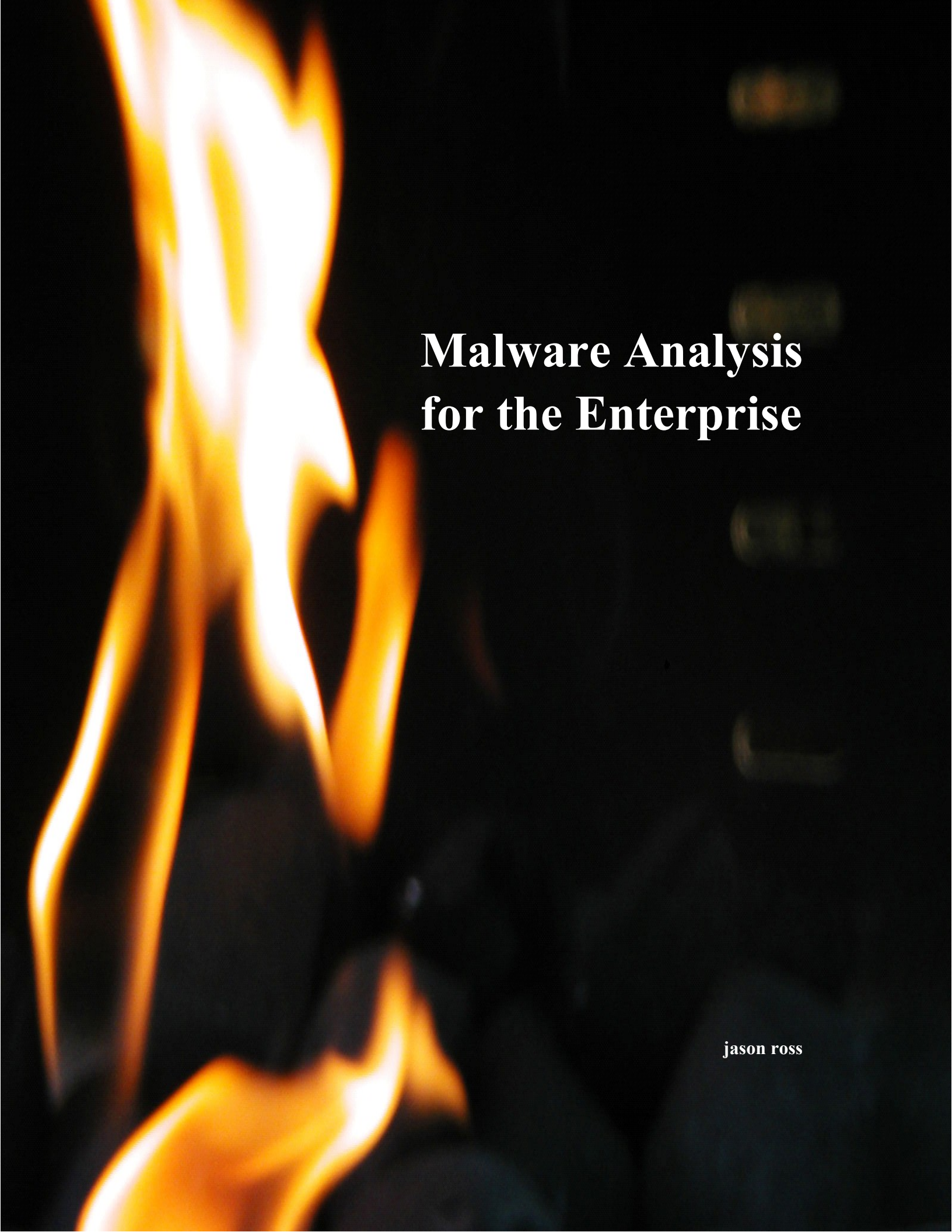

Malware Analysis for the Enterprise

jason ross

Table of Contents

Introduction

In a typical organization, an attack from malicious software (known as *malware*) is not likely to go completely unnoticed. Detection of an attack may come through one or more technologies such as antivirus software, intrusion detection systems, or it may come from systems compliance monitoring.

Unfortunately, detection of the attack is no longer sufficient to identify the full risk posed by malware. Often, detection occurs after the host has already been compromised. As malware evolves and grows increasingly complex, it is utilizing self-defense mechanisms such as root kit technologies to hide processes from the kernel, disable antivirus software, and block access to security vendor websites and operating system update information.

Faced with these threats, once a host's integrity becomes compromised a crucial part of the incident response process is to determine what activity the malicious code is engaged in, and specifically whether any data may have been compromised and to where it may have been sent.

How Does Malware Analysis Help?

The Need For Analysis

The only way to really determine what a piece of malicious software is doing is to analyze it. The anti-virus industry has researchers who do this as a key part of their business. In the past this was sufficient, because the motivating factor behind viruses was largely fame. Because of this, viruses were generally written by single individuals, and were designed to infect as many machines as possible. As a result, once a researcher was made aware of a threat, they could analyze it and create signatures which could be pushed out by the anti-virus vendor, to protect everyone in the same fashion as they had been infected, en masse.

Additionally, at this time malware was generally not very complex, in part because the authors didn't have the resources needed to create very complicated programs. This relative simplicity meant that an infected host could usually be cleaned with a high chance of success. While there were some truly devastating viruses, they were not very common.

Times have changed (it's a business, not a kiddie)

Where malware was once being designed for fun, research, fame, or even to promote socio-economic activist ideals, today creating and managing malicious software has become a solid business model for criminals, and it is part of a robust underground economy.

Because it's a business product, malware today has completely different goals than in the past. To meet these new goals, the complexity of the code, as well as the infection process, has increased substantially. Additionally, we are seeing the delivery pattern of malware change to meet the needs of the clients being served by the malware industry. For example, rather than infecting as many computers as possible, a piece of malware may be limited to perhaps a few selected computers within a specific organization. At the same time, another piece of malware may be served via a legitimate web site which has been compromised[1], to as many people as visit the site. These differences in methods exist to meet the needs of clients with different goals.

Because of the shift in the way malware is being developed and deployed, the methods used to mitigate the threat of malicious software in the past are no longer effective. Malcode which affects an organization may exist solely within that organization. It is not effective in cost, nor codebase, for the anti-virus industry to manage a vast number of "one off" signatures, yet this is what is being done. Further, as the industry expands and resources are added which are needed to compete with the business of malware creation and distribution, problems arise as a result of the fact that there are no standards for malware management. For example, when an organization becomes aware of the fact that a host on their network has been compromised by a piece of malware, it is often necessary for them to learn more about what the malware does, and how to remove it from the infected host.

This process is made very difficult and confusing when each vendor has differing information about the malware. Adding to the challenge is the fact that often the same piece of malicious software will have multiple names, as each vendor picks their own way to uniquely identify it.

The signature arms race

Anti-virus products work by creating a binary signature of a piece of malicious software. If a file on the system matches the signature, it is determined to contain that malicious software, and is dealt with according to the policies that have been set up either by the vendor, or the organization deploying the product. This works reasonably well when the quantity of unique malicious software is relatively small; however, this method does not scale well. As the number of unique

1 Websense released their 2009 First Quarter *State of the Internet* report with dismal statistics of mass ownage. They reported a 671% growth in malicious web sites in the past year, 77% of which were legitimate sites that had been compromised.(*Websense Security Labs Report - State of Internet Security Q1 - Q2 2009)*

samples grows, managing the signatures required to identify them becomes problematic. Further, if the number of unique samples increases at a significant rate, the amount of time a particular piece of malcode is able to remain undetected increases as well, as the resources required to develop new signatures often do not increase to keep up with the influx. This leads to what is essentially a signature arms race, where the authors of malware take advantage of the time between their software being deployed, and the time it takes the anti-virus industry to analyze it and develop a signature pattern. In an effort to come out on top of this race, the industry has developed heuristic detection, which is used to categorize and group classes of malicious activity. While this does help, it is not sufficient to catch all malicious activity. Further, to prevent heuristics from working effectively, malware is deploying in multiple stages. Since heuristics watches for specific types of activity being performed by an executable, the compromise has been broken down into several steps, making it possible for a machine to be at least partially compromised without the anti-virus product detecting it. Apart from heuristics, many anti-virus products have taken to identifying any software which is packed as being malicious. Since many legitimate software packages also use packers to decrease the size of their programs, this fosters complacency in the end user as the number of false positives increases.

Further complicating this situation is the fact that since malware has become a business product, it now comes with a support model. Often included in this support is a guarantee that a given piece of malware will remain undetectable. Should an anti-virus product create a signature to detect the malware successfully, the author will alter the binary such that it no longer matches the signature – for the life of the support plan. This is accomplished in a number of ways:

- creating routines which encrypt the code using strong cryptographic ciphers, and randomized keys
- completely altering the codebase itself in an automated fashion by using polymorphic routines
- packing[2] and compressing the executables

Each of the above methods alters the resulting binary in ways that make it difficult to analyze, let alone create a single signature pattern for it.

As a result researchers are flooded with samples which may all be the same piece of malcode, but because each one has different properties, they require different signatures. Online resources such as Virus Total assist malware authors in this process by allowing them to easily determine the detection rate of their malicious binary.

For all of these reasons, there is a need for malware analysis to become part of an organization's standard security practice[3], and just not something that is relegated to highly skilled technicians employed by the anti-virus industry and researchers alone.

Where Does Malware Analysis Fit In?

If analyzing malware is to be an essential component of an organization's security posture, it's important to understand how it relates to the process and policies already in place at that organization.

Infection is an incident

Because malware has been part of the computer security threat landscape for so long, and due to the media attention given to high profile attacks, viruses have become common. As a result, malware is often not seen for the serious risk it poses. The quirky names often given to viruses, (such as *Slammer*, *Melissa*, or of course *I Love You*), exacerbate this tendency to trivialize an

2 In 2007, Panda software released a study which stated that 78% of of new malware at that time used some form of file packer (*Panda Software: Packing malware, growing threat 6/5/2007)*

3 This need is reflected in job postings. Monster.com shows 86 positions open as of October 21, 2009 that contain *malware* as a keyword hit, and 27 which specifically are looking for *malware analysis*. The majority of these positions are for industries outside of the information technology sector.

infected host as a nuisance rather than a true security threat. Thus, despite the fact that the infection process and purpose of malware have significantly changed, the response to infection and compromise has essentially remained the same: identify, create a signature, and clean. As a result, infection handling is generally left out of the incident response policies. This is a mistake.

Malware is typically deployed in a multi-stage process, the end result of which is frequently complete control of the victim host by an attacker. This means that each alert from an anti-virus product could in fact be notifying you of the fact that you've now got a hostile host on your network. Worse, the attacker using the host is using whatever credentials are available, which means the theoretical "malicious insider" problem has just become real, only the insider isn't Bob from Accounting, it's a hostile foreign entity that now owns Bob's computer and is sending data from it to some other compromised host they control over an encrypted tunnel. Accordingly, how an organization deals with infected hosts has a number of implications. For example, if the infected host (or the end user the host belongs to) accesses sensitive information, there could be a number of legal and compliance problems that arise, including notification to customers that their data may have been compromised[4].

In light of this, it makes sense that malware which is discovered on the organization's network needs to be analyzed to minimally determine the following:

- Was the host successfully compromised?
- If it was, how was it compromised?
- What occurred after the compromise?
- Was any data taken?
- If data was taken, where was it sent?
- Were any other hosts compromised as well?

These are all questions that an organization needs to be able to answer so they can determine how to form a proper response to the incident. Many of the answers can be obtained by analyzing the malware. As such, malware analysis belongs in an organization's incident response policies and procedures.

4 Section 13402 of the *HITECH Act* requires HIPAA covered entities to "notify affected individuals… following the discovery of a breach of unsecured protected health information". See the HITECH Act Breach Notification Guidance. If an infected host under the control of a botherder accesses such information, it should rightly be considered a breach.

How Does Malware Today Work?

In April 2009, FireEye published[5] an excellent report which showed (among other things) the inter-relationship between malware families and various botnets (see Figure 2: Complicated Inter-relationship of Botnet Webs.)

The report demonstrated in a very clear way that malware is extremely complex and inter-related, as the image in Figure 2: Complicated Inter-relationship of Botnet Webs demonstrates.

Based on information in the report, it is apparent that malcode authors and botherders are collaborating with each other. Because of this, thinking of an infected host in terms of single virus infections is not accurate, and does not reflect the complexity of the true landscape.

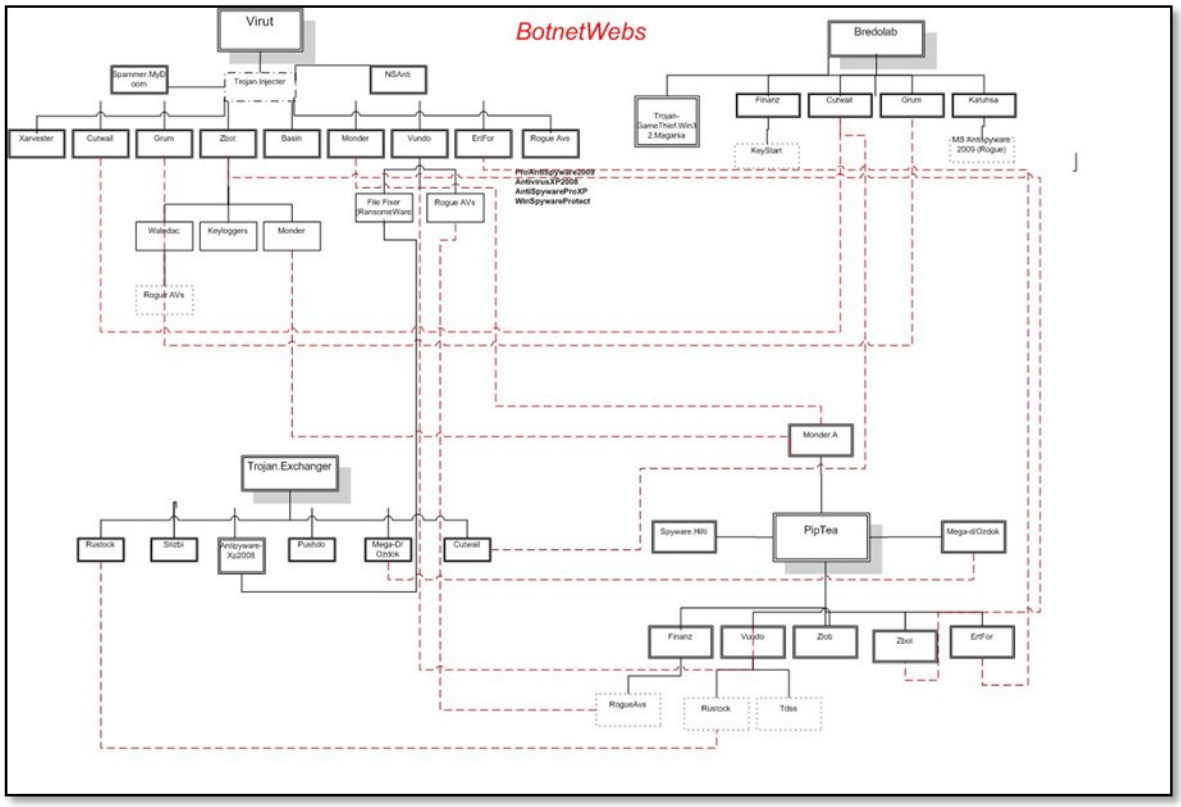

Figure 2: Complicated Inter-relationship of Botnet Webs

Droppers and Downloaders and Rootkits Oh My!

One of the more common methods being used to spread malicious software is compromised web sites. A link to the malware is placed on websites which are either owned by the attackers, or are compromised legitimate sites. These executables are generally not detectable by anti-virus software, and have no other purpose than to get loaded onto the victim and begin the second wave of attack, which generally consists of using HTTP to retrieve additional malicious software.
The second stage malware handles things like disabling anti-virus software, manipulating host based firewall rule sets, installing root kits to hide malicious activity from the OS, and in some cases inserting code into the boot sector of the hard drive to allow it to remain in place even if the

5 FireEye blog post: *BotnetWeb: A Collection of Heterogeneous Botnets..*

OS is cleaned.

It's increasingly the case that more than one type of virus is utilized at this step in an effort to ensure successful compromise. If the anti-virus software triggers, it is usually at this point, however, by this time it is already too late, as the host has already been compromised successfully. While the anti-virus may have caught one of the new malware installation attempts, it is quite likely that there were others.

How can you say you're clean if you can't trust the OS?

If the malware was successful in root kit installation, any investigative work being done at this stage is useless, as any information reported by the operating system kernel is suspect. Often security professionals will respond to an anti-virus alert that indicates a host was compromised by doing the following:

- Login to the host to investigate
- Viewing the processes running on the system
- Check open network connections

If "nothing unusual" is found, often the decision is made that the host is clean, and the anti-virus software did its job. The problem with this is that once a rootkit is installed, nothing the kernel tells you can be trusted. The author of the malware can use the rootkit technology to hide processes, registry entries, network connections, directories, etc.

For this reason it is becoming the recommended practice that if a host becomes infected, it should be wiped and reinstalled from scratch. However, even this is not enough, as the use of boot sector rootkits is growing. To ensure the system is no longer infected, it is necessary to format the boot sector of the hard drive as well.

Playing With Fire (How To Analyze Malware)

There are two techniques which can be used to perform an analysis on a piece of software to understand what it does:

- *Static (Source Code) Analysis* – Analyzing the source of the malware. Typically this involves reverse engineering the binary executable. This can be problematic in some countries due to overly restrictive laws regarding software.

- *Runtime (Behavioural) Analysis* - Observation of network traffic and any changes made to the operating system environment as the executable runs. This method is riskier than static analysis due to the fact that the host is intentionally compromised during the process.

Static analysis

The first method, known as static analysis, requires special skill sets, including an extensive understanding of assembler (usually for the x86 chipset), software debugging techniques, and increasingly a solid understanding of encryption methods. Typically this level of skill means hiring a specialist, so reversing malware has generally been left to the anti-virus companies and various security research labs. As malware becomes increasingly advanced however, industries with high need for security (such as the aerospace or pharmaceutical industries for example) are beginning to employ in-house researchers with these skills.

Runtime analysis

The second technique, referred to as run time analysis, has a significantly lower cost associated with it since it does not specifically require a need for a specialist. This process involves gaining a solid understanding of what a piece of software does from simply observing the system prior to, during, and after it has been successfully run. The experience needed to perform these tasks may already be available within the IT staff of an organization. It is for this type of analysis that a sandnet is used.

What is a sandnet?

Many organizations are familiar with the concept of a sandbox, or testing, host. Such systems are often used to isolate new code, server software, or even new operating systems, from the production environment or network. As its name implies, a sandnet expands this concept beyond a single host, to an entire network dedicated to testing and analysis. Specifically, a sandnet used to analyze malicious software provides a virtual Internet, within which all traffic generated and any actions taken by the malware sample that is undergoing analysis can be logged and examined.

Virtual Machines vs. Bare Metal

The first factor an organization must consider when setting up a sandnet is whether to use physical or virtual machines for the purpose. Many features of virtualized environments are ideal for the tasks a sandnet requires, because the analyst is able to use technologies such as cloning to easily create victim and services hosts as needed. Further, with the use of snapshots, it is a fairly simple process to boot up a clean virtual host, analyze a given piece of malware, and then restore the environment to its initial state once the evaluation has been completed. Additionally, depending on the virtual machine technology being used, there may be other features available which are useful for analysis, some of which are discussed below. Given these advantages, it seems a natural choice to use a virtualized environment to perform malware analysis.

There are a few reasons that a virtual host may be undesirable as the analysis platform however, one of the most important being that the malware being analyzed may be checking to see if it is being run in a virtual machine.

Smart malware authors check for VM

A key factor to consider when determining whether to use a virtual host or a bare metal machine as a victim is the fact that malware is increasingly utilizing mechanisms to determine whether or not it is being run inside a virtual machine. There are a number of techniques that can be used to do this. These range from something as simple as checking to see if the hard drive volume name, or network card MAC matches default virtual machine settings, to more esoteric solutions[6] involving differences in the way the kernel handles functions inside a virtual environment, etc.

Dumb malware authors also check for VM

As the malicious software industry grows, a number of design tools have been created to assist in creating malware. Some of these are quite advanced, and rival (or even exceed at times) commercial software design tools in the quality of the user interface. This means that enabling a malicious software executable to perform virtual machine detection may literally be as easy as clicking a checkbox (see Figure 2: SharK 3.1 Anti-Debugging Features below).

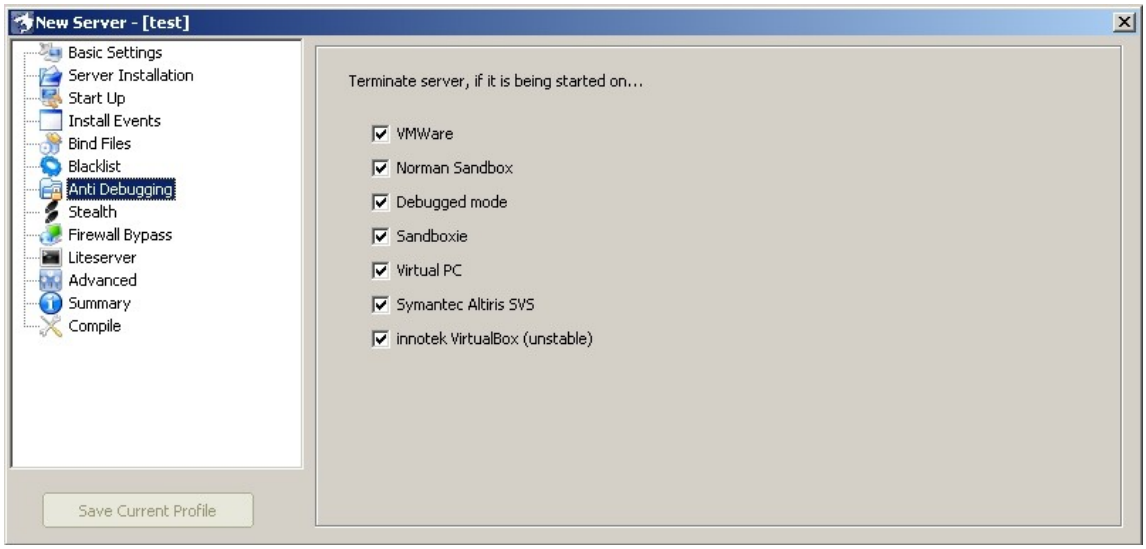

Figure 2: SharK 3.1 Anti-Debugging Features

These malware builder kits are not terribly difficult to find on the internet, and can often be acquired freely. As a result, malware authors who are not highly skilled are able to create executables with advanced features.

Despite the fact that malware checking for virtualized environments, it is still very often the case that an organization prefers to use a VM for analyzing malware, usually due to the cost benefit this strategy provides. As a result, the remainder of this document will focus on using a virtual environment for analysis. Some of the processes used herein may differ slightly for a bare metal lab, but in general they are largely the same. Because Sun's VirtualBox product often escapes notice from malware authors, it was chosen as the platform to be used.

Setting Up The Sandnet

At a minimum, a sandnet should have two hosts, one to provide services such as DNS, HTTP, and monitoring capabilities, and one to serve as a victim host, upon which the malware sample will be run.

For the purposes of this document, the services host was set up with the Debian distribution of

6 For examples of these refer to Joanna Rutkowska's RedPill, and Tobias Klein's Scoopy NG

Linux. The victim host OS is Windows XP Service Pack 3. No patches other than the service pack were installed.

The output below shows the result of list listing the virtual machines which have been set up. The services host has been named *linux*. The victim host has been named *winxp_sp3_01*.

```
> VBoxManage list vms
VirtualBox Command Line Management Interface Version 3.0.8
(C) 2005-2009 Sun Microsystems, Inc.
All rights reserved.

"linux" {ad59f194-585e-49c5-a54c-5e92322b1188}
"winxp_sp3_01" {7a554f4e-6aea-42f1-a3c5-488d43f161ff}
```

Network configuration

The network traffic generated by machines in the sandnet should isolated from any production network, including the public Internet. In a bare metal lab, this could be accomplished with the use of firewalls, or simply by not connecting up the hub, switch, or router used by the lab to any other equipment.

In the virtual environment, each machine should be configured to use the *Internal Network* option. This can be verified by using the VBoxManage utility as follows: (some output has been removed to preserve clarity):

```
>VBoxManage showvminfo winxp_sp3_01

Name:            winxp_sp3_01
Guest OS:        Windows XP
UUID:            7a554f4e-6aea-42f1-a3c5-488d43f161ff
Memory size:     512MB
VRAM size:       12MB
Number of CPUs:  1
NIC 1:           MAC: 080027D32767, Attachment: Internal Network
'intnet',

> VBoxManage showvminfo linux

Name:            linux
Guest OS:        Debian
UUID:            ad59f194-585e-49c5-a54c-5e92322b1188
Memory size:     256MB
VRAM size:       12MB
Number of CPUs:  1
NIC 1:           MAC: 080027355D36, Attachment: Internal Network
'intnet',
```

Note that the Internal Network has been assigned the name 'intnet'.

If you wish to enable DHCP for the sandnet, you can use the VBoxManage tool to set this up. When you do this, you'll need to specify the network name you want the DHCP server to respond to (this can be determined using the showvminfo as demonstrated above), an IP address for the server, as well as the lower and upper IP addresses in the DHCP pool:

```
> VBoxManage dhcpserver add --netname intnet --ip 192.168.3.1
--netmask 255.255.255.0 --lowerip 192.168.3.100 --upperip
192.168.3.250 --enable
```

Monitoring and logging traffic

The VirtualBox VM environment has the capability to dump all network traffic to and from a given VM into a pcap file. This is extremely useful when analyzing malware, as using other means to capture the traffic from a virtual machine can be problematic. To enable this feature, perform the following steps:

```
> VBoxManage modifyvm vmname –nictraceNIC# on –nictracefileNIC#
filename.pcap
```

Note that there is **no** space between the *–nictrace* and *–nictracefile* command the the number of the NIC you wish to use. For example, to capture the traffic to and from the *linux* virtual machine using the first network adapter (which shows up as "NIC 1" when running the showvminfo command) the following command could be used:

```
> VBoxManage modifyvm linux –nictrace1 on –nictracefile1
"C:\Users\Test\linux.pcap"
```

Once you've finished the analysis, you can disable the trace option by using the same command and specifying 'off':

```
> VBoxManage modifyvm linux –nictrace1 off –nictracefile1
"C:\Users\Test\linux.pcap"
```

Services Host Setup

OS Configuration

For the services host, the Debian distribution of Linux was chosen. The system was installed using the netinst[7] image, and using only the 'standard system' package group during the installation process. Once the system installation is complete, some additional software should be installed:

```
# apt-get install php5 bind9 openssh-server tcpdump tshark postfix
build-essential
```

When postfix asks about the type of site, pick "Internet Site" and accept the default mail name (which will be the hostname selected during the OS install).

DNS Service (ISC Bind 9)

BIND has been configured such that it is the SOA for every domain request that it receives, and it will reply to any requests with the IP address of the services host. It is further setup to provide the address of the services host as the MX for any domain that is requested. This configuration allows any DNS calls being made by the victim hosts to be observed, and any software on the victims which tries to communicate with the internet is instead directed to the services host, so that the content of the communication can be analyzed. There are a number of files which must be configured to set this up.

named.conf

```
include "/etc/bind/named.conf.options";

key "dnskey" {
   algorithm hmac-md5;
   secret "hash";
};

controls {
   inet * allow { 127.0.0.1; } keys { "dnskey"; };
};

zone "." IN {
   type master;
   file "/etc/bind/db.wildcard";
};
```

named.conf.options

```
options {
   directory "/var/cache/bind";
   allow-transfer { none; };
   logging {
      channel query_log {
         severity info;
         print-time yes;
```

7 Available at http://www.debian.org/distrib/netinst

```
         file "query.log" versions 5 size 50M;
      };
      category queries {
         query_log;
      };
   };
   listen-on-v6 { any; };
};
```

wildcard zone configuration (db.wildcard)

```
$TTL   604800
@   IN   SOA   localhost.  root.localhost. (

                         2009102201  ; serial

                             604800  ; refresh

                              86400  ; retry

                            2419200  ; expire

                             604800) ; negative cache ttl


@               IN          NS          localhost.

*               IN          MX 10       192.168.3.101

*               IN           A          192.168.3.101
```

Web Service (Apache 2)

When using Apache to analyze malware, it is useful to have a more robust logging mechanism than that which is used by default. Fortunately, Apache2 comes with a few options that fit the need, one of which is mod_log_forensic. This module comes preinstalled in Debian, but it must be enabled. This can be done using the a2enmod utility as follows:

```
# a2enmod log_forensic
```

To configure the web server up such that it utilizes the functionality of this module, the following line must be added to the server configuration file[8]:

```
ForensicLog    /var/log/apache2/forensic_log
```

Once the module is enabled and configured, Apache can be restarted to activate the changes:

```
# apache2ctl reload
```

8 This can be found in */etc/apache2/sites-available/default* on Debian

Now when an HTTP request is made to the web server, detailed information about that request will be logged to the file specified in the server configuration. An example of the data that is recorded can be seen below:

```
+2021:4adf8568:0|GET / HTTP/1.1|Accept:*/*|Accept-Language:en-us|
Accept-Encoding:gzip, deflate|User-Agent:Mozilla/4.0 (compatible;
MSIE 6.0; Windows NT 5.1; SV1)|Host:192.168.3.101|Connection:Keep-
Alive|Cache-Control:no-cache
-2021:4adf8568:0
```

SMTP Service (Postfix)

Edit /etc/postfix/main.cf to setup virtual mail hosting by appending the following to the end of the file:

```
virtual_alias_maps = hash:/etc/postfix/vmail_aliases
virtual_mailbox_domains = *
virtual_mailbox_base = /var/spool/vmail
virtual_mailbox_maps = hash:/etc/postfix/vmail_maps
virtual_minimum_uid = 500
virtual_uid_maps = static:500
virtual_gid_maps = static:500
```

Obtain the *nocando* shell from the Team Cymru website (this shell is "a denial shell that provides logging when users with this shell attempt to gain access"). The source code will need to be compiled, and the resulting binary should be installed into a system directory:

```
# cd /usr/local/src/
# wget http://www.cymru.com/Tools/nocando-1.4.tar.gz
# tar zxvf nocando-1.4.tar.gz
# cd nocando-1.4
# make
# cp /nocando /usr/local/bin/
```

Create a virtual email user and group (*vmail* in the examples below. The UID and GID need to match the settings used in the Postfix main.cf):

```
# groupadd –g 500 vmail
# useradd –g 500 –u 500 –s /usr/local/bin/nocando –d /var/spool/vmail
vmail
```

Create the mail spool for the virtual mail domains and set the permissions on the vmail directory structure such that the vmail user has access to the directories and files::

```
# mkdir –p /var/spool/vmail/spamcan
# chown –R 500:500 /var/spool/vmail
# chmod 0755 /var/spool/vmail
# chmod 2750 /var/spool/vmail/spamcan
```

Create the virtual mail mappings and load them into postfix editing /etc/postfix/vmail_maps to include the following:

```
@ spamcan/
```

Create an empty virtual alias map, then generate the postfix hashes for each of the virtual mapping files. Once that's done, reload the postfix configuration:

```
# touch /etc/postfix/vmail_aliases
# postmap /etc/postfix/vmail_aliases
# postmap /etc/postfix/vmail_maps
# postfix reload
```

Generic Listener Service (Netcat)

If a piece of malware is attempting to send traffic of an unknown nature to a remote host, it may be useful to capture that information by setting up netcat to listen on the port being used by the malware and dumping any incoming data to a text file. For example, to set up a netcat listener on TCP port 8080 and log any input received to a file, the following command could be used:

```
# netcat –nvlp 8080 –o tcp_8080.txt
```

An example of the contents of the tcp_8080.txt file after receiving a request from Internet Explorer on the victim host to the listening netcat service is shown below:

```
< 00000000 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a # GET /
HTTP/1.1..
< 00000010 41 63 63 65 70 74 3a 20 69 6d 61 67 65 2f 67 69 # Accept:
image/gi
< 00000020 66 2c 20 69 6d 61 67 65 2f 78 2d 78 62 69 74 6d # f,
image/x-xbitm
< 00000030 61 70 2c 20 69 6d 61 67 65 2f 6a 70 65 67 2c 20 # ap,
image/jpeg,
< 00000040 69 6d 61 67 65 2f 70 6a 70 65 67 2c 20 61 70 70 #
image/pjpeg, app
< 00000050 6c 69 63 61 74 69 6f 6e 2f 78 2d 73 68 6f 63 6b #
lication/x-shock
< 00000060 77 61 76 65 2d 66 6c 61 73 68 2c 20 2a 2f 2a 0d # wave-
flash, */*.
< 00000070 0a 41 63 63 65 70 74 2d 4c 61 6e 67 75 61 67 65 # .Accept-
Language
< 00000080 3a 20 65 6e 2d 75 73 0d 0a 41 63 63 65 70 74 2d # : en-
us..Accept-
< 00000090 45 6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 #
Encoding: gzip,
< 000000a0 64 65 66 6c 61 74 65 0d 0a 55 73 65 72 2d 41 67 #
deflate..User-Ag
< 000000b0 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 34 2e 30 # ent:
Mozilla/4.0
< 000000c0 20 28 63 6f 6d 70 61 74 69 62 6c 65 3b 20 4d 53 #
(compatible; MS
< 000000d0 49 45 20 36 2e 30 3b 20 57 69 6e 64 6f 77 73 20 # IE 6.0;
Windows
< 000000e0 4e 54 20 35 2e 31 3b 20 53 56 31 29 0d 0a 48 6f # NT 5.1;
SV1)..Ho
```

```
< 000000f0 73 74 3a 20 31 39 32 2e 31 36 38 2e 33 2e 31 30 # st:
192.168.3.10
< 00000100 31 3a 38 30 38 30 0d 0a 43 6f 6e 6e 65 63 74 69 #
1:8080..Connecti
< 00000110 6f 6e 3a 20 4b 65 65 70 2d 41 6c 69 76 65 0d 0a # on:
Keep-Alive..
< 00000120 0d 0a                                           # ..
```

If the malware is using UDP for communication, you can accommodate that by using the *–u* option netcat provides.

A quick note about javascript obfuscation

When analyzing malware, it is likely that an analyst will discover javascript code which has been obfuscated. Before further analysis can be performed, it becomes necessary to de-obfuscate this code. To perform the task, there are a couple different methods which can be used[9], one fairly easy and quite effective way to do this is to use the SpiderMonkey javascript engine from Mozilla. Didier Stevens, a respected malware researcher, has added some functionality to the main codebase which is particularly handy for malware analysis[10].

Due to the growing number of malware samples which include javascript (examples could be drive-by downloads injected into a web site, or malicious code that has been added to an Adobe PDF document), it is a good idea for an organization wishing to perform malware analysis to become familiar with this process. However, because this topic strays from run-time analysis and begins to address source code analysis and reverse engineering, it is beyond the scope of this document to provide coverage of how to perform these tasks.

9 A very nice summary on this process can be found in the *Decoding Javascript Roundup* SANS ISC diary entry.
10 See Didier's blog post for specific information on the changes he made, as well as the source code.

Victim Host Setup

OS Configuration

When it comes to setting up a victim host to analyze malware, there is a nearly infinite number of configuration choices. However, when the goal is to determine the effect a given piece of malicious software has to an organization, it makes sense that the victim machine should emulate as closely as possible the machine upon which the malware was discovered.

Analysis Software

To perform the run time analysis of the malware, it will be necessary to install a wide range of tools and applications. The ones that will be useful differ based on what the malware being analyzed does. Some may be rendered useless as the malware attempts to disable security software. A brief list of some of the more useful software is presented below. Where possible, the description of the tool is taken directly from the website the tool is located at:

Name	Description	Website
AutoRuns	This utility, which has the most comprehensive knowledge of auto-starting locations of any startup monitor, shows you what programs are configured to run during system bootup or login, and shows you the entries in the order Windows processes them.	http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx
BinText	A small, very fast and powerful text extractor that will be of particular interest to programmers. It can extract text from any kind of file and includes the ability to find plain ASCII text, Unicode (double byte ANSI) text and Resource strings, providing useful information for each item in the optional "advanced" view mode.	http://www.foundstone.com/us/resources/proddesc/bintext.htm
CaptureBat	Capture BAT is a behavioral analysis tool of applications for the Win32 operating system family. Capture BAT is able to monitor the state of a system during the execution of applications and processing of documents, which provides an analyst with insights on how the software operates even if no source code is available	https://www.honeynet.org/node/315
GMER	GMER is an application that detects and removes rootkits .	http://www.gmer.net
IceSword	IceSword has a Windows Explorer-like interface but displays hidden processes and resources that Windows Explorer would never show. It isn't a "click-here-to-delete-rootkits" product but a sophisticated discovery tool that can protect against sinister rootkits if used before they infect a machine.	http://www.antirootkit.com/software/IceSword.htm
LordPE	LordPE is a tool e.g. for system programmers which is able to edit/view many parts of PE (Portable Executable) files, dump them from memory, optimize them, validate, analyze, edit, …	http://www.woodmann.net/collaborative/tools/index.php/LordPE
Malcode Analyst Pack	The Malcode Analyst Pack contains a series of utilities that were found to be necessary tools while doing rapid malcode analysis.	http://labs.idefense.com/software/malcode.php

Name	Description	Website
	Included in this package are: - 4 explorer shell extensions - manual TCP Client for probing functionality. - mail server capture pot - spoofs dns responses to controlled ip's - HTTP, IRC, and DNS sniffer - Shellcode research and analysis application - aids in quick RE of packed applications - embeds multiple shellcode formats in exe husk - detect hidden processes	
MalZilla	Web pages that contain exploits often use a series of redirects and obfuscated code to make it more difficult for somebody to follow. MalZilla is a useful program for use in exploring malicious pages. It allows you to choose your own user agent and referrer, and has the ability to use proxies. It shows you the full source of webpages and all the HTTP headers. It gives you various decoders to try and deobfuscate javascript aswell.	http://malzilla.sourceforge.net
PEID	PEiD detects most common packers, cryptors and compilers for PE files. It can currently detect more than 600 different signatures in PE files.	http://www.peid.info
RegShot	Regshot is an open-source(GPL) registry compare utility that allows you to quickly take a snapshot of your registry and then compare it with a second one - done after doing system changes or installing a new software product.	http://regshot.sourceforge.net
Strings	Working on NT and Win2K means that executables and object files will many times have embedded UNICODE strings that you cannot easily see with a standard ASCII strings or grep programs. So we decided to roll our own. Strings just scans the file you pass it for UNICODE (or ASCII) strings of a default length of 3 or more UNICODE (or ASCII) characters. Note that it works under Windows 95 as well.	http://technet.microsoft.com/en-us/sysinternals/bb897439.aspx
Spider Monkey	SpiderMonkey is the code-name for the Mozilla's C implementation of JavaScript.	http://www.mozilla.org/js/spidermonkey/ (see also: http://blog.didierstevens.com/programs/spidermonkey/ for a malware analysis specific implementation)
TDIMon	With free TDIMon you can monitor the entire TCP and UDP activity on your computer.	http://download.chip.eu/en/TDIMon-1.01_163195.html
Whats Running	What's Running is a product that gives you an inside look into your Windows 2000/XP/2003 system. Explore processes, services, modules, IP-connections, drivers and much more through a simple to use application. Find out important	http://www.whatsrunning.net

Name	Description	Website
	information such as what modules are involved in a specific process.	
WireShark	Wireshark is an award-winning network protocol analyzer developed by an international team of experts.	http://www.wireshark.org

Conclusion

When a host's integrity has been compromised, it is absolutely essential that an organization be able to determine what activity the malicious code is engaged in and whether any data may have been compromised. It is only by obtaining that information that an accurate understanding of the risk posed by the compromise can be had. Once that risk is understood, a proper response to the incident can be provided.

Appendix A: Online Analysis Labs

Name	Description	Website
Anubis	Anubis is a service for analyzing malware. Submit your Windows executable and receive an analysis report telling you what it does. Alternatively, submit a suspicious URL and receive a report that shows you all the activities of the Internet Explorer process when visiting this URL.	http://anubis.iseclab.org/
Norman SandBox	Norman has a tool that allows free uploads of program files that you suspect are malicious or infected by malicious components, and provides instant analysis, with the result also sent to you by email.	http://www.norman.com/security_center/security_tools/
Virus Total	Virustotal is a service that analyzes suspicious files and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines.	http://www.virustotal.com/

Appendix B: Malware Sample Resources Online

Name	Description	Website
Offensive Computing	An online repository of malware, and forums related to research and analysis of malicious software.	http://offensivecomputing.net