild.

While performing malware analysis, the malware was not able to completely execute. However enough information has been gathered to build our network defenses. There will be times this will occur when performing malware analysis. The key to our analysis is that there is enough information to build the defenses.

12. Malware Defense

With the knowledge gained from the malware analysis, it is time to build defenses against the malware using multiple layers of defense. Remember when building defenses to use the defense-in-depth philosophy.

GIAC Rock's network will provide many services such as a web server, a mail server and an application server. However, the main focus will be on the defense of the network against this particular piece of malware. To protect GIAC Rock's network there are four layers of defense that are installed:

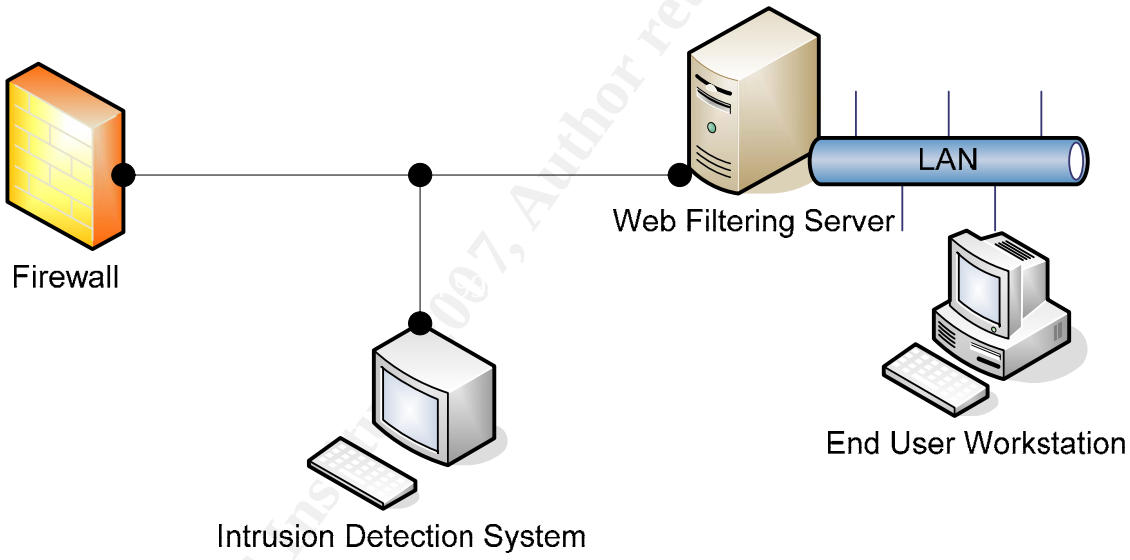
Firewall system

Web filtering system

Intrusion/Prevention Detection Systems (IDS/IPS)

Host base Intrusion Prevention Systems (HIPS)

Below is a network diagram of how the GIAC Rock network is setup. Notice that all outbound traffic from the end-user workstation must go through the web-filtering server. Also take note of the IDS placement to see all traffic being sent to the Internet.



Per the GIAC Rock Incident Response Plan, containment will begin by using firewall access-list to block all outbound traffic with the destination IP address of 208.72.170.203. Also GIAC Rock Incident Response Plan allows for the blocking of all inbound traffic from a known infected server, in this case the server with the source IP address of 208.72.170.203.

Because of the analysis, it is known that if the malware can not connect to TCP port 5190 it will attempt to connect to other ports on this server. By blocking the whole IP prevents the malware from ever connecting to the www.jorbanblack.com server.

The next step during the containment phase will be to add the jorbanblack.com domain to the web filtering software. This will prevent the users from accessing the website itself.

The final step during the containment phase is to block the installation of the registry keys found during the malware analysis via GIAC Rock's deployed HIPS. This will cause the malware to fail during installation.

With containment now completed, the eradication phase of the Incident Response Plan can begin. GIAC Rock's plan requires that all infected host must be rebuilt, and any restored data must be scanned with AV software before the recovery phase of the Incident Response Plan can begin.

The last phase of GIAC Rock's Incident Response Plan from a defense building stand point is to monitor for re-infection of the malware. There are two layers that will be used during this phase: IDS and Firewall log monitoring.

First an IDS rule will be created to alert network administrators any time a connection attempt is made to the jorbanblack.com server. It is very important that the IDS rule is as

specific as possible. Here are some specifics that will be but in an IDS rule:

Destination IP address 208.72.170.203

Protocol TCP

Destination Port 5190

TCP Flag SYN

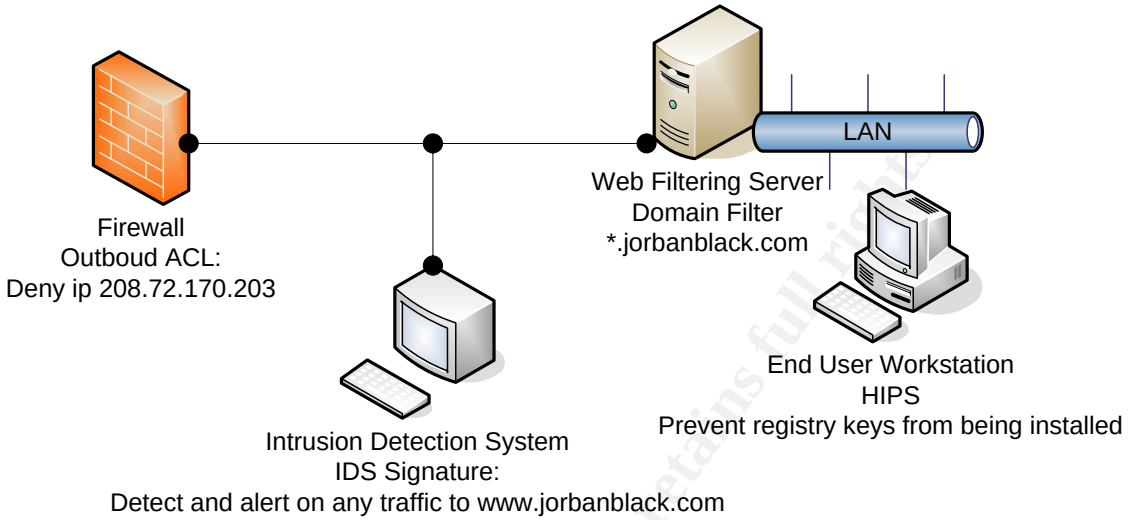
The above rule will alert whenever a connection is made to the www.jorbanblack.com server. However, this rule will miss if the IP address of the jorbanblack.com server changes. This will mean that the IDS rule will have to be modified if the IP address does change.

The final phase of recovery will be to monitor the firewall logs. A rule should be created that denies any attempt to the IP address 208.72.170.203. If this rule is alerted the host that made the attempted connections should be investigated.

By using GIAC Rock's Incident Response Plan, systematically defenses were built to protect the GIAC Rock network from infection of this malware. If the malware were to get into the GIAC Rock network, their defense-in-depth philosophy would not allow for this malware to successfully spread. This shows how malware analysis can bring added value to a company.

For a better understanding of how the defense-in-depth philosophy is employed at GIAC Rock's against the one-time.exe malware the diagram below shows the multiple layers,

the roles the layers play and the rules that should be applied to each layer:



By taking the defense-in-depth philosophy to defending GIAC Rock's network against the one-time.exe malware a single, double, or even triple failure will ensure this malware will not be successful in their environment. This example shows not only how malware analysis brings value to an organization but how defense-in-depth brings value too.

13. Conclusion

The reader should now have a basic understanding of malware analysis as well as a view of the exciting and ever evolving field of malware analysis.

Throughout the course of the paper, critical elements have been discussed which began with the key terms of malware analysis. This purpose has been two-fold: to allow the reader to see the underlyings of how malware analysis fits into an organizations' Incident

Response Plan as well as shed light on the value malware analysis brings to an organization. By defining and identifying malware analysis, tools used to perform the analysis, an introduction to malware analysis has been provided. Detailed methodology is outlined so that the reader can perform a successful analysis on malware, as well as build defenses with information gathered in the analysis to protect their organization utilizing the defense-in-depth philosophy.

There are many great references, many which have been used in this paper, that are available about malware analysis. Training is available as well from SANS on malware analysis.

14. Credits

There are several people that I would like to thank for their help during the writing of this paper.

I would like to thank my GSEC advisor, Charles Hornat, for all of his guidance during the preparation and documentation. His knowledge and resources have been invaluable.

I would also like to thank Matt Jonkman from Bleedingthreats.net for providing me with access to the malware that was analyzed for this paper.

And finally a very special thank you, to my wife, for all her continued love and support.

15. References

Computer Economics, 2007 Malware Report: The Economic Impact of Viruses, Spyware, Adware, Botnets and Other Malicious Code, Retrieved 2007, November 23 from http://www.computereconomics.com/article.cfm?id=1225

Eldad Eilam, (2005). Reversing: Secrets of Reverse Engineering. Indianapolis, IN: Wiley Publishing.

eWeek, Metasploit Creator Releases Malware Search Engine, retrieved 2007, November 24 from http://www.eweek.com/article2/0,1759,1990158,00.asp

GIAC, Analysis of the Incident Handling Six Step Process, Retrieved 2007, November 24 from http://www2.giac.org/resources/whitepaper/network/17.php?id=17&cat=network

Honeynet, Know Your Enemy: Malicious Web Servers, Retrieved 2007, November 24 from http://www.honeynet.org/papers/mws/KYE-Malicious_Web_Servers.htm

Lorna Hutcheson (2006), Malware Analysis The Basics, Retrieved 2007, November 24 from http://isc.sans.org/presentations/cookie.pdf

Merriam-Webster Online. Retrieved 2007, July 23rd, from www.m-w.com

SANS, Retrieved 2007, November 24, from https://www2.sans.org/training/description.php?cid=799

Ed Skoudis, & Lenny Zeltser (2003). Malware: Fighting Malicious Code. Upper Saddle River, NJ: Prentice Hall PTR.

Peter Szor, (2005). The Art of Computer Virus Research and Defense. Upper Saddle River, NJ: Symantec Press.

Techtarget.com, Retrieved 2007, November 24 from http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci1121085,00.html