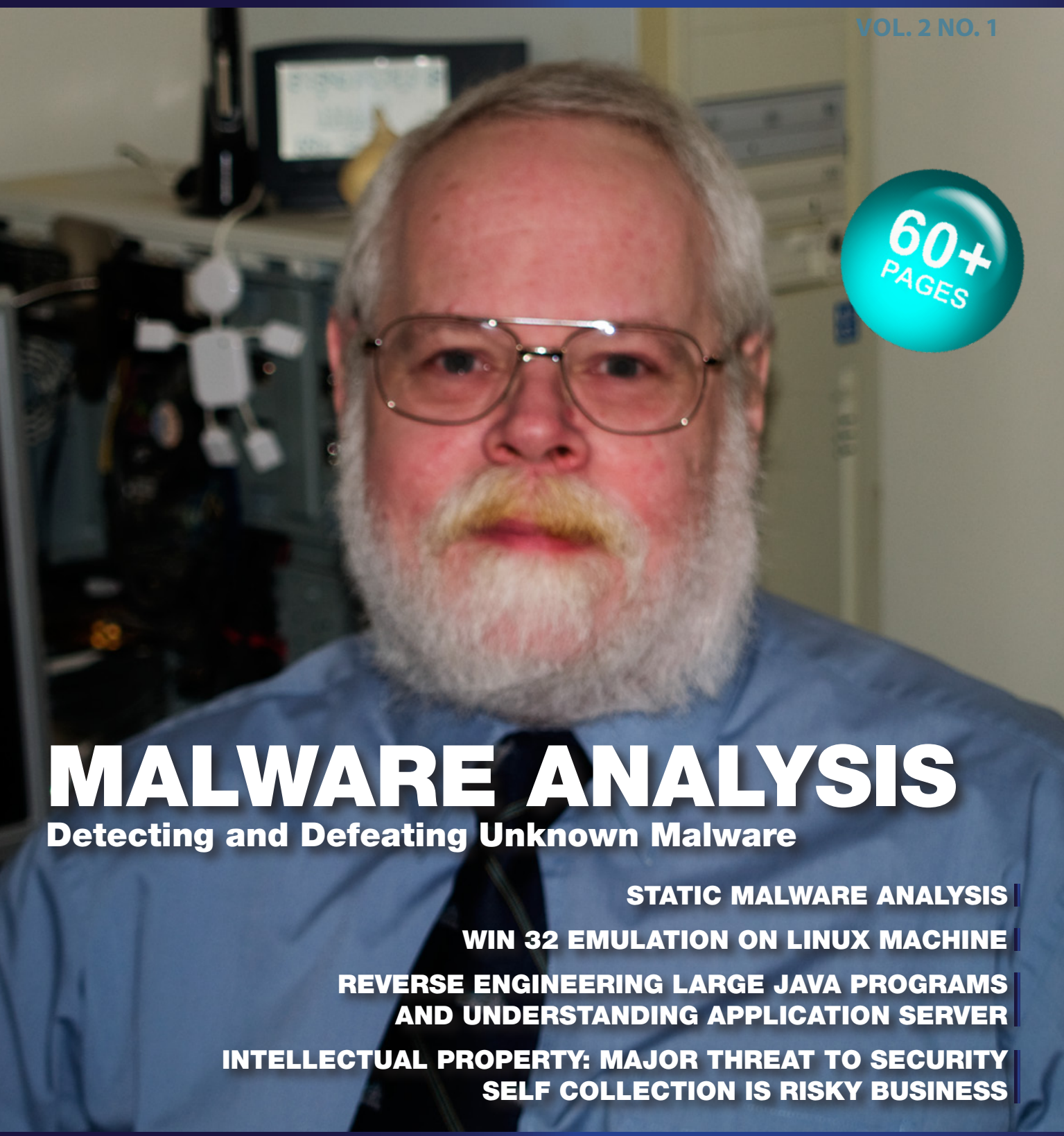
eForensics Magazine

COMPUTER

VOL. 2 NO. 1



MALWARE ANALYSIS

Detecting and Defeating Unknown Malware

STATIC MALWARE ANALYSIS

WIN 32 EMULATION ON LINUX MACHINE

REVERSE ENGINEERING LARGE JAVA PROGRAMS AND UNDERSTANDING APPLICATION SERVER

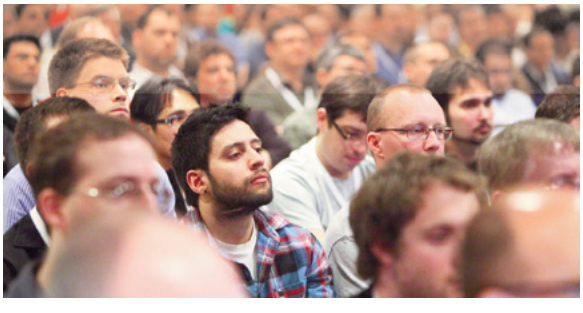
INTELLECTUAL PROPERTY: MAJOR THREAT TO SECURITY

SELF COLLECTION IS RISKY BUSINESS

Issue 01/2013 (5) January

Big Data gets real at Big Data TechCon!

The HOW-TO conference for Big Data and IT Professionals



Discover how to master Big Data from real-world practitioners – instructors who work in the trenches and can teach you from real-world experience!

Come to Big Data TechCon to learn the best ways to:

- Collect, sort and store massive quantities of structured and unstructured data
- Process real-time data pouring into your organization
- Master Big Data tools and technologies like Hadoop, Map/Reduce, NoSQL databases, and more
- Learn HOW TO integrate data-collection technologies with analysis and business-analysis tools to produce the kind of workable information and reports your organization needs
- Understand HOW TO leverage Big Data to help your organization today

Over 50 how-to practical classes and workshops to choose from!

BigData TECHCON

April 8-10, 2013
Boston, MA
www.BigDataTechCon.com

Register Early and SAVE!

A BZ Media Event

SPTechCon
The SharePoint Technology Conference

March 3-6, 2013 → San Francisco

Get the scoop on SharePoint 2013!

Register Early and SAVE!

The Best SharePoint Training!

Choose from over 90 Classes & Workshops!

Check out these NEW! classes, taught by the industry's best experts!

- How to Install SharePoint 2013 Without Screwing It Up — Todd Klindt and Shane Young
- What IS SharePoint Development? — Mark Rackley
- SharePoint Performance: Best Practices from the Field — Jason Himmelstein
- Creating a Great User Experience in SharePoint — Marc Anderson
- Ten Best SharePoint Features You've Never Used — Christian Buckley
- Understanding and Implementing Governance for SharePoint 2010 — Bill English
- Building Apps for SharePoint 2013 — Andrew Connell
- SharePoint Solutions with SPServices — Marc Anderson
- Lists: Used, Abused and Underappreciated — Wes Preston
- Planning and Configuring Extranets in SharePoint 2010 — Geoff Varosky
- Creating Simple Dashboards Using Out-of-the-Box Web Parts — Jennifer Mason
- Integrating SharePoint 2010 and Visual Studio Lightswitch — Rob Windsor
- Solving Enterprise Search Challenges with SharePoint 2010 — Matthew McDermott
- Getting Stuff Done! Managing Tasks with SharePoint Designer Workflows — Chris Beckett
- SharePoint 2013 Upgrade Planning for the End User: What You Need to Know — Richard Harbridge
- Ten Non-SharePoint Technical Issues That Can Doom Your Implementation — Robert Bogue
- SharePoint MoneyBall: The Art of Winning the SharePoint Metrics Game — Susan Hanley
- Intro to Branding SharePoint 2010 in the Farm and Online — Randy Drisgill and John Ross
- How to Best Develop Requirements for SharePoint Projects — Dux Raymond Sy

Lots more online!

A BZ Media Event

Central Contractor Registration

Follow us: twitter.com/SPTechCon

www.sptechcon.com

eForensics Magazine
TEAM

Editors: Stanisław Podhalicz,
Joanna Kretowicz
jaonna.kretowicz@eforensicsmag.com

Betatesters/Proofreaders: Roxana Grubbs, Colin Renouf, Gavin Inns, Gabriele Biondo, Vaman Amarjeet, Kevin McAleavey, Salvatore Fiorillo, Hussein Rajabali, Serrhini Mohammed, Stanley Samuel, Miguel Guirao, Alex Rams andJeff Weavern

Senior Consultant/Publisher: Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic@software.com.pl

Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

DTP: Ireneusz Pogroszewski

Production Director: Andrzej Kuca
andrzej.kuca@software.com.pl

Marketing Director: Joanna Kretowicz
jaonna.kretowicz@eforensicsmag.com

Publisher: Software Media Sp. z o.o. SK
02-682 Warszawa, ul. Bokserska 1
Phone: 1 917 338 3631
www.eforensicsmag.com

DISCLAIMER!

The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.

Dear Readers!

We would like to present you with the latest issue of eForensics Computer Magazine!

Taking advantage of its publication, I would like to welcome a number of our new readers who have joined us lately. It's not easy to become a forensics expert, but our community is growing stronger and the interest in digital forensics is more and more visible. We hope to help cyber security communities develop their techinques that make our computers safe and secure in the modern, industrial world. In the opening article, antimalware expert Kevin McAleavey provides us with the inner secrets of malware analysis. This is a long, detailed and interesting article that I would gladly recommend. Since the malware can not only threaten the process of conducting investigations, it can also threaten the evidence obtained from those investigations itself. Evidence tainted by unknown and undetected malware has acquitted those prosecuted, and investigations of malware attacks have failed because of the inability to locate the direct evidence of an attack. Your travel through the world of malware continues with our expert Ram Shmider who takes a differen approach to the subject. To start your malware analysis journey you need to keep in mind that the files or machine you are working on are infected with real live malware. As your antimalware tools might not be able to detect and stop this malware, it's important to learn basic steps that a malware analyst would take when interacting with the infected environment. Our expert leads how to set your own isolated lab for tests up. We received a lot of your requests about publishing more articles touching upon Linux. We respond to your request with the article of Alexey Lyashko who leads you further with subject of malware; presenting Emulation of Win32 Environment for Malware Research on a Linux Machine. He explains why Linux may be a powerful utility for unpacking and decrypting protected malicious executable files, taking memory dumps, etc. You will extend your knowledge of how processes are created and managed in both Linux and Windows operating systems. All this is the tip of the iceberg. In fact, the deeper you delve, the more you achieve.

In the next set of articles we chaange the subject and our dear friend and contributor, Colin Renouf, explains how to reverse engineer complex enterprise Java applications. He looks at the tools and techniques, and then applies them to understanding how a common enterprise level application server works – allowing us to relate the applications running upon it to the application server itself and the platform on which it runs.

In the last two articles we cover more theoretical subjects. As well as identifying direct evidence of a crime, digital forensics can be used to identify sources, for example, in copyright cases. According to Jason Pfoutz, intellectual property problems can become a major threat to security. Do you think that everything should be free or maybe individuals should get what they paid for? The answer may seem to be simple but in reality it's not.

In the last article our expert, Elias Psyllos, poses a question whether Certified Forensic Examiner help should be used every time in any electronic discovery matter. Whenever a matter that requires the collection or preservation of Electronically Stored Information arises, the first thought of most companies is to have their internal IT department involved. The author explains the risks involved while performing self-collection.

I would like to thank you for all support and feedback that we receive from you in your messages. As it's the first issue in 2013, we would like to ask you for a feedback concerning our work. We would be more than pleased if you could respond to these questions: Are we on the right track? What are your expectations towards the magazine? Which topics are you most interested in? We are here for you and as we repeat, the everyday reader's opinion is the most valuable for us. It is You who creates eForensics!

Joanna Kretowicz & eForensics Team

MALWARE ANALYSIS: DETECTING AND DEFEATING UNKNOWN MALWARE

by Kevin McAleavey, The KNOS Project

Cyber-attacks against control systems are considered extremely dangerous for critical infrastructure operation. Today, the protection of critical infrastructures from cyber-attacks is one of the crucial issues for national and international security. Over the past ten years, intrusion detection and other security technologies for critical infrastructure protection have increasingly gained in importance.

What you will learn:

- How to locate suspicious programs and how to determine if they're malware or benign
- Using file hashes in order to verify the origin and authenticity of suspicious programs
- How to locate the startups for malware, all of the associated components and locations where malware can hide
- How to locate rogue services and unkillable malware processes and regain access to the system
- Useful tools to aid in the diagnosis and mitigation of malware

What you should know:

- Familiarity with use of the REGEDIT Windows Registry editor
- Familiarity with use of the Windows CMD Command line
- Familiarity with the normal layout of a Windows file system

According to antivirus vendor Symantec, over 1 million new pieces of malware are created every day. In 2011, Symantec saw over 403 million new malware samples according to Kevin Haley, Director of Security Technology and Response at his "Symantec Security Awareness" presentation in October 2012. The samples received by Symantec and other antivirus and antimalware vendors are analyzed primarily by automated systems, and occasionally by human analyst's to become "signatures" placed in their detection databases on a daily basis.

Despite all these known samples, it is common for malware to slip right past security solutions undetected and unmitigated, leaving more system's infected with each passing day despite these efforts. Those of us who have been fighting the "war on malware" know full well that we've been losing the battle badly, and so do Information Technology Administrators and Managers.

When malware strike's a computer, the obligatory response by IT Department's is to remove the machine, wipe and reformat the hard drive, reload a fresh copy of the operating system and then reimage the drive, with the normal compliment of authorized applications and configurations, whereupon it is returned to the victim in a known and "trusted" state. While this causes great inconvenience to the victim, who has now lost whatever work that may have been on the machine, it is the only practical mean's to remove malware, given that the antivirus and antimalware industry has been losing the battle against malware for years now.

In some situations however, the importance of being able to collect evidence or critical data from a machine which has been infected with unknown and undetected malware, requires technician's with forensics experience to collect the valuable data and preserve it without spreading the malware on the infected machine further. It becomes quite challenging when malware has caused the machine of interest to become inoperable or inaccessible.

In this article, I will explain numerous ways by which malware can be located and circumvented sufficiently in order to be able to access and recover critical data where required. I caution in advance that being able to successfully restore the machine to operable condition does not mean that there isn't additional malware which remains hidden and continues to pose a risk. I will also refer to "security software" packages from various vendors generically as "antivirus" for the sake of simplicity.

Once a machine has been compromised it can no longer be "trusted", since it is impossible to be certain that any cleanup is ever complete. Attempting to access data from an infected machine must be carefully considered, as to its values and risk's, before proceeding.

Each type of malware is different, and will have different effects on different machine's. Some are easily located and mitigated, some are highly complex and difficult to disable, and some are downright destructive. Worse, there are so many locations in which they can hide within' the Windows operating system, protecting them from easy discovery.

The one thing that all malware have in common is finding a means to plant a payload onto the victim's machine, and then trying to remain hidden from detection. It is the payload that must be found, and in most cases, the file or means by which the payload was "dropped" onto the system in the first place, in order to prevent the payload from reappearing. Additionally, that payload may also consist of "helpers" in order to keep the malware hidden.

We will assume of course that all security and antivirus software is in place, and fully up to date on any machine in question. And in the case of a forensic's investigation that a proper image of the original content's of the machine has already been completed.

It is commonplace for even the best security software to fail to perform its duties and detect all intruders. Security software depends on specific "signatures" in order to identify malware, and changing just one digital bit in a malware file will cause it to slip right past ordinary antivirus/antimalware "signatures".

Malware author's carefully test their code in order to ensure that it does not match, and therefore " trigger" AV "signature's" before they release each version, in order to take advantage of this inherent design weakness. Thus, in order to find "unknowns" it is necessary to use the same skill set that malware analyst's use to hopefully locate and defeat them.

MALWARE TYPES

The computer security industry has a history of confusing definitions for the various classes of malware, often based on the limitation's of a particular vendors choice's or design limits in their coverage of malware. As a result, there is much confusion as to the specific meaning's of various definition's, and proper classification of specific threat's. Therefore, I'd like to explain what those of us in malware analysis labs define specific malware types as, and their expected capabilities, regardless of individual vendor naming policies. For those who are already certain of the definitions I'm about to present, please feel free to skip to the next section.

VIRUSES

Viruses are a self replicating program that infects individual files on a computer. They modify the contents of one or more file's, and usually hide inside legitimate files. In the old days, they would append themselves to the end of a file, making it larger than the original file. In more modern versions of Windows, compilers leave a lot of dead space inside legitimate files, making it easy for viruses to copy their payload into those empty spaces, thus leaving the size of the file itself unchanged and harder to detect. Many viruses will also avoid modifying the date and time stamp, further complicating forensic detection. They will change the entry point address in the file header to point to the viral code, so in order that this new code will run first before calling the original content's.

A practical way of checking on whether a file contains a virus is to use an MD5 or SHA1 "hash" against a known good copy of the original file. File hashing is the "secret sauce" of the antivirus industry, its how most AV signatures are created and how samples are managed and shared within the industry. This hashing method is extremely useful for efficiently investigating malware and will be detailed later in this article.

WORMS

Though closely related, worms and viruses are two entirely different types of malware. Both have the ability to self-replicate and propagate by attaching themselves to files although not all worms are "file infectors." However, while viruses copy themselves from machine to machine through media such as a USB device, worms replicate through networks directly. Worms can travel through the internet or local networks and inflict mayhem ranging from deleting files to creating backdoors or botnets that provide remote control of a system. Worms are designed to perform this function autonomously without human assistance, through network's or through physical media.

BOTS

"Bot" is derived from the word "robot" and are automated malware that controls network communications or services. Bots often automate tasks and provide information or services that would otherwise be conducted by a human being. A typical use of bots is to gather information, and then interact automatically with *instant messaging* (IM), *Internet Relay Chat* (IRC), or other web interfaces. They may also be used to interact dynamically with websites, and recent ones have featured highly customized protocols of their own.

Bots are self-propagating malware designed to infect a host and connect back to a central server or servers that act as a *command and control* (C&C) center for an entire network of compromised devices, or "botnet." With a botnet, attackers can also launch broad-based, coordinated "remote-control," and flood-type DDOS attacks against their target(s).

TROJAN HORSES

Trojan horse malware are programs which appear harmless, and may also appear to be useful and completely functional. However, they will contain other components which will install malware onto the machine while keeping the victim busy with a diversion, or appearing to fail, causing the end user to think the trojan was merely defective in some way.

They might arrive as an email attachment, or present themselves as a useful application on a website enticing the user to download and install them. Because it does not have the ability to self replicate, trojans are a completely different animal from viruses or worms. Trojans require human assistance in order to spread, usually through “social engineering” means. They often deliver destructive payloads and can also install other types of malware.

RATS AND “BACKDOORS”

RATs are “Remote Access Trojans” also called “Backdoors”, and is the choice for espionage and exfiltration of data. They provide a "Backdoor" into the system through which external actor’s can remotely control a computer, running other malicious code if he/she chooses. They can even use these hijacked systems called “zombies”, to launch attack’s on other’s. RATs provide all the capabilities of legitimate remote access tools, and then add numerous other capabilities depending on the design.

These are the preferred tools of cyber criminals, and APT operation’s, and are designed to remain stealthy when in operation and often install other malware components as part of the overall infection. Their primary purpose is remote control of the individually infected machines through these “servers” which can be remotely accessed manually. RATs differ from bots in that RATs must be connected to individually whereas bots normally communicate collectively with other bots or controllers autonomously as a distributed network. Some RATs also offer bot capabilities, thus blurring definition’s among some vendors.

SPYWARE

Spyware, also called “adware” by some vendors, is the most commonly found malware on machines. Spyware is any program that tracks and reports your computing activity without consent. While it is not designed to inflict damage, spyware can seriously affect the performance of machines. Spyware is often bundled with free software, and automatically installs itself with the program you intended to use. Sign’s of spyware include sudden modifications to your web browser, unwanted additional searchbar’s or “toolkits”, or redirects of your search attempts and the frequent displaying of pop-ups.

Spyware and “adware” are often defined interchangeably, but “adware” simply displays advertisement’s, whereas spyware returns more detailed information than just a cookie to the site, which displays the ad’s such as browsing history detail’s, or other personally identifiable information about surfing habits. Spyware should not be confused with more serious malware like keyloggers, RATs or other exfiltration tools, these are largely a privacy issue.

Many antiviruse’s do not detect all spyware, given that numerous purveyors of spyware will sue in order to have the "false positives" removed from detection databases in security products. When investigating strange, undetected program’s on machines, spyware is often found and can prevent analysis from continuing further when actual malware exists. Therefore, when spyware is found, diagnosis should not end just because some spyware was found on a victim’s machine, it is often a sign that the hunt has only begun.

KEYLOGGERS

Keyloggers are a particularly nefarious type of malware, in that their purpose is to surreptitiously collect critical information such as; logins, passwords or other sensitive data which is manually entered from a client machine. Keyloggers will then record all keystrokes entered on the victim’s machine into a file, which can be transmitted back to the source of the infection or to a botnet controlled by that source. Some more advanced keylogger-type malware will also index and collect for transmission, critical files on the victim machine or the network’s to which it connects as part of a larger espionage campaign.

DROPPERS

Droppers are a compressed package of malware. Their design dictates that they be as small as possible, in order to not raise suspicion of a surreptitious download in progress. They are often hidden inside email, document’s, or by other means and

are quite compact. Their only purpose is to gain entry into a system, and once installed they will download other component's quietly in the background, or they will contain various file's needed for a successful infection. Whereupon, they will install all of the other pieces that malware requires to carry out its function. Droppers are usually compressed with a specially obfuscated "wrapper" that is designed to elude detection by antiviruse's by means of encryption and other techniques.

ROOTKITS

Rootkits are the most difficult of all malware to detect, as their purpose is to completely hide malware from the user as well as security software intended to detect malware. Rootkits operate at the system level and are designed like other system and hardware drivers, using special code in order to hide the malware installed at the user level. Some rootkits are almost impossible to detect, even with forensics, and run as system services, device drivers, or can even be written to firmware in the system's hardware such as; video cards, network cards, BIOS, or any other device which allows updating of the hardware's "firmware." Their entire function is to hide malware, and the only visible symptom of their presence is unexplained reboots, or blue screen crashe's, if they have any bugs in their code. Absent programming bugs however, are extremely difficult to detect. Rootkits run at the lowest levels of a particular operating system for which they are designed.

BOOTKITS

Bootkits are the latest concern, although despite the hype, their practical application is still largely theoretical and impractical. However, when the specific hardware configuration in the potential victim's computer is fully known, it is practical to construct one. Bootkits consist of custom code designed to function within a system's "firmware" such as; BIOS, printers, video cards, ethernet and WiFi device's, and any other hardware that is capable of having its firmware "flashed" or upgraded. Bootkits are differentiated from rootkits because a bootkit does not require the services of an operating system they are completely self-contained independently of the operating system, whereas rootkits are designed to work after the OS has been booted.

As a computer boots, BIOS (or in newer machines, EFI or UEFI data on the local hard disk) contains pointers to the various boot firmware in modern components. By placing malware in the hardware itself, a "bootkit" can be started before the operating system and maintain control throughout a session. However, any such "bootkit" has to be written specifically for that hardware, and will not work on another version of that hardware, owing to the variety of hardware designs. The term "bootkit" is also often referred to as, "boot sector infectors" though they're still technically considered a "virus", and because they run independently of the operating system, antivirus software is highly likely to never detect them at all.

Bootkits can function regardless of the operating system that is booted, and therefore a very valuable tool for APT and other situations where the expense of such custom malware justifies the effort. It was one of many considerations in our own KNOS design despite the current rarity of this threat.

Bootkits can only be defeated by clearing and reflashing firmware with tools provided by the manufacturer of the hardware itself. Even then, the hardware should no longer be considered "trustworthy" since most "bootkits" are placed following the last byte of firmware code in the device, and many "reflash" tools fail to overwrite the infection.

PSEUDO-ROOTKITS

I coined this term back around 2000 with our BOClean product to describe perfectly legitimate software which has been installed onto a machine in order to act as malware. Pseudo-rootkits are legitimate program's which were installed without authorization and when audited, will turn out to be completely legitimate. They include things such as FTP software, Remote desktop software, keyloggers used by parent's and corporation's to monitor use of machine's, chat software, torrent software and numerous other "name brand" products that security software will not detect because it's a "legitimate tool."

However, they are not installed in the normal location where they would be installed legitimately and have been modified with other "legitimate tools" to remain hidden while they're in use by remote actors. Pseudo-rootkits will not be listed in "Add/Remove software" since they weren't installed by legitimate means. Our former BOClean product was the only security product which set off warnings when this was the case, warning the user that such was operating without their knowledge and gave them the option to remove them. Most security vendors will not even attempt to detect these at all even if they are "out of place."

EXPLOITS

Exploits are malware which are designed to exploit weaknesses and design errors in existing software or operating systems and are often embedded in web sites, documents and file servers which consist of scripts such as; Active X, Java applets, Javascript, PDF files and multimedia file's which will, thanks to poorly written code, perform unauthorized actions on the victims machine. Persistent exploits require that a copy be stored locally, usually in the internet browser's cache file's, or in the TEMP space. There are numerous other examples of exploits as well, which are triggered each time a malware page is visited.

WARNING SIGNS THAT UNDETECTED MALWARE MIGHT BE PRESENT

The majority of malware is poorly written and will often noticeably affect the computer. When malware fails to be detected by your antivirus or other security software, the following symptoms merit further investigation:

- Degraded computer functionality.
- Antivirus, firewall or other security software has been disabled.
- Odd behavior, such as unexpected reboots or icons no longer responding.
- Popup windows appearing when not browsing the internet.
- System errors, blue screens, or reliable programs suddenly crashing.
- Strange traffic on the network, as well as slow browsing.
- Disabled functions such as Task Manager, Regedit, User switching, login, logout or shutdown.
- Files or programs on your PC that you do not recognize.
- While surfing the internet, certain sites such as *www.microsoft.com* or sites with antivirus, or other security software vendors do not work.
- There are folders in your Windows Explorer, but clicking on them doesn't open them.
- After a reboot, Windows reports a Data Protection Violation in "Windows Explorer", and shuts down Explorer to restart it right away.

HUNTING FOR MALWARE

Unplug your network cable and manually turn your computer off. Reboot your computer into "Safe Mode with Command Prompt". As the computer is booting tap the "F8 key" continuously, which should bring up the "Windows Advanced Options Menu", as shown below. Use your arrow keys to move to "Safe Mode with Command Prompt" and press Enter key.

Make sure you log in to an account with administrative privileges (login as admin).

Once the Command Prompt appears you have only a few seconds to type in *explorer* and hit Enter. If you fail to do it quickly within 2-3 seconds, Malware on the system is likely to take over and not let you type anymore. Keep trying if necessary until you succeed.

If you managed to bring up Windows Explorer you can try to run System Restore with the following commands:

- Win XP: C:\windows\system32\restore\rstrui.exe and press Enter
- Win Vista/Seven: C:\windows\system32\rstrui.exe and press Enter

Follow the steps to restore your computer into an earlier day prior to the infection if you have any idea as to when it first started showing symptoms. System restore often disables recently added malware and lets you get back into the machine. However, a large amount of malware disables your ability to get at many parts of your system. If this is the case, read below where I describe how to re-enable deliberately disabled functions.

If you were successful, then you can proceed with hunting for the malware. A system restore will only hopefully prevent malware from starting. It will still be present on the machine, hopefully inactive at least temporarily in order to allow you to find and remove it.

Now the hunt can begin. This can become quite involved, but generally you'll want to proceed in this order:

- Clean all temp folders and internet caches from all browsers.
- Perform an Antivirus scan while running in Safe Mode.
- Audit running processes.
- Audit the 'startup' files and registry entries.
- Audit network connections.
- Check the file system for files that have been hidden and REALLY hidden files.
- Check for unusual services.
- Audit the HOSTS file and Windows TCP/IP Settings (redirects to incorrect sites are often done by modifying these).

If the above doesn't solve the problem, then it's time to dive into the system manually. This is the hard stuff, and requires an absolute need to access the data on the infected machine since it can become quite labor and time intensive now.

MALWARE HUNTING THE HARD WAY

Look for odd processes such as normal service names which are misspelled. svchost.exe is expected, scvhost.exe is NOT. Random character file names are almost always guaranteed to be malware, and they change each time the system is rebooted in most cases.

Another frequent indicator of malware are filenames with double extensions such as program.pdf.exe where the double extension serves to trick victims into believing that the file in question is something other than an executable like in this case where it might appear to be a PDF file if "show file extensions" is disabled in the file explorer view.

Incorrect icons for files are also a reason for further investigation. With "show file extensions" enabled in the file explorer, executables which display a file folder icon, or an archive file icon or perhaps a document icon are highly suspicious. Legitimate programs will provide an icon that shows a product logo or an icon logically associated with that program and won't mislead the user visually as to what the icon represents.

Another clue which should be followed is to check for valid signatures for any executable which claims to have originated from a large vendor. Most major enterprises now sign their code and the absence of a signature, or misspelled “properties” information is also an important clue. Here, examining the “properties” for unknown programs is quite useful.

Unusual processes with high CPU utilization with seemingly legitimate process names which are unexpected are also a sign of possible infection by malware. The Windows Task Manager is usually circumvented by malware to prevent malware from appearing in the task list, however you might get lucky and spot some this way.

Searching the web using process names you are unsure about will give clues to the legitimacy of individual processes. There are a few sites that provide databases which will give a detail listing of processes, including file sizes and verdict as to whether they're legitimate or suspicious.

Installing a tool which will allow you to take an MD5 or SHA1 "hash" of any suspect files will make searches for those processes a whole lot saner and will make it easier to confirm whether the files are legitimate or suspect. Most malware search sites will use MD5 and/or SHA1 hashes in order to confirm their verdicts and thus being ready to determine the MD5 or SHA1 hash of suspect files is strongly recommended to save time. These tools include:

- Microsoft File Checksum Integrity Verifier (generates MD5 and SHA1 hashes) which can be downloaded here: *http://www.microsoft.com/en-us/download/details.aspx?id=11533*
- I prefer this tool myself: HASHCHECK, which installs as a shell extension right into Windows' file explorer: *http://code.kliu.org/hashcheck/*

WHY DO WE NEED FILE HASHING?

As I indicated earlier, "file hashes" are, the "secret sauce" of the antimalware business. By performing an MD5 or SHA1 hash on a file, a unique long number is generated that is unique to that specific file. It's a means of generating a quick and dirty "signature" for a known file whether it's legitimate or it is malware. It permits security software vendors to feed a file to their signature database without so much as looking at a sample that's been flagged as malware. No effort, no time, put it in the blender, out pops a new "definition." It's cheap, it's fast and it's lazy for the security vendors.

The downside with using hashes for antivirus signatures is that if one single BIT of code changes, due to encryption, a different packer, even adding blank characters to the end of a file making it one byte larger, that hash will no longer match the "known malware" signature and thus it will go undetected until the new variant gets hashed and added to that signature database. THIS is why your antivirus failed you! They probably saw a different copy than the one that just got installed on your machine and the signature will only spot that other identical copy, not necessarily yours. This is also the reason why the "body counts" are so high for malware and so many variant versions of the same exact malware in their definitions.

Years ago, our operation made a product called "BOClean." We actually examined each and every piece of malware in memory and then created a memory-based definition for malware that would always detect the malware no matter how it was repacked, encrypted, polymorphed or modified since all programs must shed all that once they are ready to run in memory on a computer.

As a result of this design, we could detect any variant knowing that there is a very limited number of coders who produce malware and we made it a point to study the authors themselves rather than their code and zeroed in on specific means of detecting the author. As a result, we detected everything new that they wrote without the need to add individual definitions. Even in the most sophisticated of “APT” malware, the authors always manage to leave some unique “signature” in their work that can be used to detect their next move.

When I was last directly in charge of antimalware labs two years ago, the number of "unique authors" was in the vicinity of only about 1,500 coders. I'd bet that the number today is less than twice that. But my premise was that each individual author had their quirks, and you could count on those showing up in the code they released. It was the secret of our reputation with BOClean. But it took a lot of work in the face of an ever expanding number of samples. Today you only see the security companies expending that level of money and effort on the likes of Stuxnet.

Since file hashes are the industry standard though, that's what they use for sample and definition sharing within the industry, as well as databases of malware publicly available to the public which you can use to research whether or not unknown files are already known to malware analysts or other vendors in the industry. MD5 hashes are the older standard, SHA1 hashes are the currently popular exchange information. So you will want to obtain both MD5 and SHA1 hashes of any files which you suspect and then use those values to perform internet searches at places like virustotal, jotti, or similar to determine if a file is legitimate or suspicious depending on the results of your search. Doing those hashes will save you a _lot_ of time in your hunt!

RELEASE THE HOUNDS, ON WITH THE HUNT!

In addition to the task manager that comes with Windows, these tools give far more detailed information for you to examine them and are less likely

to be fooled into hiding malware unlike the Windows task manager:

- GUI Process Explorer from Sysinternals: *http://technet.microsoft.com/en-us/sysinternals/bb896653*
- GUI Autoruns from sysinternals: *http://technet.microsoft.com/en-us/sysinternals/bb963902*

Lacking those tools, the next step is to look for odd entries in the "startup" sections of the registry using Windows' built-in tools. Keep in mind that legitimate programs are seldom found in multiple start locations, such as run, `runOnce`, `runOnceEx` and runservices, but it's quite common for malware to copy its startups to multiple locations in order to ensure that if one is found, others will successfully start the malware each time the machine is rebooted. If you are unable to access the registry or built-in tools, or are unable to run any programs at all, I will explain down below how to get around disabled functions and services.

TO LOOK FOR THE NORMAL STARTUP LOCATIONS WHERE MALWARE MIGHT BE STARTED

```
winkey + R | msconfig | Startup Tab
```

or

```
winkey + R | regedit | check the following
```

Audit the 'startup' locations. The most commonly used startup locations in the registry are:

- `HKLM\Software\Microsoft\Windows\CurrentVersion\Run`
- `HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- `HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnceEx`
- `HKLM\Software\Microsoft\Windows\CurrentVersion\RunServices`
- `HKCU\Software\microsoft\windows\CurrentVersion\Run`
- `HKU\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Run`

AUDIT NETWORK CONNECTIONS

WINKEY + R | CMD | NETSTAT –NAO

Connections continually in 'SYN sent' state on 445, 139, or other established connections on odd ports, such as 6667(IRC) established connections on typical but unexpected ports such as FTP, TELNET, even 80(http) where inbound connections aren't normally expected are a sign of malware running surreptitiously. Inbound connections to websites and other services normally occur on ports higher than 1024, but outbounds on ports below 1024 are highly suspicious when a server is not being deliberately run on the machine in question.

A list of TCP and UDP ports and their expected purposes are listed here: *http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers*.

Once again, sysinternals to the rescue: *http://technet.microsoft.com/en-us/sysinternals/bb897437*.

CHECK FOR HIDDEN FILES

There are files that are required for your system to run, many of these are hidden from accidental deletion. Malware takes advantage of such file attributes to hide from standard file searches, but the Windows folder view menu will allow you display any HIDDEN, SYSTEM and READONLY file settings on your system so that you can browse almost all of the files on the system being examined. There are also files known as "super-hidden files" which require the extra step of going into the View tab of the file explorer and specifically unchecking "Hide protected operating system files (recommended)" as well as ensuring that all other file types are made visible. Please be aware that malware can still hide their files from the File Explorer through the use of rootkits.

When performing this search pay particular attention to directories in the `%PATH%` variable such as C:\windows\system32, most malware tends to be placed in the system "path" environment setting and loaded via the registry without an absolute path. DO NOT delete any of the files listed by this command unless you are positive they are malware, you can easily hose a completely functional system by doing so. To delete files, you will need to UNSET hidden, system and read-only attributes first.

Be mindful also, that there are "super-hidden" files that Microsoft will just not let you access which begin with a $ sign as the first character of the filename. These are reserved for the system only and have been used to hide malware rootkits. One of the most infamous rootkits which I tracked down many years ago was the SONY rootkit, installed by DRM contained on their commercial music CD's. I wrote up a set of manual instructions for those who didn't use our BOClean product here: *http://www.dslreports.com/forum/remark,14817570*.

Malware can also hide in *Alternate Data Streams* (ADS), which hides files inside other files. One clever way of hiding malware, as well as purloined files in the act of espionage is to write the content's to an ADS, whereupon they're almost impossible to find without knowing the name of the ADS itself. These files are usually written to important system files and sometimes buried inside folder entries themselves rather than as files so as to elude detection. They will contain a colon (:) between two filenames such as for example winstart.exe:malware.exe or similar. The colon is the clue that you're dealing with an ADS file that will not appear in a directory listing. Most antiviruses do not look for these by default.

An excellent tutorial on how ADS works can be found here: *http://windowssecrets.com/top-story/hide-sensitive-files-with-alternate-data-streams/*.

And an even more useful description of what would be involved in deleting them can be read here: *http://www.bleepingcomputer.com/tutorials/windows-alternate-data-streams/*.

The following tool can locate them: Sysinternals Streams: *http://technet.microsoft.com/en-us/sysinternals/bb897440*.

Some malware will install as a system service. While removal of services is more difficult, you can usually at least disable them for temporary clean up:

```
winkey + R | msconfig | Services Tab
winkey + R | services.msc |
From the Command shell -> tasklist, taskkill,
tasklist /svc
```

Once malware has been identified, it's best to remove it while in safe mode. Some malware will have additional processes and DLL's (and possibly rootkits) that can prevent you from removing them by means of “injecting” their code into legitimate system processes. Deleting startup locations and files while in safe mode can usually restore a system to working condition but remember, there's no such thing as a "trusted machine" once it's been owned.

Listing 1. *Windows registry keys*

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServicesOnce
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServices
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnceEx
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SafeBoot\Option subkey
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\*service* >ImagePath
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\Currentversion\explorer\User Shell Folders
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\SharedTaskScheduler
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\Browser Helper Objects
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellExecuteHooks
HKEY_LOCAL_MACHINE\Software\Microsoft\Internet Explorer\Toolbar
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Active Setup\Installed Components
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\ShellServiceObjectDelayLoad
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnce
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\RunOnceEx
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Shell Extensions\Approved
HKEY_LOCAL_MACHINE\Software\CLASSES\batfile\shell\open\command
HKEY_LOCAL_MACHINE\Software\CLASSES\comfile\shell\open\command
HKEY_LOCAL_MACHINE\Software\CLASSES\exefile\shell\open\command
HKEY_LOCAL_MACHINE\Software\CLASSES\htafile\Shell\Open\Command
HKEY_LOCAL_MACHINE\Software\CLASSES\piffile\shell\open\command
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\BootExecute
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Terminal Server\Wds\rdpwd\StartupPrograms
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit
HKEY_CLASSES_ROOT\exefile\shell\open\command
HKEY_CLASSES_ROOT\comfile\shell\open\command
HKEY_CLASSES_ROOT\batfile\shell\open\command
HKEY_CLASSES_ROOT\htafile\Shell\Open\Command
HKEY_CLASSES_ROOT\piffile\shell\open\command
```

Finally, Rootkits: Rootkits are almost impossible to find although many aren't all that sophisticated. There are numerous tools which will attempt to detect a rootkit's files, but they're more prone to false positives than actually finding genuine rootkits. They include:

- Sysinternals Rootkit Revealer *http://technet.microsoft.com/en-us/sysinternals/bb897445*
- Hitman Pro (I recommend this comprehensive scanner which uses multiple AV engines to scan, but doesn't remove them) *http://www.surfright.nl/en*

Bear in mind that just because a "rootkit detector" fails to find any rootkits doesn't mean that there aren't any. Rootkits operate at the operating system level and can easily hide themselves completely since most security software operates at the user level, which is segregated from the operating system level. In Win7 and later, that isolation is even more profound than in earlier versions of Windows almost guaranteeing that rootkits will never be detected unless they're caught in the act of being installed.

LOCATING MALWARE WHICH MIGHT BE LOCATED IN STARTUP FOLDERS

The most ordinary autostart locations in Windows are in folders actually named "Startup." On all Windows computers, there is an individual Startup folder for every user (account) who has ever logged on in addition to a Startup folder shared by all users of a particular system. These folders can often be managed through the paths above or Windows Start button → All Programs → Startup → Right click. In XP they are:

- C:\Documents and Settings\<USERNAME>\Start Menu\Programs\Startup
- C:\Documents and Settings\Default User\Start Menu\Programs\Startup
- C:\Documents and Settings\All Users\Start Menu\Programs\Startup
- C:\windows\tasks

Whereas in Vista and later they've been changed to

- c:\ users\<USERNAME>\appdata\roaming\microsoft\windows\start menu\programs\startup
- c:\ProgramData\Microsoft\Windows\Start Menu\Programs\Startup

Startups can also be tucked away in these locations:

- C:\autoexec.bat
- C:\Windows\Win.ini
- C:\Windows\System.ini
- C:\Documents and Settings\Administrator\Local Settings\Temp\
- C:\windows\system32\
- C:\WINDOWS\Prefetch

In these file folders, if you opt to sort by date, you can sometimes see what has recently been touched, added or changed so long as the malware didn't change the particular file's time and date stamps. Malware will usually be listed with DLL, OCX, SCR, VBX, BAT, CMD or EXE file extensions, sometimes with random file names and in most cases, you should be able to search the internet for details by filename to determine if it's a known threat or not. This is when it's helpful to also know the exact file size in bytes as well as having an MD5 or SHA1 hash of the file in question to make matching your mystery easier.

If the filename doesn't turn up in an internet search, it's likely to be rogue, but again, you can't count on that either. Sometimes malware will use filenames similar to genuine system files such as rundll32, but will be located in the wrong directory. If you see something called rundll33 then it's a pretty safe bet that it's malware and should be deleted. System files are usually in SYSTEM32 and below, its presence in WINDOWS or TEMP is also suspect.

However, the vast majority of malware is started by entries in the Windows Registry, and here it gets dicey because there are just so many places that malware can be started from such as these particular Windows registry keys (see Listing 1).

The most difficult part of the registry hunt is that there are so many entries in each of them which culminates in an executable, DLL file or other library file being loaded and started. Unfortunately, each and every one has to be examined to determine whether it is legitimate or not. As I indicated at the outset, there are just too many places for malware to hide in the Windows operating system.

And these only account for EXE files. DLL's and other potential malware have other locations from which they can be started. Be particularly cautious when you spot a startup entry which begins with "RunDll" or "RunDll32" in front of a DLL, OCX or other file. RunDLL is Microsoft's "run a DLL as an executable" function which will allow malware to run as a library instead of an executable and thus loads it into the operating system itself instead of running it as a separate program. Any of those files must also be examined and verified as legitimate as well.

DLL's can also be started using entries called "ImagePath" or "ServiceDll" in the registry, and these are even harder to find because they're associated with "UUID" and "CLSID" entries buried in the registry. These can be searched for in the regedit utility and there are a LOT of them. These keys will directly call DLL's, OCX's and other library type files which can also contain malware.

To further complicate matters, Windows will not allow you to simply delete malware while it is run-

ning. Running processes and threads are protected by the operating system, and any attempt to delete it will be met with one form or another of an "access denied" message and any attempts to remove the rogue file will fail. The only way to delete a running file is to either kill the process (and even here, many protect themselves from this possibility) or to remove the corresponding startup entry in the startup folder and/or the registry and then reboot in hopes that there aren't other elements of the malware that will simply put the entry right back into the startup location and then restart after a reboot.

And it gets even worse with rogue DLL's and libraries which are used to "inject" code into other legitimate, running processes. Code injection has been the preferred method of infecting computers since DLL's cannot be killed, they must be "unhooked" by means of "object dereferencing" instead. The only other alternative to unhooking DLL's is to kill the process to which they've attached. Malware authors solved that problem as well by attaching them to numerous processes including core operating system processes themselves. Kill these processes and the machine remains infected, but spontaneously reboots or freezes in order to spank you.

Hopefully, the entire malware "system" is started by an executable which then loads its other components. If you find the main startup, you can hopefully defeat the rest of the chain upon a reboot by preventing the starter application from running. Pay particular attention to any startups however which include that "RunDLL" command in front of a library file - that may be the startup as well. If you can keep the rogue from starting after a reboot, that's the major part of the battle won!

SHUTTING DOWN ROGUE SERVICES

Since a Windows service can be turned off, Microsoft hasn't felt the need to let users delete services outright from the Services window. But services can cause all sorts of problems, whether they're unwanted add-ons to otherwise useful software, left behind by buggy uninstallers, or inserted surreptitiously by malware. So here's how to remove a service completely:

Open the Services window (services.msc) and double-click the service you want to remove. Highlight the text next to Service name (the first entry under the General tab) and press Ctrl-C to copy the name to the clipboard.

Next, open a Command Prompt window in Administrator mode and type the following at the prompt:

```
sc delete ("Rogue Service")
```

where (Rogue Service) (in quotes) is the name of the service you just copied. Press Enter, and if the removal was successful, you should see this message:

```
[SC] DeleteService SUCCESS
```

Return to the Services window and press F5 to refresh the list, and confirm the service is now gone. SC (or "Service Control") is an often neglected, but useful way to control running services, and can often stop rogue services.

UNKILLABLE PROCESSES

Antivirus and other security software require extraordinary means in order to kill rogue programs. Some methods involve the abrupt "TerminateProcess" function in code, while others require an even more extreme step in providing a kernel device driver that will literally unhook the file from the system and then delete it. Absent special utilities designed to terminate unkillable processes, or unhook injected threads, if the task manager doesn't allow you to kill a rogue program, or it keeps coming back after an apparently successful "kill" there are a few methods which I've found can do the job without these specially written utilities. They include:

Use the "ntsd" command to kill any process that thinks it's special and can't be killed using the task manager:

```
ntsd -p [pid] -c
```

If that fails, then try this trick using the following exact command line in a command prompt window:

```
reg add "hklm\software\microsoft\windows nt\
currentversion\image file execution options\
malware.exe" /v Debugger /t REG_SZ /d "C:\Windows\
System32\bogusfile.exe /F /IM malware.exe"
(the quotes are necessary)
```

Where the name "bogusfile.exe" is a nonexistent filename, what this does is force Windows to attach a debugger that doesn't exist and that will cause the program (named "malware.exe") to begin to start but not run because it's waiting for a debugger program that doesn't exist. A reboot should keep the program from running.

Another potential malware killer is "RKILL" which can be found here: *http://www.bleepingcomputer.com/download/rkill/*.

FINDING MALWARE

Many malware programs try to keep users and administrators from interfering with their operation and possibly shutting them down by disabling numerous programs such as regedit, task manager, and in extreme cases, not allowing you to run anything nor even shut down the computer. This slight of hand is accomplished by changing file associations in order to ensure that any time you try to start a program, the malware will be started first, and then call the program you wanted.

They will also use permissions to disable access to the system whereupon you will see a message indicating that the "Administrator" has denied access to you. One of the methods of complete denial to any resources is "Association hijacking" which can prevent programs from running as well as being yet another automatic startup source for malware.

To check for association hijacking, you'll want to check these registry keys:

HKEY_CLASSES_ROOT\.exe\PersistentHandler
- (Default) value should equal: {098f2470-bae0-11cd-b579-08002b30bfeb}

HKEY_CLASSES_ROOT\exefile\shell\open\command
- (Default) value should equal: "%1" %*
- IsolatedCommand value should equal: "%1" %*

Hijacking can occur for OTHER file associations such as HTML, PDF and many others. You will see those association keys in the same HKEY_CLASSES_ROOT areas as ".exe" and "exefile" but their entries can often be more complex. It helps when checking to have the same registry entry open for cross-checking with a machine that is uninfected to confirm whether or not to edit the entry.

A cute little brute-force method that also works when programs cannot be started is to copy the original utility file and then rename its file extension. For example, if you can't start any EXE files, try renaming the copied ".exe" to ".com" or ".scr" which are likely to work. If that gets regedit or your antivirus program running again, you're home free.

Some additional tips to work around disabled functions can be read here:

- *http://dottech.org/11980/re-enable-critical-windows-components-disabled-by-malware/*
- *http://malwaretips.com/Thread-Remove-malware-when-traditional-tools-fail*

PREPARING FOR THE INEVITABLE

Install these tools on your machines BEFORE you need them!

Sysinternals complete suite of system forensics utilities: *http://technet.microsoft.com/en-us/sysinternals/*.

Being prepared in advance to be able to generate MD5 or SHA1 "hashes" on suspicious files against "known good ones" or to identify specific malware in your searches with the Microsoft File Checksum Integrity Verifier is highly recommended: *http://www.microsoft.com/en-us/download/details.aspx?id=11533*.

Once again, I prefer this tool myself: HASHCHECK, which installs as a shell extension right into Windows' file explorer: *http://code.kliu.org/hashcheck/*.

Making a recovery USB stick - look for USB memory sticks that feature a physical write inhibit switch so they don't get infected too. Having all of your tools handy on a CD, DVD, or USB stick allows you to bring your arsenal to the machine in question without risking installing from the internet on an infected machine. The "write inhibit" switch on USB devices equipped with one will prevent any malware from the machines you're working on from spreading via that USB device. Of course, having a bootable CD or DVD is even better because there's no possible way to write to it at all.

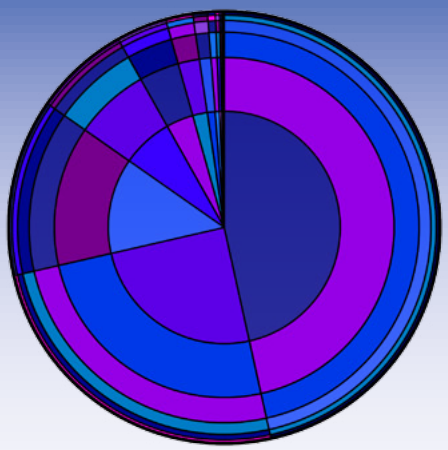
Most antivirus companies offer a downloadable "rescue disk" which you can download and burn to a CD, DVD or USB stick as well. Check with your security vendor to see if they have one available. The only downside is that each vendor has their own and therefore you depend on that vendor to detect and deal with any malware on the machine based on their detection database. And given that you'll be doing this only if they've failed, that might not help. If you're unable to boot the suspect machine, this will at least let you gain access to it.

In my situation, I use a custom build of our own product on a bootable USB stick, the KNOS Secure Desktop. On it I have all of the necessary tools, including antivirus, network and malware scanners as well as other tools that will allow me to investigate an infected Windows machine using KNOS itself in GUI mode. It even permits me to locate and examine super-hidden items such as the SONY Rootkit I mentioned earlier without difficulty. Some Linux live cd distributions will also suffice for accessing Windows machines although they have limits if there's no other means available to gain access to the infected machine.

Having the tools you need beforehand though cannot be emphasized strongly enough. Locating and defeating undetected malware is a formidable challenge, and I hope this lengthy dissertation has been helpful in proceeding with the task successfully. Good hunting!

Author bio

Kevin McAleavey is currently the co-founder of The KNOS Project, located in Voorheesville, New York (US) and serves as its chief architect. The KNOS Project manufactures a malware-resistant, secure desktop operating system based on BSD known as the "KNOS Secure Desktop" for use by client end users which delivers a familiar, user-friendly and functionally complete desktop environment that can be run from a DVD, USB memory sticks or installed onto a hard drive and ensures privacy and security in its operation. Since it doesn't use the host computer's hard drive, it can be run separately from the existing operating system with no risk or damage to the original desktop's contents, nor will it perform any writes to its host system.



The KNOS Project

The unique design of KNOS provides a secure "lockbox" which prevents tampering or abuse of any of its components and operates completely in memory so as to leave no forensically traceable contents when run on a computer. KNOS is delivered on read-only media with the ability to retain that full level of security when installed onto writable media. It is designed for full time desktop use on military, corporate and individual computers in order to properly secure client desktops with guaranteed "lockdown." It can also serve as a complete replacement of the existing Windows, Linux or OSX operating system already installed on the existing computer. We also work with OEM's if a factory install is desired on new equipment.

The KNOS Project also offers a prefabricated, fully complete desktop version with a complete suite of applications known as our "KNOS Secure Desktop, version 9" for use by the general public and is available for purchase directly from our site in both 32 bit and 64 bit versions. We also have an "Internet LITE" version which provides only internet surfing capabilities without the full complement of end user applications provided with our full version. The KNOS Project also provides a specialized version specifically for forensics use with a complete complement of investigative tools plus the capability of examining original evidence without any risk of alteration in order to quickly determine if further forensic investigation of a machine is warranted.

The KNOS Project's primary mission however is to deliver highly customized, specifically built versions of the KNOS system and user environment to exact specifications and requirements and can also provide a custom toolkit which permits external institutions the ability to develop their own secure desktop operating environment with our tools under their complete control, construction, and distribution independently. Our construction toolkit permits institutions to fully acquire and control the sources of all components which will result in the finished product under their complete control if preferred.

Prior to creating the KNOS Project, Kevin spent his career in the privacy and antimalware industry developing the first anti-cookie privacy software in 1997 known as NSClean and IEClean, with the first antitrojan software known as BOClean released in 1998. Prior to that, Kevin worked for the State of New York as an electronics engineer and network administrator for several state agencies. Kevin's experience in malware research and countermeasures goes back to 1985 and he remains fully active and current in antimalware research in order to ensure that the KNOS product cannot be impacted by viruses and other malware.

Kevin has also written numerous white papers and articles on computer security over many years and still serves as an expert technical resource in recent articles in computer security and other technical publications. He has also provided extensive technical consultatation to numerous government, military and law enforcement entities over the years, as well as assisting in investigations and prosecution as an expert witness. He continues to do so under the auspices of The KNOS Project.

STATIC MALWARE ANALYSIS

THE FIRST STEP IN MALWARE ANALYSIS

by Ram Shmider

When you start your journey into malware analysis you need to remember that the files or machine you are working on are infected with real live malware.

What you will learn:

- the beginning steps that a malware analyst will do when researching the infected environment
- tools used for investigation of an infected environment
- how to extract information from the suspected files
- what you need to set up your own isolated lab for more tests.
- how to get information as file type, resources information etc. ge from a windows base PE file format

What you should know:

- Know what computer virus is.
- Be familiar with the Windows operation system and its architecture.
- Be familiar with the windows portable executable (PE) file format.
- Some programming experience can be helpful (C++ WIN32 API).

Due to this, you need to remember that you do not know what this malware will do if you accidently run it. It might spread itself into other computers in your network or try to use your e-mail account to send an e-mail to all of your contacts. Or, it might delete files from your computer and steal important information that you have on your computer, or use a stealth technology to hide itself or make use of some zero base exploit to attack other computers over the web. Also, it can turn the station into a zombie that at the right time will be part of a global cyber attack on some website. The limits are only the imagination of the Malware author.

Your firewall, your antivirus and other anti-malware tools might not be able to detect and stop this malware. The best thing to do, if you are going to "play" with malware is to create a virtual system, without network connectivity. Then, on that system you can start your analysis of the malware without fear that it will harm your computer. In using this environment you can use the tools that you need to run the malware, in order to understand how it works and what it does.

A well known environment that you can set up is called a "Virtual Environment" by using virtual software like; VMware, Virtual PC, or a service based sandbox, or you can set up an isolated environment that replicates the real environment that the malware was found on.

Malware analysis can be split into two methods; static analysis and dynamic analysis. In Static analysis you should not run the malware, but examine it using several tools to extract

as much information from the malware as possible, with this we will be able to get some useful information about the malware and continue our research using dynamic analysis.

We will run some tools against the malware file, to get information from the files, including:

- The file type
- Does the file is packed by some kind of file packers
- List of import/export function that it uses
- External files that it use
- Strings from the file
- Resources information

In dynamic analysis we will prepare the virtual environment, with some debugging and monitoring tools, and run the file using the debugger and/or the monitoring tools to get as much information as we can to understand how the malware is working. This method requires that you are familiar with; X86 assembly language, Reverse Engineering how the operating system operates in user mode and kernel mode, memory management and more. Most of the debuggers show you information in readable machine code, and by using these tools you will actually be able to see what the malware is doing in the system, and with that information you will be able to create a method to identify and remove the malware.

To identify malware, the identification will be the hash signature from the malware files. Removing malware after investigating the location the malware used to save files on the system, and the registry keys that it changed including, investigating some of the algorithm's it used, will enable you to create some kind of a pattern that recognizes the malware and allows you to remove it from the system so that the malware is marked as removed.

In the rest of this guide I will show you the steps that you need to do in order to do a static malware analysis, using a real malware file.

If you use a computer, chances are you will probably encounter some type of virus, and use an antivirus to find it, to give you information about how to remove it from your computer. So where can you find real malware to do testing?

Do you remember the e-mails you used to get that contained an attachment in it saying your antivirus added a message, and the file was suspected of having a virus and therefore removed?, this can be a good source to test malware on. You might also find malware on USB key's, malware loves to put a file on a USB drive. Try to scan it with your antivirus, but remember not to delete the file. In your antivirus configuration, you need to disable any real-time monitoring that you are doing so you are able to copy or move the file, and leave no access to the file, because you will not be given the option to copy, or move the file.

There is a rule that you need to know, never run the file, you should rename your malware file (re: <name>.ex_) so it will not run automatically if you accidently double click on the file.

Let's not forget the web; the web is full of viruses that you can download directly from some sites or by using some P2P software. There are torrent files that contain about six thousand viruses for testing that you can search for and download, but also keep in mind that there are some very old DoS virus's that still exist.

TESTING ENVIRONMENT

In order to check a malware file, or files, use an environment that is isolated, it can represent a copy of the real environment, including services and other applications. But this does not include a real network connection, or at least the option to connect the inner network to the internet.

If you are only checking malware, and you do not need a full environment, you can easily set up a Virtual Machine with your desired operating system and do your checking on that environment. This is a good learning process for offline checking, and it is much cheaper because it does not depend on any third party services, although you do need an application to run a virtual environment.

You can also create a "replica" of an isolated environment that represents the real environment that the malware was found on, which is also known as Sandbox. By using this you can try to trace the things that the malware did, or that it is really doing by isolating the running programs without the damage.

TIP

If you need to run a static analysis on a Windows based malware, you might also considering using a Linux environment. You can use Linux environment with tools that can check windows base PE Executable files, this method might be safer since the malware was only designed to run on a Windows operating system.

DOCUMENTING

It is highly recommended that you document each step you are doing on a live infected system. It can, and will give you more information about the malware, you need to get the location of the malware on the system "path", ie. running process, service and drivers, logon users, open ports open network sessions, system logs information, browser history, event viewer, and any other useful information that can be recorded will be valuable for the rest of the analysis.

STEP 1 – SCAN THE FILE WITH ANTIVIRUS PROGRAM

To check if the file is a known malware, and to get more information about it and know who to remove it, you can scan the file using your local antivirus. Keep in mind that verification is recommended so that your local antivirus is up to date.

There are lots of online antiviruses that you can use to scan your file, and there is a service that you can use that is called Virus Total (*https://www.virustotal.com/*), that uses a list of antiviruses to scan your file and give you a report about the file that you just scanned.

STEP 2 – CREATE A UNIQUE FILE SIGNATURE

The next step, in case that the antivirus does not recognize the file as a virus, is to create a unique md5 signature to check against malware MD5 databases. In case we do not find any recognized signatures on the sites, we might upload the file to one or more of the sites, and this gives other researchers the option to investigate the file. (Figure 1).

List of MD5 hash signature sites:

- *http://www.malwarehash.com/*
- *http://hash.cymru.com/*
- *https://www.virustotal.com/#search*
- *http://www.threatexpert.com/*
- *http://virusscan.jotti.org/hashsearch.php*
- *http://fileadvisor.bit9.com/services/search.aspx*
- *http://www.xandora.net/xangui/*
- *http://malwr.com/*
- *http://www.malwarepatrol.net*
- *http://vxvault.siri-urz.net/ViriList.php*
- *http://minotauranalysis.com/exetweet/*
- *http://www.nictasoft.com/ace/search-by-md5/*
- *http://malware.lu*
- *http://sarvam.ece.ucsb.edu/*
- *http://securitybot.org/md5-hash-db.php*

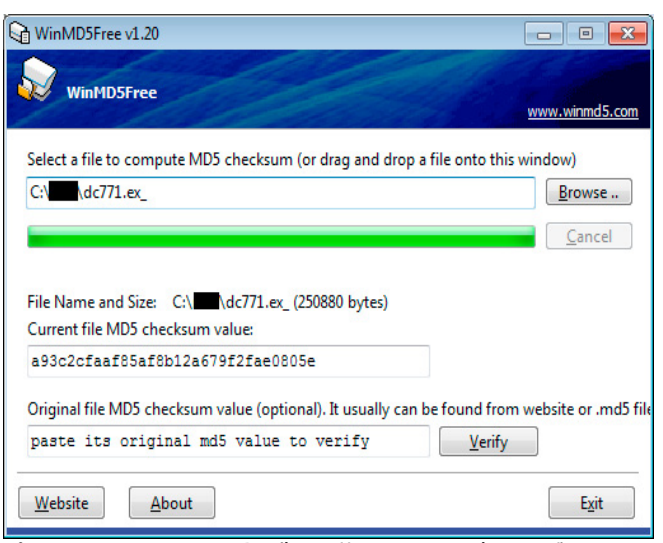


Figure 1. *Using WinMD5 (http://www.winmd5.com/) to get MD5 signature from the file*

STEP 3 – USE A HEXEDIT TO TRY AND CHECK THE FILE TYPE

You can download a free Hex Editor like hex edit from *http://www.hexedit.com/* and open the file. In this step we will try to see what the file type is, although most of the malware are in executable format some of them might contain more than one file. Using this tool we can read the file in hex and see what type it is, we can also read other section of the file to search for readable text, even though we will run a tool to extract text from the file.

SOME KNOWN FILE TYPE SIGNATURE

- MZ. – Windows executable format (EXE, DLL, SYS).
- Rar! – Rar file format (WinRAR).
- PK – Zip file format (WinZip).
- RIFF – AVI Movie file format.
- %PDF – PDF file format.
- ID3 – MP3 audio file.
- Any script file can be read as text files (html, js, vbs, php, pl, py…).

More file types can be found using web search.

STEP 4 – CHECK IF THE FILE IS A PACKED FILE

One well know tool that can be used to check if an exe. file is packed is the PEiD tool, it is a free tool that can be download from the web, although the development of the tool had been stopped, it can be used to get a list of well-known file executable packers.

If you find that the file is an executable packer and you find what the tool was that used to pack the file, you can go to the pack website, download the tool and then try to extract your malware from the executable packer. After the file is unpacked you can get more information using the tools in this document. (Figure 2).

LIST OF WELL KNOW EXECUTABLE PACKER

- UPX
- ASPack

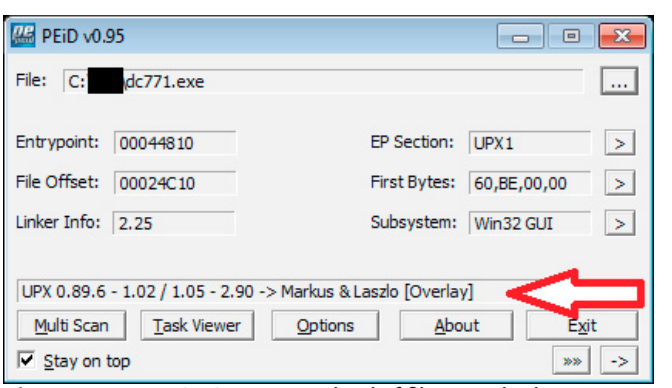


Figure 2. *Using PEiD v0.95 to check if file is packed*

- MPRESS
- VMProtect
- Themida

STEP 5 – GET HEADER INFORMATION FROM THE FILE

All executable files contain a header, and in this header you can get information about the file. Using the information from the PE file header, you can extract some useful information about the file, get some string and known section, and then you can get import and export function names and the files that will use the DLL list. You can get resource information, if the file contains one:

The *Portable Executable* (PE) contains the following thing:

- DoS Header.
- File Header.
- Optional Header.
- Section.
- Imports & Exports.
- Resources
- Extra Data.

Some known PE section names:

- .text – This section contain the executable code.
- .rdata – This section contain Read Only data.
- .data – This section contain global data.
- .idata – This section contain Import function data.
- .edata – This section contain Export function data.
- .rsrc – This section contain resource data (Figure 3 and Figure 4).

There are a lot more tools that can be used to get information from a PE file, you can find them using search sites over the internet, some of the tools are free and some cost money.

STEP 6 – GET EXPORT/IMPORT FUNCTIONS FROM THE FILE HEADER

With information about the exe Import and Export function and the DLL that it use we can get some more information on the file.

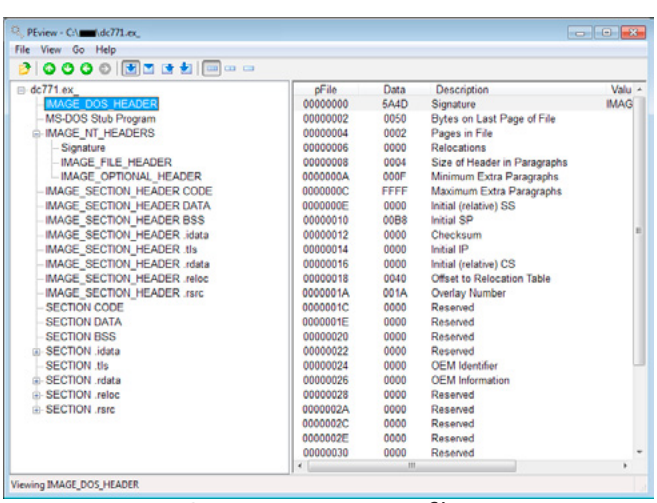
If there are not so many exported or imported functions and the one you can see are from this list there is a big severity that the file is using an executable packer.

- GetProcAddress
- LoadLibrary
- LoadLibraryEx
- VirtualAlloc
- VirtualAllocEX
- GlobalAlloc

You can look on the imported DLL to get more information about what type of malware is using.

This DLL list will give you some more information about it:

- ws2_32.dll – Use for windows socket programming.
- wininet.dll – Use for internet protocols like FTP and HTTP.
- Advapi32.dll – Use for cryptography, but not only.
- kernel32.dll – Use for kernel function, can be used in rootkits (Figure 5).



Fugure 3. *Using PEview to view an exe. file*

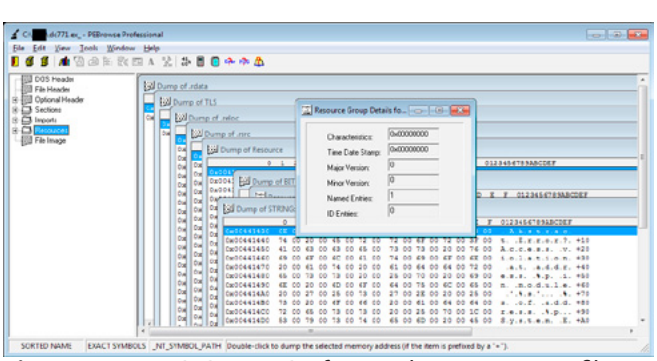


Figure 4. *Using PEBrowse Professional to view an exe. file*

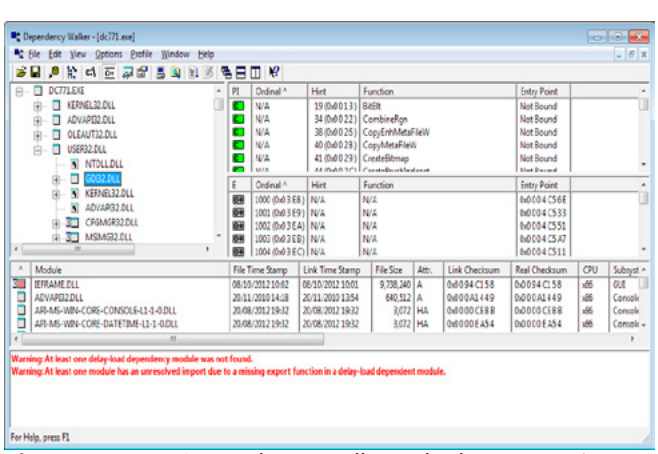


Figure 5. *Using Dependency Walker to look at import/export DLL & function*

STEP 7 – GET STRING FROM THE FILE

Using a string tool to extract to be able to read the text from the file can be good for getting more information, especially for error messages and other text that can be used to understand what the file is. You can sometimes get messages that the developer has left in the file, like remarks, you can get function names, and other text that might give you very important information. There are also a lot of strings that do not have any meaning and you need to ignore them. (See Figure 6).

In the above example you can see some strings that include; UPX!, Some DLL names and some function names. The TObjectd3 might indicate that the file was built using the Delphi programming language.

STEP 8 – CHECK FOR ADS

Alternate data streams (ADS) is a method that enables you to add "extra"data to any file or folder on your system. It is supported only on a NTFS file system, and might be used by malware to so call hidden information "behind" files.

In current Windows Based Operating systems the dir. command gets a new parameter that will allow you to list the files and the ADS, if any.

```
MZP
This program must be run under Win32
3.07
UPX!
StringX
Y6MD8
TObjectd3
...
KERNEL32.DLL
advapi32.dll
oleaut32.dll
user32.dll
LoadLibraryA
GetProcAddress
VirtualProtect
VirtualAlloc
VirtualFree
ExitProcess
RegCloseKey
VariantCopy
CharNextA
...
```

Figure 6. *Using strings.exe to get readable string from the exe file*

```
cmdline
dc771
dc771.exe
dc771.exe.html
dc771.exe.txt
dc771.ex_ ( 14 logger.txt, $DATA )
dc771.ex_.txt
dc771.txt
```

Figure 7. *Using the adsinfo.exe tool*

```
05/02/2012  11:59 PM    <DIR>          cmdline
11/19/2012  01:59 AM    <DIR>          dc771
11/19/2012  01:58 AM           169,472 dc771.exe
11/19/2012  02:00 AM            18,002 dc771.exe.html
11/25/2012  02:01 PM            26,416 dc771.exe.txt
11/25/2012  03:13 PM                 0 dc771.ex_
                                    14 dc771.ex_:logger.txt:$DATA
11/25/2012  02:00 PM               118 dc771.ex_.txt
11/20/2012  12:05 AM            62,919 dc771.txt
```

Figure 8. *Using the DIR /R command line on windows 7 (does not exist in XP)*

I can put any data that I want as a stream, to any file or folder that there is in the system, to a file system in default that will not show you nor report you about ADS, malware can use this as another method to hide data in the system, keep in mind that some files had an ADS in them with information related to the file and that information is legal. (Figure 7 and Figure 8).

STEP 9 – GET RESOURCE INFORMATION

Resource are images, icons, dialogs, version, string etc that can be attached to a file, using tools to extract this data you can get very nice information on file, it can be from images, dialogs and strings that might be in the file and in dialogs that the file use.

Keep in mind that not all the malware include dialogs or images in them (Figure 9).

STEP 10 – GET METADATA INFORMATION

Metadata data might be a great asset in malware analysis, when you get information from malware files about its metadata you can get the time stamp information, when it was created, modified and access, using this information you can find out when the file was created on this system or last time it was updated, and check in the logs in that timestamp might give you more information about what happened. You can also check for file properties who know you might find something useful in the file properties also.

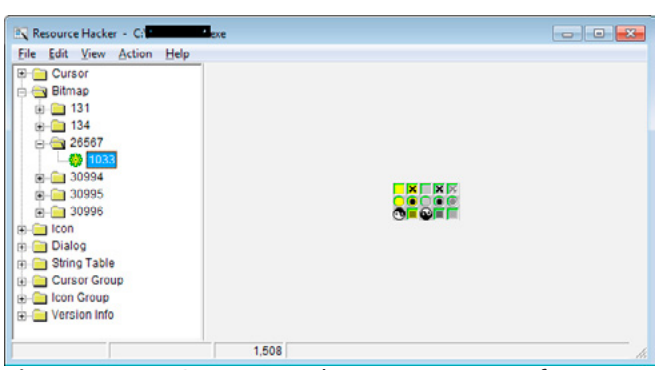


Figure 9. *Using Resource Hacker to get resource information*

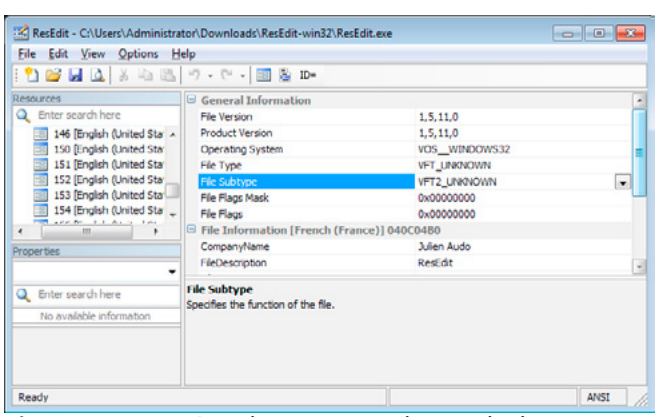


Figure 10. *Using ResEdit to get metadata and other information*

You can also get icons, dialogs, string table, images and other resources from the file, and some of them might have some value information that give you more about the file/malware.

Other files type have some more information in there metadata, in office document, PDF files, image files, movies files and other files type you might get very useful information from its metadata (Figure 10).

THE NEXT STEPS

With Static malware analysis, you can safely gather all kind of information from a suspected file that can give you basic information about the file or files that malware uses.

Static malware analysis is the first step that you can perform on any file before digging into dynamic analysis, which will involve a much deeper investigation, and knowledge due to the fact that you will actually run the malware to get into it to try to figure it out, what it does, and how it does it.

You saw here several methods to perform on malware file to get as much information as we can from the malware files, keep in mind that malware writes using the lastest method to manipulate many of today tools, and by that, more accomplished and sophisticated malware is getting harder to extract information from.

With the information you collect using static malware analysis, you are ready for the next step in dynamic malware analysis.

Author bio

Ram Shmider is ESM developer and integration, Professional developer, Security Hobbies. He has been in computer for more than 20 years, from the happy days of the DOS operation system, used to run my own BBS back there, I am a professional developer with experience in windows internal, device driver and web development working on my new security site – http://securitybot.org.

WIN32 ENVIRONMENT FOR MALWARE RESEARCH ON A LINUX MACHINE

by Alexey Lyashko

Malware analysis is a complex discipline, which includes many aspects, such as, research of malicious scripts or analysis of malicious executable binaries. The later, may require that a piece of software being dissected be run in a debugger or any form of a sandbox. Although, it is definitely possible to setup such a sandbox environment on a Windows machine, it is much safer to do so on an isolated, not Windows compatible operating system.

What you will learn:

- How processes are created and managed in Linux and Windows operating systems.
- Extend your knowledge of memory management mechanisms of both platforms.
- Gain more experience in system development under Linux in C and Assembly.
- Extend your knowledge of the PE file format.

What you should know:

- Knowledge of the C programming language.
- Prior experience of software development under Linux.
- Knowledge of Windows internal mechanisms (process and memory management).
- Understanding of Linux internal mechanisms in order to efficiently use them for Win32 emulation.

Despite all your efforts to isolate certain malware from the real environment, there is always a possibility (primarily, due to Windows' complexity) that it would break out, thus, ruining all your arrangements. In such a case, using Linux, which is known for its stability, unless you try to alter too many things in its kernel, low cost (well, it is simply free) and simplicity (those claiming otherwise may lack sufficient experience with the system), may be a perfect choice at least, so it was in my experience.

Such mechanisms guarantee that a software with a decryption mechanism you want to use is in order to decrypt itself, would not break out of your sandbox and do any harm to your arrangement, unless your choice is to use Virtual Machines, therefore significantly speeding up the process.

IMPLEMENTATION

One may argue about the need for such a setup for a very long time as it definitely has its pros and cons. Let us omit this from the article and concentrate on the implementation assuming that you are interested in this specific setup. This article covers a basic system sufficient for malware's self-decryption only. Of course, you are free to alter it according to your needs.

WHAT'S WRONG WITH WINE?

There is definitely nothing wrong with Wine. Simply – it is not an emulator, but a compatibility layer, which makes it possible to run Windows programs on several POSIX-compliant operating systems (read more at *http://www.winehq.org/about*). Given that its source code is freely available, and you have all the possibilities to alter it according to your needs

– it may be a good choice as well. However, it is still complicated as it emulates (at least intends to emulate) the whole operating system upon another operating system. Even if you want to log a couple of API calls, this may result in a tone of needless information, simply because of the way Wine loads an executable.

LOADER

Loader is the base of all. It is the Alpha and Omega of the proposed setup (well, Windows begins with a loader as well). It is responsible for the initiation of the Win32 process on our Linux machine, its maintenance and termination.

The first thing we have to consider while implementing a loader is the amount of sequential memory we want to leave for the Win32 code, keeping in mind the possible need to load additional libraries unless you are willing to deal with relocations (see *http://en.wikipedia.org/wiki/Portable_Executable* – Relocations). The easiest way would be to locate our loader as high as possible. While you can specify a custom image base address for a Windows executable, it will not necessarily run on Windows, on Linux the situation is different – you may safely tell the linker that you want your executable to be located at, for example, `0x90000000` and be sure that it will load to that exactly address. Simply pass the following parameter to the linker in the gcc command line:

```
-Wl,-Ttext,0x90000000
```

This will provide us with enough sequential virtual space for almost any need.

LOADING PE EXECUTABLE

The structure of a PE executable is beyond the scope of this article. Detailed description may be found on the MSDN website. However, let us review the loading process briefly.

First of all, allocate a buffer for file's headers, read them in and check for PE signature. Its file offset is in a word at offset `0x3C` in the DOS header. If it points to this sequence of bytes 'P','E','\0','\0', then you are trying to load a PE file, otherwise, this setup is useless.

Once you get the value of the image base for the victim executable, you should allocate a page of memory at that address. Do not use `malloc()` function, instead, you should operate with `mmap()` and `munmap()` unless you are implementing real `malloc()`/`free()` functions for your setup.

```
void* mmap(void *addr, size_t length, int prot,
                  int flags, int fd, off_t offset);
```

`addr` – desired base for mapped region (for example, `0x00400000` for standard PE file's first page) or NULL if you do not care where to map a page;
`length`: the length of the mapping. In our case it is the size of the page, which is usually 0x1000 but may vary;
`prot`: desired memory protection for the allocated region (see `mmap()` man page);
`flags`: you would, most likely, use `MAP_PRIVATE | MAP_32BIT`.

```
int munmap(void *addr, size_t length);
```

`addr`: start address of the region to unmap;
`length`: length of the memory to be unmapped.

Start copying headers to the newly allocated space.

Needless to mention, that you would have to map additional pages along parsing the headers and continuing to load the victim executable. All the section sizes and offsets are aligned (well, they should be) on a page boundary, so it would be quite easy to map the whole executable into memory. In case any offset or size is not aligned, it is your choice whether to terminate the process and report or to make it aligned.

LINKING PE EXECUTABLE

The next step, once your victim is fully loaded and correctly mapped, would be to link it against DLLs. In this particular case, we do not need any DLLs as we only need the executable to decrypt itself (unless we are working with any advanced protector like Themida, for example). In either way, linking is telling the executable where all the requested API functions or any other functions exported by other modules are located in memory. You will need to parse the PE import table for that and place correct addresses there. The description of the import table may be found in the same document, which describes the PE format.

However, our example does not require any additional library. The opposite – we are interested in stopping victim's execution as soon as it is decrypted. The best way to indicate that is to detect an API call. We would need to implement a single dummy API function in the loader, which would be capable of dumping the decrypted memory and correctly terminating the process. You would, most likely, redirect all victims' API calls to this dummy function.

STARTING AND TERMINATING THE VICTIM EXECUTABLE

Once your victim is loaded, mapped and linked, it is a perfect time to pass the execution flow to its entry point. The worst that may happen in case you did something wrong, is that you will, most likely get a segmentation fault or any other error message and the program will exit.

You have to define a function pointer in order to call victim's entry point. That would, most likely be:

```
typedef int (*victim_start_t)(void);
```

and then:

BIBLIOGRAFHY

- Dial M for malware – Virus Bulletin, August 2006. In co-authorship with Tomer Honen (currently Search Quality Training team lead at Google).
- A Simple Virtual Machine – I-Programmer (*http://i-programmer.info*), February 1st, 2012
- Executable Code Injection in Linux – I-Programmer (*http://i-programmer.info*), March 26th 2012
- Trivial Artificial Neural Network in Assembly Language – iMasters (*http://imasters.com.br*), April 2012 (selected as best of 2012 at iMasters)
- Linux Threads Through a Magnifier: Local Threads – iMasters (*http://imasters.com.br*), June 2012 (selected as best of 2012 at iMasters)
- Linux Threads Through a Magnifier: Remote Threads – iMasters (*http://imasters.com.br*), July 2012
- Hijack Linux System Calls:Part I – III – iMasters (*http://imasters.com.br*), July 2012
- Simple Virtual Machine – iMasters (*http://imasters.com.br*), July 2012
- Simple Runtime Framework by Example – iMasters (*http://imasters.com.br*), July 2012
- Passing Events to a Virtual Machine – iMasters (*http://imasters.com.br*), August 2012
- Advanced DLL Injection – iMasters (*http://imasters.com.br*), August 2012
- Executable Code Injection the Interesting Way – iMasters (*http://imasters.com.br*), August 2012
- Faking KERNEL32.DLL – an Amature Sandbox – iMasters (*http://imasters.com.br*), August 2012
- Dynamic Code Encryption as an Anti Dump and Anti Reverse Engineering Measure – iMasters (*http://imasters.com.br*), September 2012
- CreateRemoteThread. Bypass Windows 7 Session Separation – iMasters (*http://imasters.com.br*), October 2012
- Time Series Analysis and Forecasting. Programming Approach – thoughts – iMasters (*http://imasters.com.br*), October 2012
- Emulation of Hardware. CPU & Memory – iMasters (*http://imasters.com.br*), Devember 2012
- My official blog *http://syprog.blogspot.com* contains English versions of the articles from iMasters as well as other posts.


```
victim_start_t  start_func = (victim_start_t)
                    victimImageBase;
```

In case, the victim uses `ExitProcess()` or similar call in order to terminate – the control would be passed to the dummy API we've mentioned above and it would have to handle the correct process termination.

EXPANDING THE PROJECT

The tiny project, as described above, is quite simple to implement, but it is still almost of no use, unless you are planning to simply dump decrypted malicious code. However, I am almost certain, that you would want to get more out of this venture.

USING ADDITIONAL DLLS

You will most likely want to add support for additional DLLs in order to extend the functionality of your Windows emulator and support a few (or more then a few) Windows API functions. Nothing is easier (although, it is a bit annoying). You may create your own replicas of original Windows DLL files. Of course, you do not need to implement each and every API function – just those you really need (e.g. ExitProcess or LoadLibraryA/W and GetProcAddress). In this case, you will have to map them as well during the linking procedure of your loader.

FILE SYSTEM EMULATION

You may also want to emulate the file system tree in order to see how the victim malware manipulates files. This involves a bit more of a pain in the neck, as you would have to implement path conversion routines in addition to implementation of FS IO functionality.

CONCLUSION

Despite all the incompatibilities between Windows and Linux operating systems, this exact fact may play on our side when it comes to malware analysis as the same incompatibility, when appropriately approached, may let you secure your research environment and get better results with a framework based on the above concepts.

Author bio

Alexey Lyashko was born on January 16th 1979 in a small town of Gorlovka, Donetsk district, Ukraine (it was then the Soviet Socialist Republic of Ukraine as it was a part of the USSR). Nothing extraordinary happened till the age of 14, when his father bought him first computer – ZX Spectrum 48K. It was hard to get any books on programming at that time at the place where Alexey lived, so he learnt programming by investigating programs written by others. In February 1995, hemoved to Israel under the youth repatriation program Naale-16. The rest of his family moved in summer later that year. In July 1997, Alexey enlisted in the Israeli Defense Force and served in the 77th battalion of the famous 7th tank brigade as a tank driver. Demobilized in August 2000. 2000 – 2003 – studied at the National School of Technology in the Technion (the polytechnic university in Haifa). Software engineering faculty. 2003 – 2005 – worked as a technical support engineer at NetVision (largest ISP in Israel at that time) 2005 – officially started the career of a reverse engineer and malware researcher by joining the Content Security Response Team at Aladdin Knowledge Systems eSafe business unit. 2008 – 2011 – Worked as a consultant with Prevx ltd. Consulting on malware trends research and development of counter measures until the company has been acquired by Webroot software. 2011 – 2012 – Reverse Engineering and Anti Piracy consultant at Irdeto. Worked closely with the BD+ team on the research of piracy trends, analysis of BD ripping software and development of counter measures. October 2012 – current time – free for hire.

Audiobook

PRODUCTION

WANTED: 100 Forensics Authors!

You're missing out on the exploding Audiobook industry and massive amounts of unclaimed revenue!

Get your book professionally narrated and produced into an Audiobook and sell on Audibles.com, iTunes and Amazon.

Watch your credibility, your fame and your profits skyrocket...

FACT:
More people are listening instead of reading because it's more convenient and easier than traditional books.

So get with the program!

Visit www.legacy-strategic-development.com

REVERSE ENGINEERING LARGE JAVA PROGRAMS

AND UNDERSTANDING APPLICATION SERVERS

by Colin Renouf

The aim of this pair of articles is to convey the techniques and tools of the trade for understanding and reverse engineering large Java applications, and using JavaEE application servers as an example to understand how external interfaces and hosted JavaEE programs interact. This is a complex subject, so only the basics of application servers will be covered, but if there is more interest in the internals further articles can be produced.

What you will learn:

- How Java works and is put to use in the enterprise
- What the different tools are to reverse engineer a Java application and how to use them to reverse engineer a large application server and application environment
- In the second article we will apply these tools to some of the internals of the WebLogic Application Server, so you will also learn about the startup of this well known application server environment

What you should know:

- A little about how to start a Java application from the command line
- A little about programming in a language such as C, C#, or Java

A basic knowledge of how to start a Java program, set up its environment, and a small amount of coding knowledge in a language such as C++, or C# is all that is required.

In the first article we will cover the basics of the Java environment and the tools for reverse engineering, and in the second article we will apply them to reverse engineering the startup of the Oracle WebLogic Application Server – a well known JavaEE application server.

At this point you might be wondering why this is relevant in a forensics context. Well, as more of the modern server side systems move to Java and its enterprise services from C written proprietary systems, an understanding of how to reverse engineer server side Java programs will become essential. However, we will also examine in the next article how to use the tools to uncover what is going on inside an application server here because most make use of a number of key open source components that are by vendors, and many types of problems where Java forensics will be called for will involve the interaction between code written by criminal and the application server it is hooking, attacking, or using. Therefore, by using a complex example of a well known proprietary application server we can see not just how to uncover the workings of a large Java program, but also how to uncover the interactions with the platform on which it runs.

JAVA BASICS – HOW A JAVA PROCESS RUNS

In the modern world a large proportion of server side web and enterprise ap-

plications are based on the Java programming language and all or some subset of the JavaEE standard for enterprise Java application development.

Java is an object-oriented programming language developed by Sun Micro Systems (now part of Oracle) by simplifying C and C++ to remove the complex features that led to bugs, e.g. Pointers, multiple inheritance, etc. The aim was to make the language portable across multiple environments so the concept of a Java Virtual Machine runtime was used; where Java language is compiled down to produce Java "byte code" as equivalent to a machine language for the virtual machine. This byte code is translated by interpretation or compiled at the class level into the native machine language for the given platform.

In Java code is written as object oriented classes that derive from the parent java.lang.Object class, and the functions operate on that and other classes from within the classes "methods" – the equivalent of a C function. One class in an application that runs as a process contains a "static void `Main()`" method as the entry point; and for JavaEE this is usually the hosting application server.

The *Java virtual machine* (JVM) has a top half that processes Java byte code and looks to all intents and purposes to the byte code as if it is a real machine, and the bottom half translates this generic platform independent representation into something that is executable by the real machine on which it is running. Rather than having virtual registers like a real machine the JVM uses a stack based machine architecture, with position on the stack taking the place of defined registers.

When the Java virtual machine, which has its core usually written in C, executes it performs its own internal bootstrapping and then gets on to the business of running Java code, i.e. loading and executing Java classes and the methods within them. The classes are loaded by a hierarchy of classloaders; the first of which is the bootstrap classloader that is usually written in C to target the native platform, and which loads the Java code that is fundamental to execution of Java code and which interfaces directly to native or C code. This then leads to the loading and execution of other classloaders that are written in

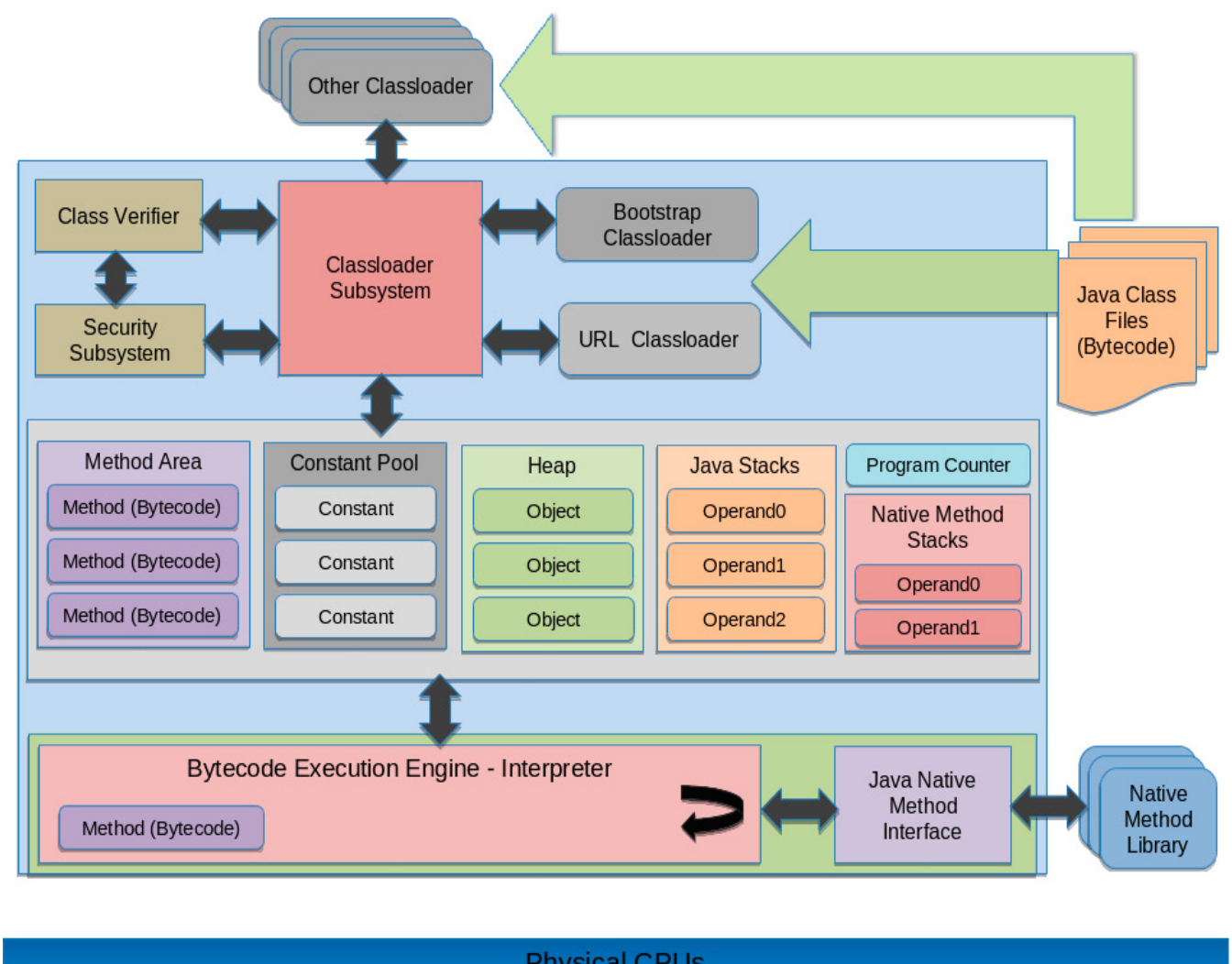


Figure 1. *Java Virtual Machine Architecture*

pure Java; the most fundamental of which is the `URLClassloader` from which many others inherit.

When the classes are loaded a class verifier checks the loaded Java byte code against a set of rules to provide the Java security and reliability features. After these checks the constants are loaded into the constant area and methods are loaded into the method area. Objects are created on the heap by the "new" operator, methods execute on the stack, with parameters and return values "pushed" onto the stack when a method call is made. Native code for the platform executes on its own stack to protect the Java code, as the native code can't be verified in the same way as the Java code can as it is dependent on the platform and what else is running on the real machine. Eventually a Java byte code interpreter executes the Java byte code instructions, or a *just in time* (JIT) compiler produces compiled native code "modules" to represent one or more classes at runtime for loading into a native code execution engine under the control of the virtual machines (Figure 1).

UNDERSTANDING JAVA PROCESSES ON THE PLATFORM

There are a few key areas that must be understood in order to understand how a typical Java process interacts with the platform on which it is running. We will look at these generically and then look at the specifics of understanding the code that starts up the Oracle WebLogic Server 11G application server.

ENVIRONMENT VARIABLES

With Java processes the environment variables used to control the virtual machine itself are critical to how the process runs, and often even which code runs.

Many years ago, whilst investigating performance and reliability issues between the IBM WebSphere Application Server, the AIX operating system, and the WebSphere MQ (formerly MQSeries) messaging engine it was necessary to reverse engineer the interfaces between each of these products to a very low level to determine how thread synchronization was working (or not in this case). The WebSphere Application Server package is IBMs JavaEE application server engine, and it runs on a Java virtual machine (an IBM one called J9).

During the reverse engineering work code seemed to execute that seemed unrelated to what appeared to be deployed; but we eventually found that multiple copies of some classes with the same name but a different implementation or at different versions were deployed in different Java archive library files (Jar files) and this was the cause of the confusion. This is seen often in the JavaEE world as many vendors take parts of Open Source reference packages for some parts of their standards compliant implementation and integrate it, but as these packages may themselves be based on smaller utility packages at different versions the Jar files would often package their own specific versions and thus duplicate the implementation. This leads to the reason why the CLASSPATH environment variable or Java virtual machine parameter (`-cp <>` or `-classpath <>`) is so important as it controls which library and which Java class is visibly in scope at a particular time.

THE COMMAND LINE

There are a number of parameters on the Java virtual machine command line that control the execution environment in terms of memory or can pass variables to the application.

- The `-D A=B` type of option allows the property A to be set to the value D.
- The `-XmxVALUE` and `-XmsVALUE` control the heap size available to the JVM to VALUE, and `-XssVALUE` sets the Java thread stack size to VALUE.
- The `-Xshare`: options control how classes are cached across Java process instances when loaded by the first instance.
- The `-verbose:X` options control the amount of information written to standard output, which is often used for log files.

Listing 1. *Java Code to Sleep*

```
class test {
    public static void main(String args[]) {
        System.out.println("About to sleep for a really long time");
        try {
            java.lang.Thread.sleep(1000000);
        } catch (Exception e) {
        }
    }
}
```

Output 1a. *Thread dump showing run time relationship between threads and classes*

```
2012-12-15 11:18:28
Full thread dump OpenJDK Server VM (20.0-b12 mixed mode):

"Low Memory Detector" daemon prio=10 tid=0xb77a3c00 nid=0x1119 runnable [0x00000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread1" daemon prio=10 tid=0xb77a2000 nid=0x1118 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread0" daemon prio=10 tid=0xb779fc00 nid=0x1117 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Signal Dispatcher" daemon prio=10 tid=0xb779e800 nid=0x1116 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Finalizer" daemon prio=10 tid=0xb778ec00 nid=0x1115 in Object.wait() [0x8be1b000]
   java.lang.Thread.State: WAITING (on object monitor)
   at java.lang.Object.wait(Native Method)
   - waiting on <0xa9ba1160> (a java.lang.ref.ReferenceQueue$Lock)
   at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:133)
   - locked <0xa9ba1160> (a java.lang.ref.ReferenceQueue$Lock)
   at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:149)
   at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:177)

"Reference Handler" daemon prio=10 tid=0xb778d400 nid=0x1114 in Object.wait() [0x8be6c000]
   java.lang.Thread.State: WAITING (on object monitor)
   at java.lang.Object.wait(Native Method)
   - waiting on <0xa9ba1060> (a java.lang.ref.Reference$Lock)
   at java.lang.Object.wait(Object.java:502)
   at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:133)
   - locked <0xa9ba1060> (a java.lang.ref.Reference$Lock)

"main" prio=10 tid=0xb7704c00 nid=0x1110 waiting on condition [0xb7871000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
   at java.lang.Thread.sleep(Native Method)
   at test.main(test.java:5)

"VM Thread" prio=10 tid=0xb7789800 nid=0x1113 runnable

"GC task thread#0 (ParallelGC)" prio=10 tid=0xb770c000 nid=0x1111 runnable

"GC task thread#1 (ParallelGC)" prio=10 tid=0xb770d400 nid=0x1112 runnable

"VM Periodic Task Thread" prio=10 tid=0xb77a5800 nid=0x111a waiting on condition

JNI global references: 863

Heap
PSYoungGen      total 8832K, used 304K [0xa9ba0000, 0xaa570000, 0xb4640000)
  eden space 7616K, 4% used [0xa9ba0000,0xa9bec318,0xaa310000)
  from space 1216K, 0% used [0xaa440000,0xaa440000,0xaa570000)
  to   space 1216K, 0% used [0xaa310000,0xaa310000,0xaa440000)
PSOldGen        total 20160K, used 0K [0x94640000, 0x959f0000, 0xa9ba0000)
  object space 20160K, 0% used [0x94640000,0x94640000,0x959f0000)
PSPermGen       total 16384K, used 1921K [0x8c640000, 0x8d640000, 0x94640000)
```

Output 1b. *Thread dump showing run time relationship between threads and classes*

```
  object space 16384K, 11% used [0x8c640000,0x8c820670,0x8d640000)

2012-12-15 11:18:33
Full thread dump OpenJDK Server VM (20.0-b12 mixed mode):

"Low Memory Detector" daemon prio=10 tid=0xb77a3c00 nid=0x1119 runnable [0x00000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread1" daemon prio=10 tid=0xb77a2000 nid=0x1118 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"C2 CompilerThread0" daemon prio=10 tid=0xb779fc00 nid=0x1117 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Signal Dispatcher" daemon prio=10 tid=0xb779e800 nid=0x1116 waiting on condition [0x00000000]
   java.lang.Thread.State: RUNNABLE

"Finalizer" daemon prio=10 tid=0xb778ec00 nid=0x1115 in Object.wait() [0x8be1b000]
   java.lang.Thread.State: WAITING (on object monitor)
   at java.lang.Object.wait(Native Method)
   - waiting on <0xa9ba1160> (a java.lang.ref.ReferenceQueue$Lock)
   at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:133)
   - locked <0xa9ba1160> (a java.lang.ref.ReferenceQueue$Lock)
   at java.lang.ref.ReferenceQueue.remove(ReferenceQueue.java:149)
   at java.lang.ref.Finalizer$FinalizerThread.run(Finalizer.java:177)

"Reference Handler" daemon prio=10 tid=0xb778d400 nid=0x1114 in Object.wait() [0x8be6c000]
   java.lang.Thread.State: WAITING (on object monitor)
   at java.lang.Object.wait(Native Method)
   - waiting on <0xa9ba1060> (a java.lang.ref.Reference$Lock)
   at java.lang.Object.wait(Object.java:502)
   at java.lang.ref.Reference$ReferenceHandler.run(Reference.java:133)
   - locked <0xa9ba1060> (a java.lang.ref.Reference$Lock)

"main" prio=10 tid=0xb7704c00 nid=0x1110 waiting on condition [0xb7871000]
   java.lang.Thread.State: TIMED_WAITING (sleeping)
   at java.lang.Thread.sleep(Native Method)
   at test.main(test.java:5)

"VM Thread" prio=10 tid=0xb7789800 nid=0x1113 runnable
"GC task thread#0 (ParallelGC)" prio=10 tid=0xb770c000 nid=0x1111 runnable
"GC task thread#1 (ParallelGC)" prio=10 tid=0xb770d400 nid=0x1112 runnable
"VM Periodic Task Thread" prio=10 tid=0xb77a5800 nid=0x111a waiting on condition

JNI global references: 863

Heap
 PSYoungGen      total 8832K, used 304K [0xa9ba0000, 0xaa570000, 0xb4640000)
  eden space 7616K, 4% used [0xa9ba0000,0xa9bec318,0xaa310000)
  from space 1216K, 0% used [0xaa440000,0xaa440000,0xaa570000)
  to   space 1216K, 0% used [0xaa310000,0xaa310000,0xaa440000)
 PSOldGen        total 20160K, used 0K [0x94640000, 0x959f0000, 0xa9ba0000)
  object space 20160K, 0% used [0x94640000,0x94640000,0x959f0000)
 PSPermGen       total 16384K, used 1921K [0x8c640000, 0x8d640000, 0x94640000)
  object space 16384K, 11% used [0x8c640000,0x8c820670,0x8d640000)
```

Listing 2a. *Java Code to Output Class Information using Reflection*

```
  import java.io.*;
import java.lang.reflect.*;
public class ShowClassFile {
    public static void main(String[] args) {

        if (args.length == 2) {
            BufferedReader reader;
            FileWriter writer;
            String testClassName;
            try {
                reader = new BufferedReader(new FileReader(new File(args[0])));
                writer = new FileWriter(args[1]);
                while ((testClassName = reader.readLine()) != null) {
                    try {
                        Class test = Class.forName(testClassName);
                        String location;
                        if (test.getClassLoader() != null) {
                            location = "Class loaded from: " +
                                test.getProtectionDomain().getCodeSource().getLocation();
                        } else {
                            location = "System Classloader loaded class from: " +
                                test.getClassLoader().getResource(testClassName);
                        }

                        // Write the details of the class out to a file
                        printClassInfo(test, location, writer);

                        location=null;
                        test=null;

                    } catch (ClassNotFoundException e) {
                        System.out.println("Couldn't find class: " + args[0]);
                    }
                    testClassName = null;
                    writer.flush();
                }
                reader.close();
                writer.close();
            } catch (FileNotFoundException e) {
                System.out.println("Input or output files not found");
            } catch (IOException e) {
                System.out.println("An IOException has occured writing to the file");
            }
        }
        else {
            System.out.println("Usage:ShowClassFile objectlistfile outputdetailsfile");
        }
    }

    public static void printClassInfo(Class clazz, String location,
                                      FileWriter writer) throws IOException {
        writer.write("\n\n**************************************************\n");
        writer.write("Class Name: " + clazz.getName() + "\n");
        writer.write(location + "\n");
        writer.write("Superclass:" + clazz.getSuperclass() + "\n");
```

These values can often be found in startup script, which for the WebLogic Server application server would be the `startManagedWebLogic.sh` script, and some values can be found in the XML configuration file for WebLogic; config.xml. These are specific to the WebLogic environment, although the general idea of the script and configuration XML file is generally common across application server environments.

For reverse engineering, once the environment variables are known – such as by copying the startup script and editing it to start a different process – the key to understanding a process is to use the -verbose:class setting on the command line. This verbose logging causes the class name and the file location it was loaded from to be written to the standard output log file for the Java process – it is from here that we can see, at least until multiple threads are in flight, which classes are likely to have loaded which other classes; thus giving the dependencies. With minimal effort this log file can be processed and used as input to a Java process for using reflection to get a look into the original code; and we will look at a sample Java application for doing this later on in this article.

THREAD DUMP

When a process is running we often need to know which threads are active, what their stacks look like in terms of which method has called which method, and what threads are blocked. To do this we need to take a thread dump. Assuming the Java process is started from a command prompt in Windows this so done by pressing the Ctrl-Break key combination, and in Unix/Linux based platforms this is done by using a kill -QUIT PID or kill -3 PID command to send the SIGQUIT signal to the process with the process ID PID.(See Listing 1).

As an example we have created a simple Java program that just sleeps for a very long time so we can examine what is going on. Then we used the kill -3 command and caused the thread dump information to be written to the standard output file. The state of the threads is given by the keywords RUNNABLE, WAITING. TIMED_WAITING, etc. The thread IDs are shown and indicators as to what is causing the threads to block. Note the compiler threads showing that JIT compilation is available, even for a simple Java class with nothing else to compile. The GC threads show that garbage collection is active to tidy up memory once the finalizer thread has freed the object references. There are also signal handlers and scheduler threads, even for the simplest of classes. Each thread shows the stack of calls that got it to its current position, and is read from the bottom up for that thread. The heap is split into objects that are transient and objects that are long lived, which makes garbage collection a lot easier to implement. (See output 1).

APPLICATION CONFIGURATION

The type of configuration information varies between applications. Originally, each JAR file had a manifest with information and JavaEE applications had such files as web.xml, ejb-jar.xml, etc. Then annotations, where control information is written into the class file itself appeared on the scene so less of the metadata is written to the files.

With the application servers and other supporting infrastructure files were in use such as config.xml or server.xml; but these vary with the type of environment in which the program runs. The only solution to understanding how these are structured is to examine the documentation for the given platform. However, do try to get an understanding of the architecture of the platform as

Listing 2b. *Java Code to Output Class Information using Reflection*

```
        writer.write("\nInterfaces:\n**********\n");
        writer.write("\nMethods:\n*********\n");
        Method[] methods = clazz.getMethods();
        for (int idx=0; idx < methods.length; idx++)
            writer.write(methods[idx] + "\n");
        writer.write("\nFields:\n*******\n");
        Field[] fields = clazz.getFields();
        for (int idx2=0; idx2 < fields.length; idx2++)
            writer.write(fields[idx2] + "\n");
        writer.write("\n*************************************\n\n\n");
        writer.flush();
        fields=null;
        methods=null;
    }
}
```

Output 2. *Class Information for the WebLogic.Server class*

```
**********************************************************
Class Name: weblogic.Server
Class loaded from: file:/home/colin/Oracle/MiddlewareJDev/wlserver_10.3/server/l
ib/weblogic.jar
Superclass:class java.lang.Object

Interfaces:
***********

Methods:
*********
public static void weblogic.Server.main(java.lang.String[])
public java.lang.String weblogic.Server.toString()
public static boolean weblogic.Server.isRedefineClassesSupported()
public static java.lang.String weblogic.Server.getUsage()
public final native void java.lang.Object.wait(long) throws java.lang.Interrupte
dException
public final void java.lang.Object.wait(long,int) throws java.lang.InterruptedEx
ception
public final void java.lang.Object.wait() throws java.lang.InterruptedException
public native int java.lang.Object.hashCode()
public final native java.lang.Class java.lang.Object.getClass()
public boolean java.lang.Object.equals(java.lang.Object)
public final native void java.lang.Object.notify()
public final native void java.lang.Object.notifyAll()

Fields:
*******
public static final java.lang.String weblogic.Server.WEBLOGIC_INSTRUMENTATION_PR
OPERTY
public static final java.lang.String weblogic.Server.DIAGNOSTIC_PRE_PROCESSOR_CL
ASS
public static final java.lang.String weblogic.Server.WEBLOGIC_INSTRUMENTATION_SE
RVER_SCOPE
public static final java.lang.String weblogic.Server.CLASSLOADER_PREPROCESSOR

************************************
```

Listing 3. *Disassembled Source for the Java Sleeping Code*

```
import java.io.PrintStream;

class test {
   test() { }

    public static void main(String args[]) {
        System.out.println("About to sleep for a really long time");
        try {
            Thread.sleep(0xf4240L);
        }
        catch(Exception exception) { }
    }
}
```

there are often hidden configuration files. For example, since many application servers and most Eclipse-based IDEs are based on the OSGi standard for Java components using a registry and version management for multi-versioning it is worth understanding the XML or manifest files used for this as well – investigation and understanding is key. There are internals books available that can open up this information to interested parties.

Often this is the most important approach as with careful configuration of debugging options and internal hidden configuration options can often result in debugging information and traces being written to an output file, and this can effectively give a source code level trace. Do be aware that this will be slow.

REFLECTION

Java has a programming interface to understand the environment, the methods, etc of referenced classes. This API is known as the Reflection API.

For our example here, the output of the verbose:class information generated when the WebLogic application server was started, was stripped using Linux “grep” and “cut” utilities to produce a list of class files loaded in the appropriate order from the correct JAR locations in the environment and a Java application that uses reflection to interrogate each class as it is loaded was produced. This is below and shows how reflection is used. The output was written to another “report” and this was scripted to use an Eclipse-based modeling framework to produce class diagrams and sequence diagrams to understand the relationships. (See Listing 2).

An example of the output is shown in following output – Output 2.

JAVA DECOMPILER (JAD)

The final tool in the toolbox of every Java reverse engineer is the Java Decompiler which generates candidate source code for a class, i.e. Java code that would map to the given byte code; but note that it may not be the actual source code itself.

There is some question as to the legality of this approach so only use it if the code is not commercial code. In reality, the information given by the above approaches is usually sufficient to be able to understand most programs.

Jad can be downloaded from the link: *http://www.varaneckas.com/jad/*.

If we use the executable to decompile our test.class file we can see that it isn't the original source code that is recovered but source code that is equivalent. The decompiler generates a .jad file by default, but this can be renamed to a .java file to support changing it and redeploying. Note that the decompiled code includes an empty constructor and a hex value in the sleep call; and that imports are named for the external interface calls. (See Listing 3).

SUMMARY

So, what have we achieved so far? We have looked at how a Java process executes on a Java virtual machine, and examined some of the tools for examining a complex Java package. In the next article we will apply these tools – using the startup of the WebLogic Server JavaEE application server from Oracle as an example.

Author bio

Colin Renouf is a long standing enterprise solutions architect with thirty years experience in the industry – concentrating on the finance sector. He has authored many magazine articles ranging from Unix, through Java and on to security; and has also written and contributed to books on the subject. He is currently contracting for a well known credit card company, but his main loves are Australia and some of its people, his children, singing, photography and just being with good company. Oh, and quantum physics as he is an eternal scientist.”

Make them hang on your every word...
Put them on the edge of their seats when you speak...

Leave them wanting more...
It can happen, but ONLY IF YOU GET THIS:

THE ELECTRONIC ADVANTAGE: 101 the Basics

This fast-paced, 4-hour, online tutorial is for any Skill level and even includes 10 case examples! All this for just $360

BONUS GIFT:
The first 50 orders get our incredible 92 Page Tech Guide, eBook, and Audiobook a $100 value

Go here now and order:
www.technologicalevidence.com

REVERSE ENGINEERING LARGE

JAVA PROGRAMS AND UNDERSTANDING APPLICATION SERVERS – PART TWO

by Colin Renouf

As stated in the previous article, the aim of this pair of articles is to convey the techniques and tools of the trade for understanding and reverse engineering large Java applications, and using JavaEE application servers as an example to understand how external interfaces and hosted JavaEE programs interact.

What you will learn:

- How Java works and is put to use in the enterprise
- What the different tools are to reverse engineer a Java application and how to use them to reverse engineer a large application server and application environment
- In the second article we will apply these tools to some of the internals of the WebLogic Application Server, so you will also learn about the startup of this well known application server environment

What you should know:

- A little about how to start a Java application from the command line
- A little about programming in a language such as C, C#, or Java

This is a complex subject, so only the basics of application servers will be covered, but if there is more interest in the internals further articles can be produced. A basic knowledge of how to start a Java program, set up its environment, and a small amount of coding knowledge in a language such as C++, or C# is all that is required. We assume that the basics of Java and the reverse engineering tooling as covered in the previous example are now understood.

In the first article we covered the basics of the Java environment and the tools for reverse engineering, and in this second article we will apply them to reverse engineering the startup of the Oracle WebLogic Application Server – a well known JavaEE application server. Most application servers follow a similar pattern, although only the Oracle WebLogic Server (known as WLS) has the "T3" subsystem at its core. Most these days, including newer versions of WLS, base their core and startup on the *Open Systems Gateway Interface* (OSGi) standard that allows multiple versions of the same library to be dynamically loaded and unloaded concurrently.

As suggested in the previous article, you might be wondering why this is relevant in a forensics context. Well, as more of the modern server side systems move to Java and its enterprise services from C written proprietary systems, an understanding of how to reverse engineer server side Java programs will become essential. However, we will examine here how to use the tools we looked at previously to uncover what is going on inside an applica-

tion server here because most make use of a number of key open source components that are by vendors, and many types of problems where Java forensics will be called for will involve the interaction between code written by criminal and the application server it is hooking, attacking, or using. Therefore, by using a complex example of a well known proprietary application server we can see not just how to uncover the workings of a large Java program, but also how to uncover the interactions with the platform on which it runs.

AN EXAMPLE – REVERSE ENGINEERING THE WEBLOGIC SERVER 11G STARTUP

Now its time to put all of the techniques and tools we covered in the previous article together to reverse engineer part of a complex package, in this case the WebLogic Server. Remember that these include the environment in which the process runs; code to use Java reflection to examine the interfaces for a class; logging from the Java virtual machine itself; and, for extreme cases only, the Java decompiler. We can use all of this tooling not just to understand the code at a source level, but also the structure of the program; which we can diagram and even automate the diagramming using tooling such as the *Eclipse Modeling Framework* (EMF) if desired.

We can understand a lot about the internal architecture of the WebLogic Server by the way it starts up, as initialization processes tend to take a "big picture" approach to starting up subsystems. This leaves us with a view as to the key subsystems within the application server on which other subsystems build.

On a Linux platform the startup is handled, by default, from the `/opt/Oracle/Middleware/`

Listing 1. *WebLogic stack trace for a failed startup*

```
<Jan 24, 2010 4:26:52 PM GMT> <Info> <WebLogicServer> <BEA-000377> <Starting WebLogic Server with
BEA JRockit(R) Version R27.6.2-20_o-108500-1.6.0_05-20090120-1115-linux-ia32 from BEA Systems,
Inc.>
<Jan 24, 2010 4:26:55 PM GMT> <Info> <Security> <BEA-090065> <Getting boot identity from user.>
Enter username to boot WebLogic server:weblogic
Enter password to boot WebLogic server:
<Jan 24, 2010 4:27:07 PM GMT> <Info> <Management> <BEA-141107> <Version: WebLogic Server 10.3.1.0
Thu Jun 11 00:26:56 EDT 2009 1227385 >
<Jan 24, 2010 4:27:07 PM GMT> <Critical> <WebLogicServer> <BEA-000362> <Server failed. Reason:

There are 1 nested errors:

weblogic.management.ManagementException: [Management:141247]The configuration directory /opt/
Oracle/Middleware/wlserver_10.3/common/bin/config does not exist and the admin server is not
available.
   at weblogic.management.provider.internal.RuntimeAccessImpl.parseNewStyleConfig(RuntimeAccessI
mpl.java:200)
   at weblogic.management.provider.internal.RuntimeAccessImpl.<init>(RuntimeAccessImpl.java:115)
   at weblogic.management.provider.internal.RuntimeAccessService.start(RuntimeAccessService.
java:41)
   at weblogic.t3.srvr.ServerServicesManager.startService(ServerServicesManager.java:461)
   at weblogic.t3.srvr.ServerServicesManager.startInStandbyState(ServerServicesManager.java:166)
   at weblogic.t3.srvr.T3Srvr.initializeStandby(T3Srvr.java:749)
   at weblogic.t3.srvr.T3Srvr.startup(T3Srvr.java:488)
   at weblogic.t3.srvr.T3Srvr.run(T3Srvr.java:446)
   at weblogic.Server.main(Server.java:67)

>
<Jan 24, 2010 4:27:07 PM GMT> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to
FAILED>
<Jan 24, 2010 4:27:07 PM GMT> <Error> <WebLogicServer> <BEA-000383> <A critical service failed. The
server will shut itself down>
<Jan 24, 2010 4:27:07 PM GMT> <Notice> <WebLogicServer> <BEA-000365> <Server state changed to
FORCE_SHUTTING_DOWN>
```

wlserver_10.3 directory. For the purposes of our investigation this is the runtime "root" directory of the Oracle WebLogic Server installation, so all further discussions will leave this out as a common starting point when referring to implementation code for Oracle WebLogic Server. Under this directory there are a number of subdirectories of interest.

```
common
inventory
server
uninstall
```

However, before concentrating on the server itself we must look at the WebLogic enterprise concepts of domains, clusters, instances and machines. A domain is a sphere of administrative control under which clusters of application server instances sit, which are assigned to one or more machines. We can cluster application server instances locally on a single machine for vertical clustering which offers multiple JVM processes to support an application for load balancing and scalability purposes and to support higher availability when a single JVM process fails, and clustering across multiple machines for horizontal clustering to give load balancing, scalability and high availability to support loss of a machine or operating system image. A single operating system image is referred to as a node and runs a Node Manager that enables stopping and starting of individual server instances, but when this is all configured the domain configuration created by config.sh (Unix/Linux) or config.cmd (Windows) refers to a MACHINE.

It is this environment that the Node Manager understands and the Admin Server controls, but the Node Manager does exist across domains in that if multiple domains are configured on a machine it can work with them all. Thus, to examine the startup we must have a properly configured local environment on our machine, with a domain, Admin Server, a cluster, and multiple server instances within that cluster all running on a configured machine. The domain configuration is held, by default, in a directory under /opt/Oracle/Middleware called user_projects/domains and within this a separate tree structure exists for each domain. The node manager requires this configuration to be in place in order to startup the instances. So, when referring to configuration it is this user_projects/domains directory structure that we will take as our configuration "root" in this chapter.

All of this information can be understood simply from installing the application server environment itself and using the help documentation for each step.

NODE MANAGER

Typically, in an enterprise environment the first thing to start would be the Node Manager, which controls the WLS processes on that operating system image and allows individual application server instances to be started and stopped, and this can be started using the startNodeManager.sh script under the server/bin subdirectory in our. Running this script results in the environment variables for WLS being set, and the weblogic.NodeManager class main() method entry point being called from the weblogic.jar file in the server/lib subdirectory.

The NodeManager class is a simple one that either displays help information, if insufficient parameter information is passed, or it calls the main method of the NMServer class in the nodemanager.server package in the weblogic.jar file and passes the parameters obtained on to it for processing. This class, in turn, reads the properties for the socket connectivity, domains, logging, etc and passes it to an NMServerConfig class from the same package for management, using classes from the weblogic.nodemanager.util package

Listing 2. *WebLogic Command Line*

```
 /opt/Oracle/Middleware/jdk160_14_R27.6.5-32/bin/java -client -Xms256m
-Xmx512m -XX:CompileThreshold=8000 -XX:PermSize=48m -XX:MaxPermSize=128m -Dweblogic.
Name=AdminServer -Djava.security.policy=/
home/colin/Oracle/Middleware/wlserver_10.3/server/lib/weblogic.policy -Xverify:none -da -Dplatform.
home=/opt/Oracle/Mi
ddleware/wlserver_10.3 -Dwls.home=/opt/Oracle/Middleware/wlserver_10.3/server -Dweblogic.home=/opt/
Oracle/Middl
eware/wlserver_10.3/server -Dweblogic.management.discover=true -Dwlw.iterativeDev= -Dwlw.
testConsole= -Dwlw.logErrorsToConsol
e= -Dweblogic.ext.dirs=/opt/Oracle/Middleware/patch_wls1032/profiles/default/sysext_manifest_
classpath:/opt/Ora cle/Middleware/patch_jdev1111/profiles/default/sysext_manifest_classpath
weblogic.Server
```

Technology is a double sided sword.
Internet makes you naked online!
Get Secured & Get Certified!

Welcome to the world of Certified Ethical Cracker
with Hands-on practical sessions.

CERTIFIED ETHICAL CRACKER

An Advance **Information Security** Course

For more detais, visit:
http://www.infysec.com/training/courses/certified-ethical-cracker

infySEC UK :

145-157, St.John Street,
London, EC1V 4PW
England, UK

Phone: +44-7405190001

infySEC India :

#37/45, P.H Road,
Arumbakkam,
Chennai- 600106
TamilNadu, INDIA

Phone: +91-44-42611142,43

infySEC
Demystifying Innovations

www.infysec.com

enquiry@infysec.com

to identify the platform (Platform) and to handle process control (ProcessControl) – including using the native library support if configured for that platform. The DomainManager class is used for configuration and security management for the Node Manager for that domain, and this uses the `ServerManager` class to interact with the Admin Server for each configured domain to obtain the up to date domain topology. After this simple interaction with a pre-configured URL (`bea_wls_management_internal2/wl_management`) to access the domain configuration, no further interaction with the Admin Server is necessary.

The `startNodeManager.sh` script displays the environment variables used, that it has obtained from the configuration properties file for the Node Manager, in the `nodemanager.properties` file under the common/nodemanager subdirectory. The details of the domain being controlled are displayed and eventually the Node Manager completes its initialization with a message to the console that it is listening for instructions from the Admin Server.

```
<Jan 24, 2010 4:08:56 PM> <INFO> <Secure socket
listener started on port 5556>
```

After the Node Manager is started the next step is to start some WLS instances.

WEBLOGIC SERVER INSTANCES

The common/bin directory contains a script that starts the WebLogic instances under the control of the Node Manager, the `startManagedWebLogic.sh` script. This takes as parameters the name of the managed server configuration to start and the URL to listen on.

```
[weblogic@fedora1005 bin]$ ./startManagedWebLogic.sh
managedserver1 http://localhost:7001
```

The script again displays the environment and configuration information, and goes on to call the `weblogic.Server main()` method to bootstrap the server itself. Entering an incorrect user id and password gives a stack trace that shows a little about the internal architecture (Listing 1).

However, in normal usage the instances, including the Admin Server, will be started for a given domain configuration. This can be found in our config "root" of `user_projects/domains` under a directory tree for the given domain, which is called `base_domain` by default. Under this directory structure we find the bin directory and in this the `startWebLogic.sh` script that starts the target server, in our case the Admin Server.

So, to examine the process of "bootstrapping" the application we need to undertake a bit of detective work. We know that the script results in the command line being generated that starts the application server instance with the appropriate configuration. This command line is complex, but ultimately results in the "main" method of some class being called. For example, we can see in the command line below that the "main" method of the webLogic.Server class is called as the entry point to start the application server instance (Listing 2).

This tells us the entry point, but not everything that is loaded and when it is loaded. For this we want to ask the Java virtual machine to log every class that is loaded by the classloader mechanism, which requires the use of `-verbose:class` setting on the command line. To enable this for a WebLogic Server instance the value must be set for the given instance via the Admin Console by clicking on the link for the instance under the Domain Structure tree control under Environments and its Servers sub-branch. The "Configuration" tab must be selected and the "Server Start" subtab to find where to add the JVM command line settings (Figure 1).

When a JVM command line option is added to the configuration it results in a server-start XML element with an arguments sub-element containing the JVM options for that server entry in the config.xml file for the domain. This gets picked up by the `startWebLogic.sh` script when it calls the `setDomainEnv.sh` script and sets the `JAVA_OPTIONS` environment variable to start the JVM process.

The first entry to appear in the output is that for the `webLogic.Server` class itself. We will trace the flow through this bootstrap thread later on.

```
[Loaded weblogic.Server from file:/opt/Oracle/
Middleware/wlserver_10.3/server/lib/weblogic.jar]
```

Soon after this class has been loaded, another set of classes are loaded from the `com.bea.core.`

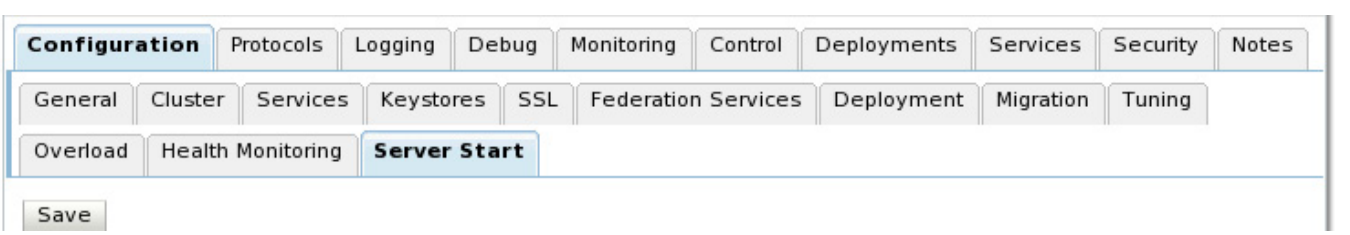


Figure 1. *WebLogic Console Configuration Options*

Listing 3a. *Java output for verbose:class*

```
[Loaded weblogic.management.DomainDirConstants from file:/opt/Oracle/Middleware/modules/com.bea.
core.management.core_2.5.0.0.jar]
[Loaded weblogic.management.DomainDir from file:/opt/Oracle/Middleware/modules/com.bea.core.
management.core_2.5.0.0.jar]
[Loaded weblogic.management.bootstrap.BootStrapConstants from file:/opt/Oracle/Middleware/modules/
com.bea.core.management.core_2.5.0.0.jar]
[Loaded weblogic.management.bootstrap.BootStrapMin from file:/opt/Oracle/Middleware/modules/com.
bea.core.management.core_2.5.0.0.jar]
[Loaded weblogic.management.bootstrap.BootStrap from file:/opt/Oracle/Middleware/modules/com.bea.
core.management.core_2.5.0.0.jar]
[Loaded weblogic.Home from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/weblogic.jar]
[Loaded weblogic.Home$HomeSingleton from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/
weblogic.jar]
...
[Loaded weblogic.utils.classloaders.ClassFinder from file:/opt/Oracle/Middleware/modules/com.bea.
core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.classloaders.MultiClassFinder from file:/opt/Oracle/Middleware/modules/com.
bea.core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.classloaders.ClasspathClassFinder2 from file:/opt/Oracle/Middleware/modules/
com.bea.core.utils.classloaders_1.6.0.0.jar]
[Loaded java.lang.IllegalArgumentException from /opt/Oracle/Middleware/jdk160_14_R27.6.5-32/jre/
lib/rt.jar]
[Loaded weblogic.utils.collections.CopyOnWriteArrayList from file:/opt/Oracle/Middleware/modules/
com.bea.core.utils_1.7.0.0.jar]
...
[Loaded weblogic.utils.collections.CopyOnWriteArrayList$COWIterator from file:/opt/Oracle/
Middleware/modules/com.bea.core.utils_1.7.0.0.jar]
...
[Loaded weblogic.utils.classloaders.ZipClassFinder from file:/opt/Oracle/Middleware/modules/com.
bea.core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.zip.Handler from file:/opt/Oracle/Middleware/modules/com.bea.core.
utils_1.7.0.0.jar]
[Loaded weblogic.utils.zip.ZipURLConnection from file:/opt/Oracle/Middleware/modules/com.bea.core.
utils_1.7.0.0.jar]
[Loaded weblogic.utils.classloaders.ClassFinderUtils from file:/opt/Oracle/Middleware/modules/com.
bea.core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.OptionalPackageProvider from file:/opt/Oracle/Middleware/modules/com.bea.
core.utils_1.7.0.0.jar]
[Loaded weblogic.utils.classloaders.NullClassFinder from file:/opt/Oracle/Middleware/modules/com.
bea.core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.classloaders.ClasspathClassFinder from file:/opt/Oracle/Middleware/modules/
com.bea.core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.io.FilenameEncoder$UnsafeFilenameException from file:/opt/Oracle/Middleware/
modules/com.bea.core.utils_1.7.0.0.jar]
[Loaded weblogic.utils.Debug from file:/opt/Oracle/Middleware/modules/com.bea.core.utils_1.7.0.0.jar]
...
[Loaded weblogic.utils.DebugCategory from file:/opt/Oracle/Middleware/modules/com.bea.core.
utils_1.7.0.0.jar]
...
[Loaded weblogic.utils.classloaders.DirectoryClassFinder from file:/opt/Oracle/Middleware/modules/
com.bea.core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.io.FilenameEncoder from file:/opt/Oracle/Middleware/modules/com.bea.core.
utils_1.7.0.0.jar]
[Loaded weblogic.utils.classloaders.ZipSource from file:/opt/Oracle/Middleware/modules/com.bea.
core.utils.classloaders_1.6.0.0.jar]
```

Listing 3b. *Java output for verbose:class*

```
[Loaded weblogic.utils.classloaders.JarSource from file:/opt/Oracle/Middleware/modules/com.bea.
core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.classloaders.FileSource from file:/opt/Oracle/Middleware/modules/com.bea.
core.utils.classloaders_1.6.0.0.jar]
[Loaded weblogic.utils.io.ExtensionFilter from file:/opt/Oracle/Middleware/modules/com.bea.core.
utils_1.7.0.0.jar]
[Loaded weblogic.utils.FileUtils from file:/opt/Oracle/Middleware/modules/com.bea.core.utils_1.7.0.0.jar]
...
[Loaded weblogic.utils.FileUtils$1 from file:/opt/Oracle/Middleware/modules/com.bea.core.utils_1.7.0.0.jar]
[Loaded weblogic.utils.FileUtils$2 from file:/opt/Oracle/Middleware/modules/com.bea.core.
utils_1.7.0.0.jar]
...
[Loaded weblogic.t3.srvr.T3Srvr from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/weblogic.jar]

[Loaded weblogic.security.spi.Resource from file:/opt/Oracle/Middleware/modules/com.bea.core.
common.security.api_1.0.0.0_5-2-0-0.jar]

[Loaded weblogic.server.ServerLifecycleException from file:/opt/Oracle/Middleware/wlserver_10.3/
server/lib/weblogic.jar]

[Loaded weblogic.t3.srvr.T3SrvrConsole from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/
weblogic.jar]

[Loaded weblogic.utils.ErrorCollectionException from file:/opt/Oracle/Middleware/modules/com.bea.
core.utils_1.7.0.0.jar]

[Loaded weblogic.server.ServiceFailureException from file:/opt/Oracle/Middleware/modules/com.bea.
core.weblogic.lifecycle_1.3.0.0.jar]

[Loaded weblogic.t3.srvr.T3Srvr from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/weblogic.jar]

[Loaded weblogic.security.spi.Resource from file:/opt/Oracle/Middleware/modules/com.bea.core.
common.security.api_1.0.0.0_5-2-0-0.jar]

[Loaded weblogic.server.ServerLifecycleException from file:/opt/Oracle/Middleware/wlserver_10.3/
server/lib/weblogic.jar]

[Loaded weblogic.t3.srvr.T3SrvrConsole from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/
weblogic.jar]

[Loaded weblogic.utils.ErrorCollectionException from file:/opt/Oracle/Middleware/modules/com.bea.
core.utils_1.7.0.0.jar]

[Loaded weblogic.server.ServiceFailureException from file:/opt/Oracle/Middleware/modules/com.bea.
core.weblogic.lifecycle_1.3.0.0.jar]

...
[Loaded weblogic.management.WebLogicMBean from file:/opt/Oracle/Middleware/modules/com.bea.core.
management.core_2.5.0.0.jar]

[Loaded weblogic.descriptor.SettableBean from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/
weblogic.jar]

[Loaded weblogic.descriptor.DescriptorBean from file:/opt/Oracle/Middleware/modules/com.bea.core.
descriptor_1.7.0.0.jar]
```

Listing 3c. *Java output for verbose:class*

```
[Loaded weblogic.management.configuration.ConfigurationMBean from file:/opt/Oracle/Middleware/
wlserver_10.3/server/lib/weblogic.jar]
[Loaded weblogic.management.configuration.OverloadProtectionMBean from file:/opt/Oracle/Middleware/
wlserver_10.3/server/lib/weblogic.jar]

[Loaded weblogic.t3.srvr.T3Srvr$2 from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/
weblogic.jar]

[Loaded weblogic.t3.srvr.T3Srvr$1 from file:/opt/Oracle/Middleware/wlserver_10.3/server/lib/
weblogic.jar]

[Loaded weblogic.diagnostics.debug.DebugLogger from file:/opt/Oracle/Middleware/modules/com.bea.
core.logging_1.6.0.0.jar]

[Loaded weblogic.diagnostics.debug.DebugLogger$1 from file:/opt/Oracle/Middleware/modules/com.bea.
core.logging_1.6.0.0.jar]

[Loaded weblogic.diagnostics.debug.DebugLoggerRepository from file:/opt/Oracle/Middleware/modules/
com.bea.core.logging_1.6.0.0.jar]

...
```

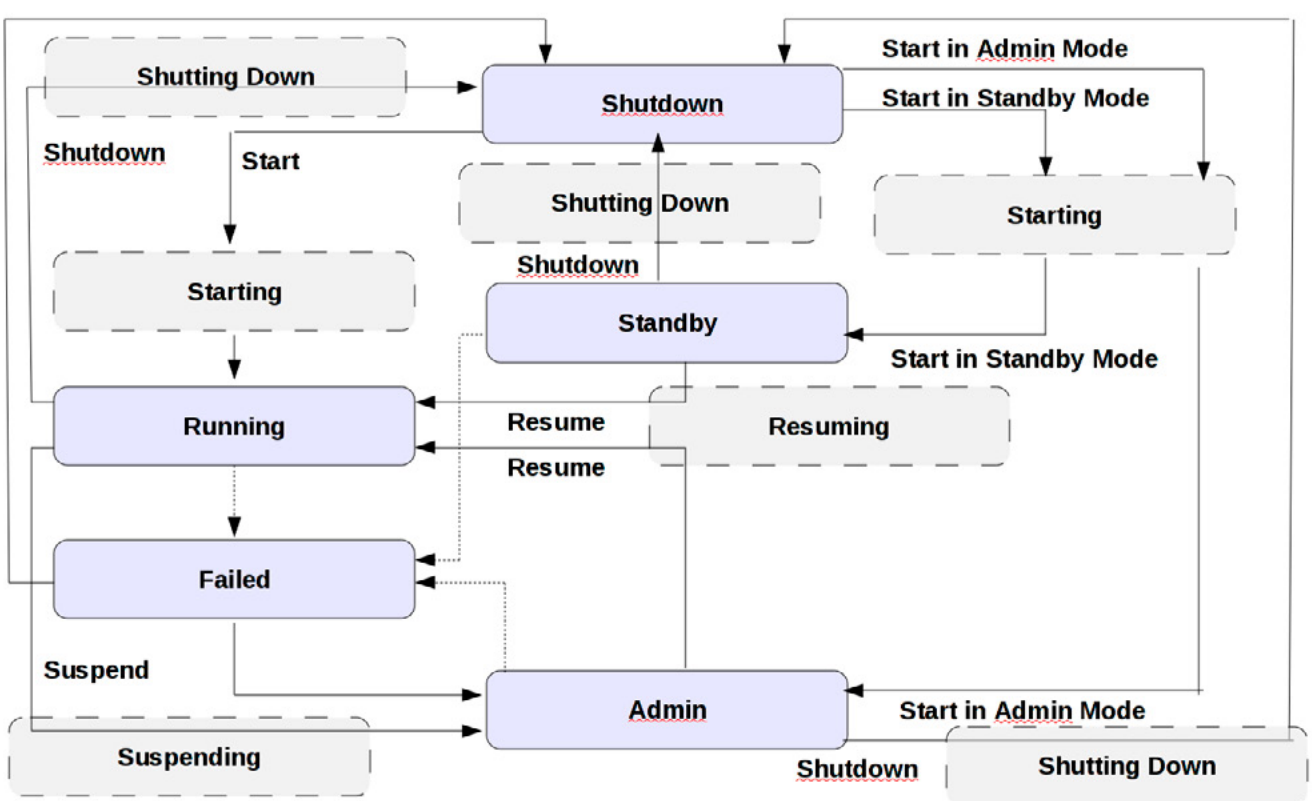


Figure 2. *WebLogic Server States*

`management.core_2.5.0.0.jar` file to start another stream of management initialization and WebLogic specific classloaders are loaded from `com.bea.core.utils.classloaders_1.6.0.0.jar` before control returns to the weblogic.jar. Note the `webLogic.Home` class being loaded before the classloaders (Listing 3).

The Server class main method starts immediately running the `T3Srvr` class subsystem from the `t3.srvr` package in the weblogic.jar file. This T3 critical subsystem is part of the core of WebLogic Server and is a kind of *Object Request Broker* (ORB), protocol handler, and messaging server inside WebLogic that predates RMI and allows the internal subsystems and some external callers to interact with the WebLogic core. The history of the WebLogic server is important here to explain how this subsystem came to exist, with the product originally developed as a three-tier server for database access via JDBC drivers under the name of Tengah so the T3 name refers to the Tengah legacy and the number of tiers.

Fortunately, at this point when the T3Srvr subsystem starts the services that underpin the application server the administrative facilities give us help in understanding the services, how they start, and in what order they are started. First, we must understand the different modes in which the services run. An instance can be in Shutdown mode (i.e. not started), Failed mode, Standby mode, Admin mode or Running mode, with a carefully coordinated set of transitions between each mode that leads to the Shutting Down, Starting, Suspending and Resuming transient pseudo modes. With each mode and transition different

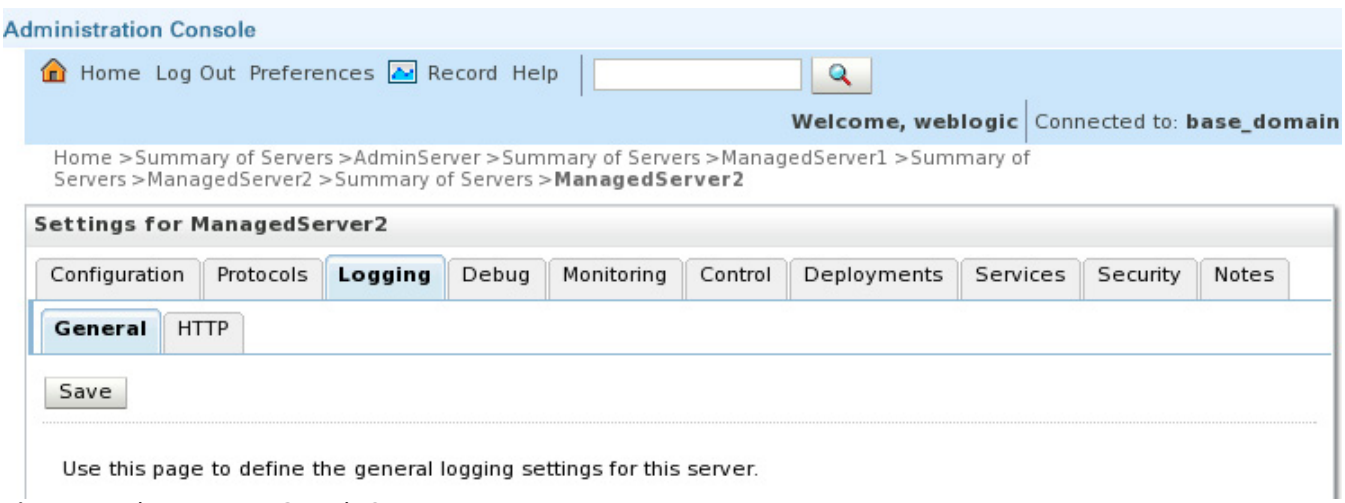


Figure 3. *Administration Console Options*

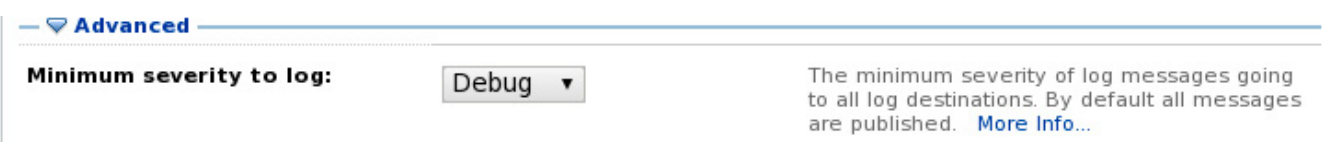


Figure 4. *Administration Console Debug Logging*

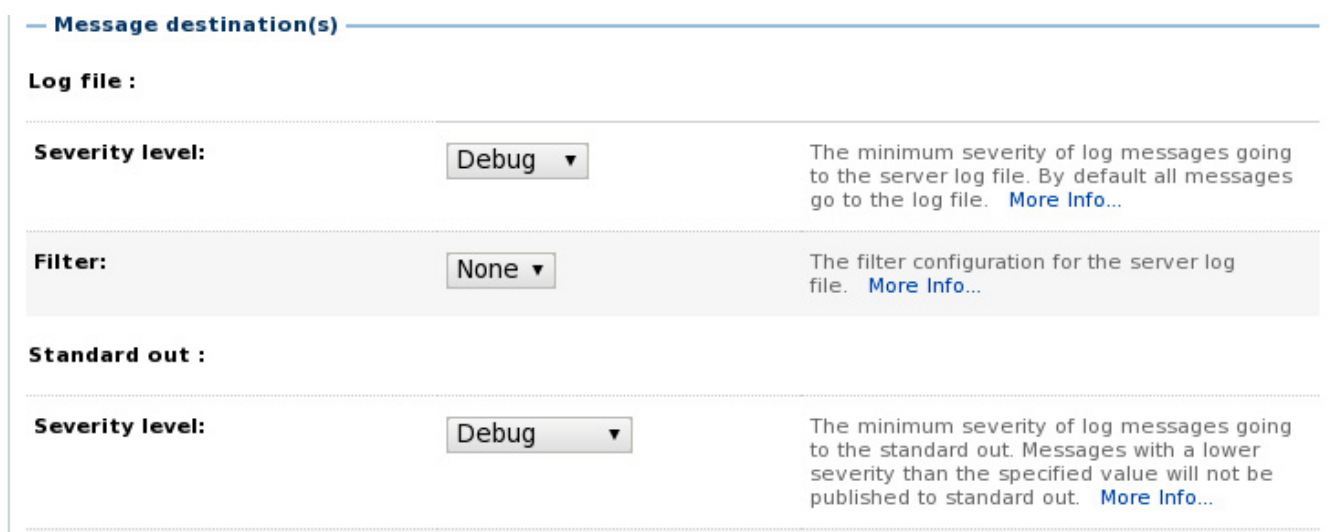


Figure 5. *Administration Console Logging Details*

IMF
INTERNATIONAL MANAGEMENT FORUM

IT Security Courses and Trainings

IMF Academy is specialised in providing business information by means of distance learning courses and trainings. Below you find an overview of our IT security courses and trainings.

Certified ISO27005 Risk Manager
Learn the Best Practices in Information Security Risk Management with ISO 27005 and become Certified ISO 27005 Risk Manager with this 3-day training!

CompTIA Cloud Essentials Professional
This 2-day Cloud Computing in-company training will qualify you for the vendor-neutral international CompTIA Cloud Essentials Professional (CEP) certificate.

Cloud Security (CCSK)
2-day training preparing you for the Certificate of Cloud Security Knowledge (CCSK), the industry's first vendor-independent cloud security certification from the Cloud Security Alliance (CSA).

e-Security
Learn in 9 lessons how to create and implement a best-practice e-security policy!

Information Security Management
Improve every aspect of your information security!

SABSA Foundation
The 5-day SABSA Foundation training provides a thorough coverage of the knowlegde required for the SABSA Foundation level certificate.

SABSA Advanced
The SABSA Advanced trainings will qualify you for the SABSA Practitioner certificate in Risk Assurance & Governance, Service Excellence and/or Architectural Design. You will be awarded with the title SABSA Chartered Practitioner (SCP).

TOGAF 9 and ArchiMate Foundation
After completing this absolutely unique distance learning course and passing the necessary exams, you will receive the TOGAF 9 Foundation (Level 1) and ArchiMate Foundation certificate.

TOGAF

cloud CSA security alliance

CCSK

CompTIA CLOUD ESSENTIALS

ArchiMate®

SABSA

For more information or to request the brochure please visit our website:
http://www.imfacademy.com/partner/hakin9

IMF Academy
info@imfacademy.com
Tel: +31 (0)40 246 02 20
Fax: +31 (0)40 246 00 17

services that form part of the core functionality of the Oracle WebLogic Server are started and stopped (Figure 2).

To see the services started in each mode and the order in which they start we simply switch on debug logging for the T3Srvr ServerLifeCycle debug option, which ultimately maps to an internal flag called "weblogic.slc" that tells the T3Srvr ServerServicesManager to log each state transition and the service names as each one starts. To do this we first have to enable Debug logging to the log files and the redirected standard output log. Start up the Node Manager (`startNodeManager.sh`) and the server instance hosting the Admin Server (`startWebLogic.sh`) and after logging in click on the Environments branch link in the Domain Structure tree control on the left side of console main page, followed by the Server link. This will bring up a list of servers and for each click on its link and select the Configuration tab (Figure 3).

When setting the configuration click the Logging subtab followed by the Advanced container link towards the bottom of the page. In this container, for the log file and standard output change the settings to log debug messages. Obviously, you don't want to do this on a production server as the performance will be severely impacted, so a throwaway development environment is probably the best sandpit to play in with this. First change the minimum severity to log section to tell the environment that it is in debug mode (Figure 4).

Next setup the destinations for the log entries to ensure we have some output to examine (Figure 5).

Once the settings have been change click on the "Save" button to commit the changes. Next select the Debug tab at the top of the page. For this we need to tell the application server which subsystems we want the debug messages to be logged for, in our case the T3Srvr ServerLifeCycle monitoring and the time taken for each subsystem to process the requests (Figure 6).

The page will, by default, contain two sub-trees; "default" and "weblogic". In our case we want the "weblogic" sub-tree so click on it and look

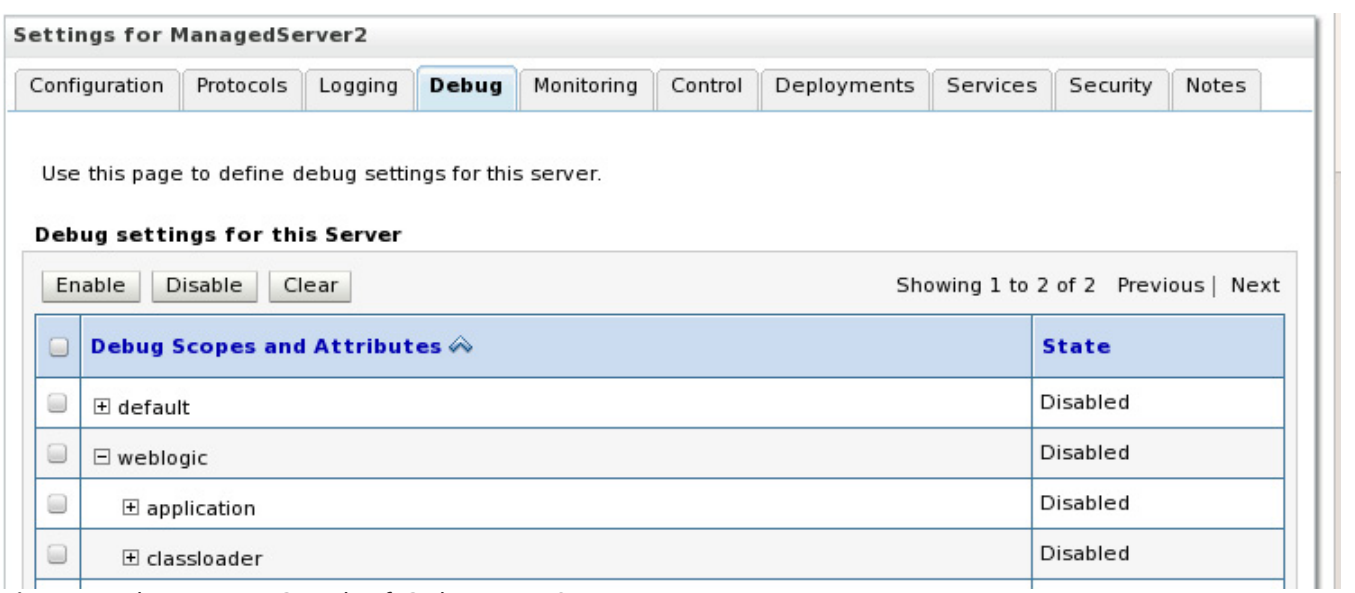


Figure 6. *Administration Console LifeCycle Logging Scope*

Figure 7. *Administration Console LifeCycle Logging Detail*

Listing 4. *WebLogic LifeCycle Logging*

```
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <Time taken to start
weblogic.diagnostics.lifecycle.DiagnosticFoundationService: 1157 ms>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <creating service:
weblogic.nodemanager.NMService>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <starting service:
weblogic.nodemanager.NMService>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <Time taken to start
weblogic.nodemanager.NMService: 136 ms>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <creating service:
weblogic.timers.internal.TimerService>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <starting service:
weblogic.timers.internal.TimerService>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <Time taken to start
weblogic.timers.internal.TimerService: 16 ms>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <creating service:
weblogic.rjvm.RJVMService>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <starting service:
weblogic.rjvm.RJVMService>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <Time taken to start
weblogic.rjvm.RJVMService: 224 ms>
<03-Apr-2010 19:33:11 o'clock BST> <Debug> <ServerLifeCycle> <BEA-000000> <creating service:
weblogic.protocol.ProtocolService>
...
```

down towards the bottom of the tree and open the "t3" branch followed by "srvr" and click on the check boxes for `DebugServerLifeCycle` and `DebugServerStartStatistics`. Next click on the "Enable" button (Figure 7).

This entire process should be repeated for each type of server you want to monitor. When the process is complete, return to the Domain Structure tree control, Environment tree and Servers subtree and click on the target server whose configuration was just changed and click on "Shutdown" followed by "Start", leaving the "Admin Server" last to process. When the application server instances restart they will log the subsystems and services being started in each mode to the application server logs and the JVM process standard output. We can collect these from the `user_projects/domains` and target domain `servers/<SERVER_NAME>/logs` directory and examine them.

If examine just the `ServerLifeCycle` entries in the standard output file we will see that the Oracle WebLogic Server runtime makes use of a considerable number of base underlying services to provide the support needed for the containers. The file will have entries similar to the following, but a large number of them (Listing 4).

SUMMARY

So, what have we achieved? In the first article we looked at how a Java process executes on a Java virtual machine, and examined some of the tools for examining a complex Java package. In this second article we applied some of these tools – using the startup of the WebLogic Server JavaEE application server from Oracle as an example. We could have taken it further and used the jad decompiler to recover something equivalent to the original source code, but it wasn't necessary in this case and avoiding doing it also helps to avoid legal issues.

Author bio

Colin Renouf is a long standing enterprise solutions architect with thirty years experience in the industry – concentrating on the finance sector. He has authored many magazine articles ranging from Unix, through Java and on to security; and has also written and contributed to books on the subject. He is currently contracting for a well known credit card company, but his main loves are Australia and some of its people, his children, singing, photography and just being with good company. Oh, and quantum physics as he is an eternal scientist."

INTELLECTUAL PROPERTY: MAJOR THREAT TO SECURITY

by Jason Pfoutz

Do you live in the mindset that everything should be free, or in the mindset that individuals should get what they paid for? It is encouraging to know that many of the common issues of digital thievery are beginning to resolve. However, intellectual property is important to safeguard, because it belongs to a specific person or company.

When someone does not want their "recipe" leaked, people should honor that – or consequences shall occur. Now, quite a few companies have been under scrutiny for digital thievery, piracy, copyright infringement, etc. Companies like Megaupload or The Pirate Bay have been under serious investigations and/or busts this year. Megaupload was one of the biggest cases in intellectual property issues and copyright infringement, and their fall shocked many people.

The file sharing sites known as The Pirate Bay, Megaupload, and many others are websites, in which users would be allowed to upload/share any content, copyrighted or uncopyrighted, allowing free access to all users. The biggest problems faced in these file sharing sites are that they refused to remove/monitor copyrighted content, which by law, was illegal to have on their website in the first place. Because of the refusal to remove such content, and some of their blatant replies to takedown notices, investigations and arrests were made in attempts to control this catastrophe.

DEMISE OF MEGAUPLOAD AND INVESTIGATION OF KIM DOTCOM

Megaupload was a popular file-sharing website, until its bust in January of 2012. A U.S. investigation checked on downloads, which contained music, videos (or movies), and other content. From the facts, it appears billions of dollars were being lost in revenues of copyright holders and legitimate companies. Kim Dotcom, founder of Megaupload, established the site in 2005, which targeted the file-sharing business.

Due to Dotcom's history of embezzlement back in the late 1990s, it became no surprise that his business activity would still be illegitimate or risky. The investigation seemed to reveal details that key points of Megaupload's business models were based on criminal intent. It also appeared that Dotcom was focused more on mining money from other businesses and people. Many uploaders were apparently getting incentives to upload (infringing) content in exchange for payments. Even small file sharing businesses worked off the infrastructure of Megaupload, which helped Megaupload become much larger.

Before Megaupload's bust, the company had helped with a secret U.S. search warrant, which, according to interviews, was targeting five of its users running their own file-sharing service, using Megaupload's infrastructure.

One of its busts included NinjaVideo, a site created in 2008 aimed at uploaded videos of TV shows, documentaries, movies, and more. The site shut down back in June 2010, but investigators figured the problem was with Megaupload, so the investigation pointed more towards them. However, arrests were made on NinjaVideo, and the founder, Matthew David Howard Smith, was charged with criminal copyright infringement in September 2011. In addition, Hana Beshara, the other founder, also arrested, had been charged with conspiracy and criminal copyright infringement.

Many others took part in operations that were also arrested and charged. The reason for the steered investigation is that although NinjaVideo hosted the content on their site, it was truly on Megaupload servers. With it being on Megaupload servers, it is possible that NinjaVideo founders and other users did not upload or infringe the content directly. Despite being discovered for the pirated videos, Megaupload did not remove them from their servers, which complicated the investigation.

During one of the investigations for Megaupload, many companies such as Sony Music Entertainment Inc., Business Software Alliance, and even Warner Bros. Entertainment Inc., sent tons of notices apparently, under the DMCA. Even the RIAA and MPAA had some claims issued. However, these notices were ignored.

Authorities claimed in many news/press releases that this was one of the largest copyright infringement cases brought to the United States, and that Megaupload commonly facilitated copyright infringement of movies, even "before the theatrical release…" The government estimated the harm to copyright holders was "well in excess of $500 million." But, with all of these charges and investigations, Kim Dotcom is not done with his entrepreneurship (wondering if he thinks this is just a bump in the road to something even better).

Kim Dotcom appears to have new plans, as on his Twitter account, he notes many new ideas, especially with a planned launch of "Mega" in January 2013. It is marked for one year after police raided Dotcom's mansion, apparently, and he still claims that nothing is wrong. Dotcom seems to think that his company did not infringe copyrights to the extent of violating laws, and that a future company is worth a try.

Now, Dotcom uses the fact that the courts' ruling of the warrants was invalid, when police raided his home, and that police may have used too much force. This furthered delayed an extradition hearing until March 2013. Because of this, Dotcom seems to have plans for Mega, and that it will top all other ventures he's done before. But, the saying still sticks, "cheaters never prosper."

THE PIRATE BAY EVADES AUTHORITIES

Megaupload was only one of the latest busts. The Pirate Bay, Swedish BitTorrent site, has been un-

der constant struggle since May 2006, from the blocking of their site in many countries, to "The Pirate Bay trial" beginning in 2008 and 2009. Four individuals involved in The Pirate Bay were charged for criminal copyright infringement. The fact that over thirty copyright claims were tacked on to the investigation of The Pirate Bay, a trial was prompted in early 2009.

All four who were arrested in The Pirate Bay case were also found guilty, and to be sentenced to one year in prison. But, did that stop these people? Not at all. They appealed the verdict, which shortened the prison sentences. Just like the case with Megaupload (dealing with the RIAA and MPAA), The Pirate Bay had some pressure from the MPAA to take the content down. That the MPAA had helped encourage the raid by police in May of 2006. The raid was unsuccessful, which was why the operations (and investigations) continued in the following years.

During their trials in 2009, charges had been partially dropped at first, because the prosecutors claimed that their business was legal because it could be used for "legal and illegal activity", so it is up to the user of the service, not the company behind it, to commit action (whether legal or illegal).

In the following trials after the first and second, attorneys kept trying to introduce evidence that was not already discussed pre-trial or shared. Also, prosecution kept trying to say that The Pirate Bay made its money helping others violate copyright law and it succeeded. The four operators of the site were convicted of being an accessory to copyright infringement, and sentenced to one-year in jail and a few million dollars in fines.

Due to the verdicts, several file-sharing websites in Sweden closed voluntarily to avoid police investigation. After all, of the fuss that occurred, and when The Pirate Bay switched to its .se domain from its .org one, The Pirate Bay is still in operations.

New data has occurred recently as The Pirate Bay founder is going to be investigated soon for fraud. Gottfrid Svartholm Warg was arrested this past September for alleged charges towards copyright violation and hacking into the computer network of Logica. Logica is a private tax services company for the Swedish government. He also apparently had many other links to data intrusion cases, some of which are unproven. However, charges are yet to be filed in the case. The officials in Stockholm continually renew the detention of Warg until the investigation is over and charges are formally made. The time spent in detention will likely be removed from the one-year sentence acquired from The Pirate Bay case.

The criminal copyright violations are still hanging over the heads of the founders, and are sure to cause major trouble for Warg when charges are officially filed. Frequently, government authorities have checked into the usage of magnet links, which was popularized by the BitTorrent community.

MAGNET LINKS AND IP ADDRESS SHARING

Magnet links are very popular on the BitTorrent community, and are characterized as a cryptographic hash value used to identify content rather than location. Therefore, this masks the URL with a cryptographic hash value that in turn cannot be tracked by authorities. It allows resources to be available for referral even if it is unavailable. The Pirate Bay dropped torrents entirely in the past few years and only hosts magnet links.

What's the difference? Torrent files are a small file that contains information about a larger file you want to download. It tells the torrent client the names of the files being shared, the URL, etc., and then allows the client to calculate the hash and check the network for uploaders who have that specific file available. After that, it offers the download, and you're done.

Magnet links, however, does away with the need for a client to figure out the hash. With a magnet link being already prepared, the client can simply get busy looking for uploaders that have your specific file available. No special file in between is needed, or anything else. It simply provides a means to connect you right away with downloads you need.

What's good about magnet links in the eyes of the authorities? It saves the company hosting the magnet links from legal trouble, because the company does not have the actual copyrighted content. They don't host the content, they won't get in trouble. This is at least more difficult for companies like The Pirate Bay to get into legal trouble, but the authorities are still picking through investigations to find out the true legality of this URI scheme. If The Pirate Bay is not directly making copyrighted material available, what can the governments do? This characterizes their eluding of the authorities, as The Pirate Bay has done everything they can to get the authorities off their back. But, the fraud-related issues and the blaming for IP address sharing is putting The Pirate Bay back in hot water.

What makes the authorities as well as security experts more worrisome is IP address sharing, which is becoming a major problem with different companies, including The Pirate Bay. Recently, The Pirate Bay was accused for collecting its users' IP addresses by the Anonymous hacking group, who published images of the administrative section of The Pirate Bay showing the users and their listed emails and IP addresses. Just to make sure no unjust claims are made, it states in

their Privacy Policy one things and then contradicts itself later.

On the Privacy Policy page of The Pirate Bay, it states, "The tracker may not be used by anyone with the intention to track usage, log IP addresses/usage or anything else that we consider intrusion of privacy or disruption of tracker service."

After that, it notes, "We also reserve the rights to publish any information regarding violations. Info hashes, IP addresses and all other information that is supplied to the tracker will be considered our right to publish."

IP address sharing is very much frowned upon in the security community, and it is hopeful that the authorities will figure something out in The Pirate Bay case, before any more damage is caused. The signification of IP packet sniffing by governmental authorities has taken a new rise recently.

Deep Packet Inspection (DPI) is rising up, especially when needed by the government and larger companies, to help examine the data part of an internet packet (a small file with data that is sent over a network), in order to spot fraudulent activity, viruses, spam, and other intrusions. Many organizations have done a good job with this, including using port mirroring, which automatically monitors a port and copies all packets over from what goes through the port. As it looks at the headers of each packet (the separate standards used for IP addresses such as IPv4 or IPv6), it can also dynamically analyze where the attacks are coming from.

How is it used in the investigations for copyright theft? The *Internet Assigned Numbers Authority* (IANA) is controlled by the US Government, which controls and assigns all IP addresses, and specific ports/protocols allowed for data input and output (TCP/UDP). Therefore, all ports can be monitored or mirrored (specifically the ports that have common violations), and can track down the copyright thieve(s) and try to subvert the problem. However, this isn't the only way to try and do that, as the RIAA and MPAA have their methods as well.

RIAA AND MPAA ATTEMPTS TO TRAP INFRINGING USERS

The RIAA lists many sites on its notorious list for copyright violators, and they claim that BitTorrent sites should filter out copyrighted content proactively. Sites that also allow similar content to stay on their servers (like RapidShare) may face the same fate as Megaupload. That's what is hopeful in the RIAA's letter to US authorities containing the notorious websites. The efforts from the RIAA, MPAA, and other sites have led to the demise of Demonoid, Megaupload, BT Junkie, etc.

Many authorities are begging the same fate for The Pirate Bay, noting that The Pirate Bay is no different from these other pirating sites. Some authorities consider that the piracy problem won't go away, but if the authorities can manage it or keep it under control by getting the file-sharing sites under control, much of the issues can be resolved with copyright violation.

Many different methods being experimented in new technology is trackers that are being placed inside applications on newer operating systems, that reads the unique key that allows the user to use the copyrighted content (means they purchased it), and compares it to the registration information that is behind the unique key (called a unique identifier or UID).

The UID is tagged with the registered user, whom was verified as the sole owner, and that the version of the app that the user is using is not present on any other device/system. This eliminates the need to search around for pirated content. The registration tagged UIDs will show up in one database in the company's infrastructure, and if it is duplicated, then alleged piracy is present. After the duplication is recognized, the company can begin investigation into the case, and resolve it quickly. This shall further eliminate any other problems with piracy.

Microsoft has implemented similar technology in Windows 8, as it requires a UID, rather than just a product key or other usage. This saves times wondering who pirated the software, and makes it very difficult to pirate.

Now, with the RIAA considering unfiltered file-sharing sites as pirate havens, it may be in order that authorities worldwide check into each unfiltered file-sharing site to check for piracy. In the meantime, changes to laws concerning intellectual property are being done to help manage the issues, as well.

On Monday, November 20, 2012, parts of the Senate and House of Representatives approved a law that would better equip in the fight against intellectual property theft and other copyright issues. The provisions for the law allow for one group to handle copyright matters, trying to appoint a government agency to rely on that will respond to copyright concerns. It also provides for copyright-based industries to be in better hands against intellectual property theft.

A classic trend from the RIAA and MPAA is to upload their own torrent honeypots, in order to catch the offenders. Now, this isn't anything new. These organizations have been doing the honeypot work for about ten years, and have functioned successfully at catching many offenders of copyright theft. What is the main aim? It was back then when it started to catch actual users who were stealing a ton of copyrighted data (obviously without paying), to now being used to track down those who actually host the copyrighted data.

The way the tracker works, that is attached to torrents, is that it collects the IP address of all the "pirates" who try to download the files. These days, people might wonder about their own privacy. However, justice must be served to stop the mess of intellectual property thievery.

Many software developers have designed tools to counteract this honeypot act, by engineering software to detect fake torrents. It automatically screens the list of torrents and reveals those that are fake in nature. It's engineered by searching for fake torrents by keyword or hash. What's sad is, most people that fall for the traps are unaware they've been trapped. For those aware of such traps, they can surely install the fake torrent tracker. The idea is to find the big offenders and find where the content is hosted.

FILE SHARING: LEGAL, UNSAFE AND UNBELIEVABLE PART II

File sharing, in general, is legal. That is – as long as it doesn't include copyrighted content. People share "files" all the time on social networks and other websites by posting content, such as videos, photos, text, etc. However, what is considered copyright theft on social networks is grounds for account deletion policy. If you post problematic copyrighted content, expect some trouble. Social networks commonly have no-tolerance policies for copyrighted data being posted on their site. Just like pornography, it'll be removed quickly.

When you go to file sharing websites, however, you may see copyrighted content posted. If you do, there is either not a policy on that site against posting copyrighted content, or the material is fresh. When different firms or individuals see their copyrighted content posted, they send a takedown notice, which gives the website some time to remove the content. Usually, the firms/individuals that send the notice give the website owners will give two-to-three written notices before legal action is taken. Sometimes, legal action is not taken, which makes it difficult for governmental organizations to be aware that fraudulent activity is taking place.

Just like it's always been on movies and other home videos, you see a copyright notice/FBI warning or other government warning about copying, modifying, etc. of the material. Obviously, nobody would want five years in prison or a $250,000 fine for copying video material. However, people do it anyway, and run the risk. This is the biggest problem to be seen light up in the file sharing industry is people thinking it's okay to steal (copyrighted) content. However, the government has been trying to make it clear for many years that it's not okay.

One big problem to be seen with a file sharing user is that viruses and other forms of malware get installed on their computer. This puts not only the computer's data at risk, but also the identity of the user. Many viruses and malware attempt to steal private data in hopes to gain a new identity. This new identity is used for monetary reasons (so they can refill their wallets). That's just one of the many troubles and struggles in file sharing and why some users stop doing it. If file sharing problems are not enough, even users are changing their smartphone and other devices to harbor illegal content.

JAILBREAKING: LEGAL, UNSAFE AND UNBELIEVABLE PART II

Jailbreaking is a hot topic in intellectual property, because it doesn't necessarily characterize the thievery of it, but the idea of modifying it. Jailbreaking is when users hack into the device's code, effectively changing it to allow third party content. When the user does this, it will then be allowed to install third party apps, program code, and other means of trouble onto the device. This could provide for very serious complications for the device, if the user is not careful.

Apple and Microsoft as well as other device makers only allow first-party apps, which mean that their devices do not allow third-party apps. This helps with device security by allowing the device makers to carefully review all apps submitted for the device. If any of them are bad or have malicious means, they would not be added to the app stores. However, for third-party apps, there is no necessity for downloading apps from the OEM app stores, but instead can download them from third-party providers, or independent developers.

By allowing third-party apps, device makers put the security of the device and its users on the line, which can be problematic, because many app makers have malicious means. Many of the malicious means include stealing personal data (contacts, credit card numbers, other personal numbers, etc. – which is a direct identity theft), damaging the device purposely, or attempting to mine money (an indirect identity theft).

For these first-party types of devices, they can be jailbroken, which can provide users the opportunity to install third-party apps. What also happens? An open door for malicious software (malware). Third party app stores are only natively supported on Android, Symbian, and BlackBerry smartphone devices.

How does this play into intellectual property, you ask? Apple and Microsoft and many other device makers have specifically designed these devices under their own developers' intellectual minds (therefore making it intellectual property), and by jailbreaking the devices, the intellectual property is being modified from its original means.

What is sad about this situation, however, it is legal under the *Digital Millennium Copyright Act* (DMCA). This act, crafted back in 1998, provides for making it illegal to circumvent digital rights management. However, it did allow occasional exemptions to be made.

During the latest round of exemptions, which will be in force for three years as of October 28, 2012, includes five circumvention exemptions allowed. It may very well be arbitrary of the DMCA for this, but until a reformation comes for the act, odd exemptions will continue to occur.

The provisions include the ability to jailbreak smartphones, but not tablets. Also, phones bought before January 2013 can be unlocked (allows the user to take it to a different carrier to have services added to it). It is legal to rip DVDs to use an excerpt for a documentary, but illegal to play it back on devices in full.

For phone unlocking, it allows users to take their phones that were bought before January 2013 to be taken to another carrier to allow it to be unlocked and have new services added to the device. However, in January 2013, users would be required to request permission of the current carrier of said phone they want new services for, to be able to unlock it.

There are many other provisions, but the point is, where does the government draw the line on intellectual property? It is to be noted that deliberate copyright infringement is disallowed under the DMCA, and those that do it can be fined and possibly put in jail or prison because of it.

Jailbreaking and other potentially harmful copyright infringement techniques should be ended, because it is costing the companies behind the products/items a lot of money and time. If users are constantly in the mindset of wanting free stuff, then these tactics will not end.

Now, even if the users will not cooperate, that's pretty usual to an extent. However, if companies won't cooperate, then consequences should be executed for offending companies. What is the biggest problem is that some companies, particularly file-sharing companies, are not cooperating with takedown notices and are not instituting filtering processes to keep infringing data off their site and out of reach.

The landscape of intellectual property protection needs to be bettered by the fact that these offending companies start cooperating. This will ensure that the government doesn't try to execute too much control over the Internet (especially through their intellectual property and anti-piracy acts), and also legitimate companies don't lose any more money. It will also keep money in the pockets of popular artists, musicians, and actors (who aren't paid as much as you think).

From the Megaupload problems to The Pirate Bay, copyright infringement has become a serious problem affecting the Internet today. It is hard to share content and take part in social media, and know that you have Big Brother's watchful eye on your every move online. Whether or not people like it, the MPAA and RIAA among others are going to keep on setting traps for offenders and the government will keep on tracking and imposing restrictions to try to deter piracy/copyright infringement.

Computer security is always a major challenge, and is characterized by the fact that users just always want to push the red button. They want to keep on breaking security measures imposed to protect them. What happens with this? It causes data loss, data thievery, and lost money for too many people. If troublemakers would stop circumventing security tactics, we could all keep money in our pockets. Computer security is continuous, and will exist as long as troublemakers (and computers) are around. In addition, computer security organizations and experts need to stay on their guard, and keep defending the very Internet we love.

When we all realize the major problems with the thievery of intellectual property, and start taking much more drastic measures, people will be hurt. But, if that saves us all from lost data and money, then is it truly worth it? Should users be allowed to do what they want? We shouldn't disagree about computer security measures, anti-piracy, and other forms of protection – because they are made for our good. And because we like it when things work out for our good, we should take a stand against intellectual property theft. Once we take a stand, the problems will be much easier to solve to make our Internet a much more happy (and less government involved) place.

Author bio

Jason Pfoutz is the owner of seCURE Connexion and SecuraGeek Association. Two different security companies. The first listed is a key service for providing security technologies, while the second listed is tech support (security) based. He does software engineering, malware researching, business strategy, tech support, etc. He's the jack-of-all-trades, but of course somebody in the company has to be. To sum it all up, he's doing software development, researching, and technical support for the security community. He also works on several other tech support forums across the Internet: TechSpot.com, GeekPolice.net, ComputerHope.com, etc. http://secureconnexion.wordpress.com "Secure Connexion – Security News. Investigations. Vulnerability Assessments."

SELF COLLECTION IS RISKY BUSINESS

by Elias Psyllos

This article is not to discredit IT Departments and the individuals that work inside them, rather, this article is meant to shed light on why a Certified Forensic Examiner (Investigator, Analyst, etc.) should be used every time in any electronic discovery matter. For those companies that have appointed or contracted a forensic matter expert to handle the electronic discovery process, you are on the right path.

What you will learn:

- The risks involved with performing a self-collection
- The importance of having a Forensic Analyst perform the collection
- Why using an outside party for collection purposes is important
- An example of a self-collection gone wrong for in a court case

What you should know:

- The purpose of performing forensic collections
- How forensic collections fit into an E-Discovery matter (EDRM cycle)
- How forensic collections pertain to the business world

However, many companies do not have an internal forensic matter expert to manage the electronic discovery process, and this article is geared toward you.

Whenever a matter arises that requires the collection or preservation of Electronically Stored Information (referred throughout the rest of the article as ESI), most companies first thought is to have their internal IT department, create the "images" of the digital media involved in the matter. This is what is known as a "self collection".

IT DEPARTMENT

The topic of "self collection" has been one area of Computer Forensics and E-Discovery that is continuously discussed and debated. Self Collection, can be defined as "one of the parties involved in a matter, creating "forensic images" (or copies that they think are forensically sound) of digital evidence that can possibly be involved in a matter for preserving, collecting, and/or analyzing Electronically Stored Information regarding a Digital Investigation or E-Discovery matter."

IT Departments and the individuals within may be capable of creating "images" or copies, however, it may not be a forensically sound image and it may not follow the policies and procedures designated by the courts for collecting ESI. The outcome of this is a copy that is not forensically sound, and in turn may be rejected as evidence. Alternatively, because the "image" was not preserved correctly, relevant data could be missing, or altered.

Another prevailing aspect of the "self collection" mentality is compa-

nies assume that having their own IT people involved in the forensic process is a good idea. The assumption that the internal IT personnel know the systems and data the best may be correct, but it may cause a negative effect in the overall process. IT staff, although an expert in the IT field, may not know or understand the correct procedures for collecting or preserving ESI. As well, internal IT departments may not have the correct paperwork, such as Chain of Custody forms to include in the preservation or collection needed in these matters. Having a neutral third party conduct the forensic process removes any chance of challenging the original data that was collected and the methods/procedures taken to collect the data. The neutral third party should involve forensic matter expert(s) that abide(s) by the procedures designated by the courts and has experience in collecting and preserving ESI.

At the same time, having an IT individual conduct analysis of the digital media can lead to vital information being lost or overlooked, that is related to the matter. An IT individual may not know exactly what to look for as they are not trained Forensic Examiners. They may see relevant data as not so relevant or may not consider various means of digital media to be relevant for collection purposes.

Using a third party Forensic Examiner or E-Discovery company, is beneficial because of the experience they bring to the table in conducting analysis and in assisting with managing relevant digital media to be collected/analyzed. The concept here is the same as if you were going to court for a matter. Although a paralegal may know the laws and can essentially guide you through the process, you wouldn't hire them to represent you. You would hire a licensed attorney who has had the experience and specific skill set to represent you in court and be able to follow all the correct policies/procedures mandated by the court.

A perfect example of why self collections are not a good choice can be seen in the case Green v. Blitz U.S.A., (E.D. Tex. Mar. 1, 2011). In this case involving ESI, the company had placed one of their own employees in charge of managing the ESI. This individual was not a forensic expert and did not have experience in managing ESI, which in turn caused relevant data to be lost because no legal holds were issued. Instead, employees were urged to delete email and ESI every so often from there systems.

Relevant data was also left out by the employee who conducted the search, as they had no experience in conducting searches on ESI. This resulted in the court rejecting the data and findings. It cost the company more money to remedy the situation, then it would of cost to hire a third party Forensic Examiner or E-Discovery Company with a Forensics Team to conduct the matter correctly from the

start.

Aside from the more noticeable reasons why you want to have a Forensic Examiner or E-Discovery company with a Forensics Team from the start, here are a few important items to consider. Forensic Examiners have the knowledge and experience working with attorneys or companies to ensure all the relevant digital media is accounted for and collected in a matter. You don't want to under collect the digital media or over collect as well; creating a data set that is either too small or too large. Over collecting data can end up costing a large amount of money when related to E-Discovery terms. Under collecting data, can result in excluding relevant information that is necessary in a matter or exclude digital media that should have been collected regarding a matter.

Forensic Examiners are trained to work hand in hand with attorneys and companies alike, in order to provide the technical support needed for these types of matters. The examiner can assist in identifying the digital media associated with the relevant custodians, (users) involved in a matter by interviewing the custodians and working together with local IT to understand the IT policies and network system.

"THE COMPANIES" IT POLICIES & PROCEDURES

Forensic Examiners or E-Discovery companies with a Forensics Team also have the experience of collecting data in any type of environment. Whether it is for a high profile matter or an "under cover" matter, forensic examiners have the knowledge and ability to adapt to any scenario.

You may be wondering what the relevance is to self collection? IT departments and individuals alike are not trained to deal with these types of scenarios, making it even more difficult to perform a collection correctly. Asking an IT individual to make an "image" of a fellow employee's hard drive creates a problem for the individual. Having the matter stay under the radar may be difficult or may cause the IT individual to make a choice between a "friend" at work and creating the image correctly. That is why using a third party Forensic Examiner or E-Discovery Company with a Forensics Team ensures the matter is handled correctly. Forensic Examiners have been trained to assess a situation and adjust their collection approach as necessary while still staying within the court approved procedures.

CHAIN OF CUSTODY

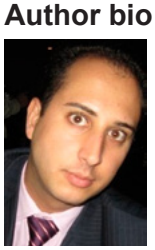
Attempting to perform a "self collection" could result in the original evidence being altered or destroyed if it is not handled properly. If the original evidence is altered in any way, the evidence will not be accepted into court. At the same time if the original evidence is destroyed prior to creating a forensically sound image, vital information could be lost forever. That is why having a Forensic Examiner to handle all the evidence, ensures the chain of custody is being maintained and any interaction with the evidence will not alter or affect the evidence.

Overall the risks associated with "self collection" are heavy in weight. As we discussed throughout this article, the various situations of one performing a "self collection" could run into, as well as the legal ramifications that may follow, as we saw in Green v. Blitz U.S.A., (E.D. Tex. Mar. 1, 2011) can cause much larger problems, then by using a Forensic Examiner or E-Discovery company with a Forensics Team from the beginning.

Having professionals handle these matters ensures that all steps involved follow the governed policies and procedures accepted by the courts. For those that continue to perform self collections or support it, I urge you to do further research of the legal ramifications as well as the risks involved in performing these collections.

For further information or to discuss a possible Forensic Collection, E-Discovery, or Network Security (Penetration Testing) matter, please feel free to reach out to us at Forensic Security Solutions Company. Feel free to contact us through our website at: *www.ForensicSSC.com* or via email at: *contact@forensicssc.com*. Thank you.

Author bio



Elias Psyllos is the Founder/ Managing Director of Forensic Security Solutions Company, a Computer Forensics, E-Discovery, and Network Security consulting firm. Prior to establishing F.S.S.C, he has served as a Forensic Examiner, Sr. Forensic Examiner, and Team Lead in the Corporate and Federal Law Enforcement Agency sides of Computer Forensics. He has conducted digital forensic projects for Fortune 500 corporations, AmLaw 100 law firms, large and medium financial institutions and corporations, non-profits, and law enforcement agencies. He has vast experience with conducting forensic acquisitions on digital media, mobile devices, targeted, multi-user, and small to large scale collections and analysis. Forensic Security Solutions Company is geared toward providing their customers with extraordinary project management and client interfacing that can be utilized for any size matter. Feel to visit us at our website, www.ForensicSSC.com.

F.S.S.C.

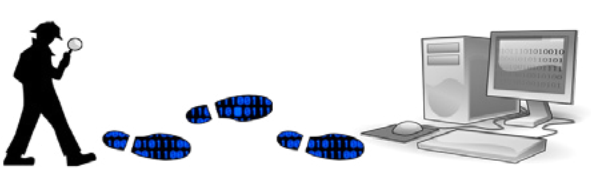
Forensic Security Solutions Co.

A Computer Forensics and Network Security Consulting Co.

- Forensic Imaging & Preservation of Digital Data
- Forensic Analysis & Investigations
- E-Discovery Collections
- Targeted & Multi-User Collections
- Risk & Threat Analysis
- Vulnerability Assessment
- Penetration Testing
- Forensic Wiping of Digital Data Sources (Hard Drives, Thumb Drives, etc.)

Forensic Security Solutions Company is geared toward providing their customers with extraordinary project management and client interfacing that can be utilized for any size matter. Feel free to check us out at www.ForensicSSC.com

F.S.S.C.



Tel: (908) 917-1482 Email: Contact@ForensicSSC.com
www.ForensicSSC.com

Security News

Vulnerability Management

Currently developing hardware and software solutions for the home and business world.

Cybercrime Investigations

Beginner Concepts