

Malware Anaylsis Workshop (Tools and Methodologies)

Tools, Techniques, and Mindset



Introduction

Who, What, Why?

Friday, July 2, 2010

Introduction

- Me Wes Brown
 - Software and Systems Hacker
 - Fond of Lisp-based and Functional Languages
 - Developed Lisp dialect with Scott Dunlop
 - Mosquito Lisp
 - Evolved into Wasp Lisp
 - Security Researcher and Malware Analyst
 - MOSREF uses Mosquito Lisp for a remote command and execution framework
 - Malware Analyst analyzed thousands of samples
 - Security Consultant
 - Penetration Testing
 - Code Review
 - SDL
 - IOActive



The LiveCD: Quick Details

- Bootable Ubuntu Environment
 - Scott Dunlop of IOActive takes much credit for this and refactoring the Malnet code into Malnet 2.
- Includes everything needed for the Workshop
 - Virtual machine, QEMU, X11, analysis tools
- Highly recommended that the environment be booted on *bare hardware*.
 - Running the LiveCD under virtualization will most likely not work very well, crash your computer, or run very slowly.
- You will need *at minimum 1GB of RAM!* The LiveCD uses a ramdisk and running forensics will consume space.
- You will need a real CPU that Linux KVM can run on, supporting virtualization instructions.
- If you don't have a CD drive, see if a neighbor is willing to create a bootable USB key for you using unetbootlin.



Agenda



- Motivations behind Malware Analysis
- Mindset behind Malware and Analysis
- Trends in Malware
- Building a Malware Lab
- Tools for Malware Analysis
- Analysis Walkthrough



Motivations behind Malware and Analysis



- Why Analyze Malware?
 - Better understanding of threats to protect network
 - Defender
 - To write software that detects malware
 - Tools for Defender
 - Aesthetic admiration
 - Admiration of Techniques
 - Writing a better mousetrap
 - Financial Gain
- Why Malware?
 - Financial gain
 - Follow the money
 - Political agenda
 - Used to be for the challenge and pranks



What Makes A Good Malware Analyst?



- Mindset
 - Meticulous data collection
 - Logical processes
 - Thinks outside the box
 - Tenacious
- Technical
 - Good systems understanding
 - Good understanding of programming
 - Some reverse engineering skills
- Attitude
 - Ties into motivations discussed earlier



Why Automate Malware Analysis?



- Too many samples to analyze manually.
 - Recent days have seen 10,000 executables with unique MD5 checksums *per day.*
 - A good malware analyst can only manually analyze a few dozen a day at most; less samples, if they are more complicated.
- Automation ensures consistency of results.
 - Consistent results can be stored in database.
 - Database can be used to search for interesting or relevant malware to analyze.
 - Analysis can all happen from the database.
- Quicker turnaround time on malware analysis.
 - Useful for critical situations where timeliness is vital.
- Malnet 1 was a demonstration of analysis automation.
- Malnet 2 is a more useful malware analyst tool.





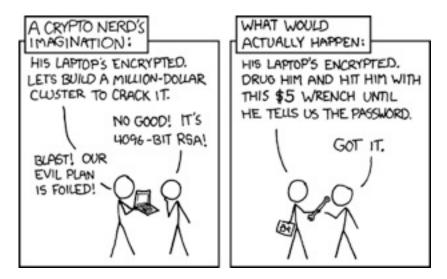
Trends in Malware

Past, Present, and Future

Friday, July 2, 2010

Attack Vectors

- In the Ancient Past
 - Viruses via floppy disks
 - Downloaded via FTP or BBS'es
- Past
 - Systems level
 - Exploitation of remote services, worms
 - System protections an NAT/Firewalls made this difficult
- Now
 - System is only as strong as its weakest link





Human Factor



- In the past, attacks were mainly technical.
 - Attackers searched for network or systems level vulnerabilities.
 - Automatic exploitation and spread.
 - Humans not involved in the attack cycle.
- In the present, exploit the human.
 - Spam email
 - Compromise a legitimate site.
 - "Drive by" site
 - Human visits compromised site, is compromised.
 - Advertising attacks
 - Especially at shadier sites such as P2P trackers.
 - Goal is to get the initial injection vector in.
 - Once vector is in, payload can be sent, and network is compromised.





Attacking through Social Networks

- Social Networks
 - Flickr
 - Facebook
 - Twitter
 - Myspace
 - Etc
- File sharing
 - Torrents
 - Warez
 - P2P
- Highly connected network
- Massive information sharing
- Rich media content



Internationalization of Malware

- Formerly, English-targeted samples.
 - Easy to conduct a strings search on.
- Cultural assumptions of what Malware is.
 - Varies from region to region.
 - One man's anti-cheating toolkit is another man's rootkit.
 - Punkbuster
 - Korean and Chinese games
- What should it be flagged at?
 - Suspicious?
 - White list?
 - Malware?



Current Attack Lifecycle

SOFTWARE

- Initial payload is small
- Initial checks
 - Mutex, OS Version, Keyboard, location
 - Conficker A didn't infect systems with Ukrainian Keyboard
- Payload is downloaded
- Backdoor/trojan/infect
- Contacts command and control server for tasks
- May fall back to secondary C&C
- Dynamically generate rendezvous point





The LiveCD

The Environment

Friday, July 2, 2010

First Steps



- Boot off the LiveCD if you have not already.
- If all goes well, X Windows will start up.
 - If not, sorry, your machine may have a GeForce 9400M
- Customize as you wish to make the environment more comfortable for you.
 - Ubuntu 10.04 based
 - Right click on desktop, 'Package manager'
- Precache the VM image
 - cp /cdrom/winxp.qcow2 .
 - Loads VM image into disk cache for faster execution.



Layout of LiveCD

- User is 'ubuntu'.
 - /cdrom/samples
 - 6 malware samples on the CD.
 - /cdrom/winxp.qcow2
 - Contains winxp VM image.
 - Do not boot this VM image directly!
- Python-based scripts
 - /opt/malnet
 - Scripts are GPL and source code is readable.
 - /opt/volatility
 - Python memory inspection framework.



Kicking the Tires of the LiveCD

- Major components
 - malnet help
 - dumpasm
 - dump-pe
 - dump-reg
 - dump-str
 - eject
 - fork-img
 - inject
 - mount-img
 - run-img
 - volatility
 - memory inspection tools
 - Used to be two main workflows in Malnet 1
 - static forensics
 - dynamic forensics
 - Streamlined and broken out into tasks for more manual control.





Building a Malware Lab

Tools for Analysis

Friday, July 2, 2010

Malware Lab

- Virtualization Platform
 - Multi-core CPUs are cheap
 - Windows images can be reverted in seconds.
 - Can run dozens of Windows images.
 - Easy to audit
 - Use Copy on Write disk images
- Must not be on any network but its own.
 - Airgapped.
 - Prevents inadverent contamination and information leakage.
- Dynamic Internet Connection
 - Preferrably a consumer-level connection.
 - Reissue new IP addresses via DHCP lease.
 - Prevents blacklists against



Virtualization Platform

- VMware
 - Why Vmware?
 - Stable.
 - Well-known.
 - Tools to analyze Vmware suspend images
 - Vmware ESXi is free, bare metal virtualization.
 - Fatal Flaw
 - Lowest common denominator.
 - Malware actively detects Vmware.
 - Virtualization drivers detectable.
 - Easy to detect.
 - » Put value 10 (0x0a) in the ECX register, and put 0x564D5868 in the EAX register. Read a dword from 0x5658.
 - Exploits to break out of Vmware sandbox now.
 - Recommend strongly against using Vmware for a Malware Lab



Virtualization Platform (cont'd)

- Xensource
 - Payware
 - Has a free product to compete with Vmware ESXi
 - Yay competition!
 - Nicely packaged bare-metal virtualizer.
 - Good performance.
 - Excellent Copy-on-Write support
- Qemu
 - Roll your own virtualization platform
 - OpenSource
 - Slower than the others.
 - Components used in KVM and Xensource.
 - Nicely segues into KVM.



Neat Virtualization Tricks



- Serial Debugging
 - Debugger and Debugee VMs with virtual serial connection.
 - Very handy for kernel debugging with tools such as WinDBG.
- Copy on Write
 - Original VM disk image is unmodified.
 - All changes are made to a separate file.
 - Can mount delta images and examine differences to see what malware changed.
- Memory Image
 - State of memory can be snapshotted while malware is run, and then disassembled and debugged.
- Fast reversion of images
 - Useful for analyzing thousands of samples in a day.



Database (aka, store everything!)

- Database
 - Needed to store data from automatic and manual analysis.
 - Malware analysis is far more useful with a corpus to compare against.
 - The more data we have on characteristics, the more we are able to do a determination of whether it is malware.
 - Reverse engineering is expensive in terms of man-power to do.
 - Identify characteristics and understand malware to allocate reverse engineering where it is worthwhile to.
- Corpus
 - Store actual malware sample.
 - Store all known characteristics.
 - Store network traces.
 - Store static forensics.



Obtaining Malware to Analyze



- Be an anti-virus or anti-malware software vendor.
 - Set up your software agent to automatically send back unknown samples.
 - Thousands of samples a day!
- Join an existing antimalware intelligence group.
 - Honeynet Project
 - Emerging Threat's Sandnet
 - Offensive Computing
- Build your own honeynet.
 - Collect malware samples from exploits.
- Beg, borrow, steal.
 - Obtain a feed from someone.
 - Offer a feed in return.



Additional Tools



- Debuggers
 - WinDBG
 - IDA
 - Ollydbg
- Tracers
 - Process Monitor (regmon, filemon)
 - Detours
 - Third party: apimonitor, strace
- Unpackers
 - PelD
 - Import rebuilders





Implementation Details

Decisions Made, and How Things Work

Friday, July 2, 2010

Virtualization Layer - QEMU



- QEMU on the LiveCD
 - About guaranteed to work on everything.
 - Does not require hypervirtualization support in CPU.
 - Supports many VM file formats.
 - Transparent KQEMU support for more speed if host hardware supports it.
- Other Options
 - KVM/Xen
 - Nice segue from QEMU as it uses QEMU components.
 - Requires HT support for Windows guests.
 - Requires extra kernel modules.
 - Vmware
 - Does not work well in a LiveCD environment.
 - Licensing, redistribution.



VM Disk Files – QEMU qcow2



- QEMU qcow2 support
 - Read only block compression
 - qemu-img convert –c –O qcow2 original-file.qcow2 outputfile.qcow2
 - Copy-on-write disk file support.
 - qemu-img create –b baseVM –f qcow2 outputVM
 - All writes to VM happen to this disk file instead.
 - Can be mounted on host OS
 - sudo qemu-nbd –connect=/dev/nbd1 outputVM
 - sudo mount /dev/nbd1p1 /mnt
 - Navigate /mnt
 - Must clean up before running dynamic forensics
 - sudo umount /dev/nbd1p1
 - Sudo qemu-nbd --d /dev/nbd1



Windows VM

- Stripped down VM
 - 156MB to fit on CD
 - Usign QCOW2 block compression
 - ~400MB XP install on a 10GB sparse disk.
- Using nLite
 - <u>http://www.nliteos.com/nlite.html</u>
 - Takes Windows XP and slipstreams service packs and patches
 - Get rid of components you do not need.
 - Creates an ISO image you can use to install.
- XP VM only has SP3, Internet Explorer, and other minimal components.
 - No Flash
- Smaller the VM, the more you can fit on cache, the faster.



Options to Windows VM



- ReactOS
 - http://www.reactos.org
 - Open Source Reimplementation of Windows
 - Theoretically ABI compatible. Theoretically.
 - Most, many malware does not work properly on ReactOS.
 - Good demo for in-place replacement of Windows XP.
- Wine on Linux VM
 - http://www.winehq.org
 - Open Source Reimplementation of Window APIs
 - Works a little better than ReactOS
- Both suffer from loss of forensics accuracy as compared to a real Windows VM



Network Setup



- LiveCD Host OS is connected to the network when cable is plugged in.
- QEMU VMs on the LiveCD by default:
 - Uses the user mode network stack.
 - Simple, straightforward.
 - Setting up TAP is complicated on Ubuntu
- QEMU VMs by default record all network traffic to a pcap file



Automatic Analysis Methodology (1)



- Other implementations (how NOT to do it):
 - Use tools within the Windows VM
 - Deploys samples via network to Windows VM
 - Relies on control and automation software within Windows VM
 - Forensics data on Windows VM to retrieve later
- Detectable! Malware can and will detect debugging and favorite forensics tools.
- If Malware crashes VM, data collection is disrupted.
- Relying on network for forensics collection and malware sample placement means no true isolation.



Automatic Analysis Methodology (2)



- How To Do It
 - Do not rely on any tools running inside VM.
 - Do not use network for malware sample and placement.
 - Do not use command and control within VM.
 - Do not rely on network transfer of resulting forensics from VM.
- Malware sample placement
 - Done using qemu-nbd and mounting the VM disk image.
 - Sample placed in predefined location.
 - Executed upon boot using RunOnce registry key.
- Forensics and data collection take place outside the VM.
 - Memory dump.
 - Disk image journal file.
 - Network capture.





Static Forensics

What Can Be Discovered Without Running It

Friday, July 2, 2010

Static Forensics Basics



- A lot can be determined without ever running the malware sample.
 - Less costly. Static forensics can be conducted in a matter of seconds.
- PE File Forensics
 - Section headers
 - Entropy of sections can be measured.
- Disassembly of Malware
 - Distorm, stream disassembler.
 - IDA Pro in batch mode is better.
- Strings dump of Malware



PE File Forensics



- Executable format for Windows
- Sections
 - Code
 - Data
 - Resources
- Imports Tables
 - What functions are being used?
 - What DLLs are being imported?
 - Look for suspicious functions
- Exports Tables
 - What functions are being used by other programs?



Entropy Metrics



- One key concept in malware analysis is the measurement of entropy.
- Entropy, or information density, is a method for measuring uncertainty in a series of numbers or bytes.
 - In layman's terms, how random is the data stream?
- By measuring entropy, we can determine the data type.

Data	Average Entropy	99% Confidence Intvl
Plain Text	4.347	4.066 - 4.629
Native executables	5.099	4.941 – 5.258
Packed executables	6.801	6.677 - 6.926
Encrypted executables	7.175	7.174 – 7.177

Source: Robert Lyda, James Hamrock, IEEE Xplore, March/ April 2007



Usage of PE Information and Entropy



- By using the combination of entropy metrics and PE file forensics, the analyst can determine:
 - Is it packed?
 - What packer?
 - Is it encrypted?
- Scripts use PE Signatures.
 - /usr/share/pefile/UserDB.txt
- Uses Python pefile library.
 - import pefile
 - pe = pefile.PE('samles/sample')
 - pe.dump_info()



Usage of PE Imports



- By looking at the calls that the malware makes, we can flag the malware for suspicious activity.
 - RegSetValue
 - CreateUser calls
- Userland applications typically do not need to make system level calls.
- However, it is obfuscated by packers. We will need to run dynamic forensics to see what actual calls are happening.



Malnet 2 Static Forensics Commands

- malnet dump-pe
 - performs analysis of pefile information
- dump-str
 - scans the sample file for sequences of printable characters
- dump-asm
 - attempts to disassemble the sample executable



Unimplemented: Signed Executables



- Looking at the signature, description, and publisher on an executable is a key part of static forensics.
- Malware often masquerades as prominent publisher updates or executables.
 - Microsoft.
- By building a corpus of valid white-listed files, we can use a database to compare samples against.
- For example, if malware masquerades as a DirectX installer; is it signed? What is the publisher, version field? Do we have valid installers of the same publisher?
- Unimplemented due to no easy native Linux tools to examine this data.

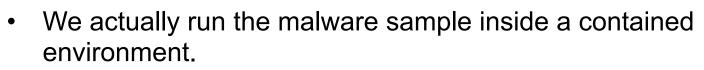




Dynamic Forensics

Actually Running the Malware Sample

Dynamic Forensics Basics



- Run inside QEMU VM.
 - Screenshots
 - Memory dump.
 - Copy on write file.
- Examine changes:
 - Registry dump.
 - Copy on write file.
 - Network packet capture.



QEMU Monitor Commands

- Execute QEMU with:
 - -monitor stdio
- Screendump command
 - screendump filename
 - PPM file output
- Memorydump command
 - pmemsave 0 0x08000000 filename
 - Physical memory dump
- Snapshot commands
 - savevm tag
 - loadvm tag
- GDB server
 - gdbserver port



Network capture

- Wireshark on LiveCD
 - wireshark pcapfile.
- QEMU has built in packet capture to pcap.
- chaosreader.pl on LiveCD
 - Breaks out according to sessions.
 - chaosreader pcapfile
- Network forensics on the LiveCD is an example implementation.
 - No honeynet.
 - No firewalling.
 - No protection.
 - Pants down!



Screendumps



- Dynamic forensics process dumps screenshots.
- By using the md5sum of the screenshots, we can determine:
 - If anything showed up on the Windows screen.
 - Build a library of signatures based on screenshots.
 - Often malware has the same net result on the screen, but different MD5sums for the executable files to evade detection.
- Make sure dynamic screen elements like the clock is turned off!



Memory Dump



- Caution
 - In the limited LiveCD environment, it will very quickly consume all available RAM with 128MB per memory dump.
- Dump of all VM 'physical' memory.
 - Pagefile is disabled to ensure that memory used is physical memory.
- Can do various techniques on memory dumps.
 - See Nguyen Anh Qunyh's eKimono.
 - virtuality Python framework included on LiveCD.



Registry Dumps



- We do registry dumps using dump-reg before and after a run.
- We can see what was changed in the registry.
 - We have no filtering capability on the LiveCD to screen out registry changes done by the course of normal Windows operations.
- Utility used is 'dumphive'
 - /usr/bin/dumphive
 - Pascal based, depreciated, but works well.



Copy on Write File



- Copy on Write file is generated during the course of dynamic forensics.
 - Journal of all changes that would have been written to the original VM disk image.
 - Can be mounted and examined after the dynamic run.
 - Much more portable and flexible than VM snapshots.
 - Much smaller so can be archived along with other malware data.
- Neat trick ---
 - Run strings on qcow2 file.
 - Will display changes in chronological order.



Malnet Dynamic Forensics Commands

- dump-reg
 - decodes registry hives in a disk image to a text file
- inject
 - transfers the sample executable into an image
- fork-img
 - derives a new Copy-on-Write image from the baseline
- run-img
 - Boots QEMU using the suppled image file





Memory Forensics

Using Volatility

Introduction to Volatility

- Open source memory forensics framework.
- Crucial part of Malnet 2.
 - We use QEMU's memory dump feature.
 - We examine memory after the fact.
- Volatility can show:
 - running processes
 - open network sockets
 - open network connections
 - DLLs loaded
 - Dump process memory
 - Show registry handles for each process
 - Show kernel modules



Fly in Amber

- By having memory dumps, we have running malware processes and connections frozen in time.
- This can be archived for the future, even if the malware no longer can run or the command and control servers are down.
- Memory images can be compressed, and even diff'ed against the same running system image to reduce the archival footprint.





Exercises

Interesting Samples To Look At



Future Directions

What Could Be Improved

Version 2.0 Product



- This is a more refined product than v1.0.
 - Not freely redistributable.
 - V2.1 or later will likely be freely distributable and downloadable from the IOActive site.
- Many improvements can be made, especially in the area of dynamic and static forensics.
 - Honeynet.
 - Restricted Internet access allowed for better data collection.
 - Automatically visiting websites to make the malware wake up
- Alternative VMs to execute malware within
 - ReactOS
 - Minimal Linux + WINE.
- Better VM interactivity
 - Hotkeys to dump screenshots and memory dump on demand.



Contacting The Guilty Parties

- Wes Brown
 - Malware analyst, hacker, programmer.
 - wbrown@ioactive.com
 - Twitter: w3sbrown
 - AIM: wesbrown18
- Scott Dunlop
 - Systems engineer, hacker, programmer.
 - sdunlop@ioactive.com
 - Twitter: swdunlop





Thank You!

Wes Brown