# ABSTRACT

Malicious code detection and removal is very important to the security of the computer system. This project presents methodologies and tools to detect any malicious code present in the system and can be used as a preventive measure to protect the system from being infected. Malicious code analysis can be static or dynamic.

Static code analysis performs control flow and data flow analysis of the programs to detect malware. Dynamic code analysis has a greater edge over static code analysis. In this technique the instructions are analyzed as the code is being executed. Thus polymorphic malware can also be detected. The dynamic code technique makes use of a virtual environment to perform the analysis. Some malware can also detect the virtual environment and change behavior accordingly to hide itself from the defensive system. Thus dynamic analysis in a virtual environment is not an effective tool until it is used with some other tool that can detect the obfuscation of malware. The proposed tool examines the code in a virtual environment along with a minifilter driver and detects any malicious code present. The minifilter driver is used to monitor the windows API calls, registry changes and is used to generate reports. These reports can be analyzed to categorize a program as a malware or a normal program.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# 1. BACKGROUND AND RATIONALE

## 1.1 Malicious code

Malicious code is a term used to refer to any code that can cause undesired effects, security breaches and potential damages to the software system without the users consent. Software is classified as malware based on the users intent rather than the features of the software. Any harmful software is not a malware. For example, defective software can be legitimate and can still cause potential damage due to the presence of harmful bugs. Malware includes trojans, viruses, worms, spyware and any kind of software with intention to cause damage to the system. [Andreas, Ulrich 2006]

Destructive malware generally spreads by using popular communication tools to spread them. For example, worms can be spread using an email tool. They usually exploit the vulnerabilities on the target system to make their entry easy and unknown to the system.

A special category of malware called data-stealing malware exists. This malware intends to steal personal and confidential information of a person or an organization. The security threats of this kind are created using software like key loggers, adware, spyware and bots. This malware is typically stored in the cache memory which is frequently flushed. Once this kind of malware gets successfully installed on the target machine, they can take-over the IDS or anti-virus programs protecting the system.

## 1.2 Categories of Malware

Malware has become the greatest external threat to organizations causing resource damage and restoration overhead. Malware can be categorized into the following:

### 1.2.1 Viruses

A virus is a self-replicating program that inserts copies of itself into the target program or target machine. Generally viruses are initiated by user interaction. This happens whenever a user opens a file or runs a program.

### 1.2.2 Worms

A worm is a self-replicating program that requires no user interaction. It replicates itself based on some conditions. Worms are further categorized into: Network Service Worms and Mass mailing worms.

Network Service Worms replicate themselves and infect the target machines by making use of the vulnerabilities in the network.

Mass mailing worms are similar to email viruses except that they are self contained and do not infect other files [Kent K, Mell P, Nusbaum J].

### 1.2.3 Trojan Horse

A Trojan horse is a non-replicating and self contained program that looks like a useful program but has hidden code that has a malicious purpose and can cause damage to the system. Generally Trojan horses are used as aiding tools for others attacker tools [Kent K, Mell P, Nusbaum J].

### 1.2.4 Attacker Tools

Attacker tools are generally delivered to a target system as part of a malware infection. There is a large variety of attacker tools. Generally these tools help

attackers in gaining unauthorized access to infected systems and their data. They can also be used to launch additional attacks. The most common attacker tools are:

#### 1.2.4.1 Backdoors

A Backdoor is a tool that can be used to bypass the normal authentication and remain undetected. Generally a backdoor is a malicious program that listens to the commands being executed on certain TCP or UDP port. There is type of backdoor known as bot, which when installed allows the attacker to gain remote control over the infected system.

#### 1.2.4.2 Keystroke Loggers

A keystroke logger is a malicious program that is used to record and monitor the keyboard usage. These tools can be used to log confidential information of the user, like user names and passwords and send them to the attacker.

#### 1.2.4.3 Root kits

A root kit is a collection of malicious files that are deployed on a target machine and then they alter the functionality of the target systems in a stealthy way. Generally root kits make many changes to the target machine making itself difficult to detect.

#### 1.2.4.4 E-Mail Generators

An Email Generator is a program that is used by the attacker to send large number of emails that contain malware, spyware, and other infections to target systems without user's knowledge.

#### 1.2.4.5 Attacker Toolkits

Attackers make use of attacker toolkits which consists of wide variety of tools and utilities that can be used to probe and attack target systems. The tools include packet sniffers, port scanners, vulnerability scanners, password crackers and other scripts. The following table shows different malware categories.

| **Characteristic** | **Virus** | **Worm** | **Trojan Horse** | **Tracking Cookie** | **Attacker Tools** |
|---|---|---|---|---|---|
| **Is it self-contained?** | No | Yes | Yes | Yes | Yes |
| **Is it self-replicating?** | Yes | Yes | No | No | No |
| **What is its method of propagation?** | User-interaction | Self-Propagation | N/A | N/A | N/A |

Table 1.1 Differentiating Malware Categories [Kent K, Mell P, Nusbaum J]

## 1.3 Vulnerability to Malware

Many factors may leave a system vulnerable to attacks. The most common being the exploits of the bugs in the operating system design, existence of over- privileged users (who can leave the system vulnerable to the malware by making wrong decisions). Once a potential weakness is determined in an operating system, it can be used to launch attacks against all the machines that have the same operating system. Once a machine is compromised the malware can act intelligently hiding itself from the anti-malware and anti-virus programs, at the same time performing its intended task.

The user should always install the patches for the design weaknesses to protect itself from being prone to malware attack. Most of the web-sites on the World Wide Web are infected by malware. For example, the social networking site Twitter had a vulnerability called XSS security issue. This vulnerability allowed the malicious program developers to inject malicious code into the HTML pages, thus all the systems that visited this Web-site and had a lack of malware protection were infected by the malware. The XSS vulnerability allows the malware owners to hi-jack user accounts and also with the help of knowledge of other vulnerabilities, they could compromise the systems [Wiki 2009].

Another example of vulnerability is the Microsoft Office PowerPoint vulnerability. This unpatched vulnerability could allow the hackers to get arbitrary code executed with the privileges of the log-on user.

## 1.4 Malicious Code Analysis

Malicious code analysis is used to refer to the process of determining the intent and nature of the malware sample. This is very important to the process of developing the detection techniques for the malware. Also it is very important for developing tools that can be used to remove the malware from the system. For a long time the malicious code analysis was a manual, time consuming and tedious task. Thus there was need for automated systems which could detect the presence of the malware and automatically act to prevent the malware from achieving its intended task.

The most important preventive measures for malware are the virus scanner, but these scanners rely on a database of known signatures for virus. Thus they are restricted to only known viruses and malware, but many new types of malware and viruses attack

the computer systems every day. So there is a need for a better malware tracking solution. Whenever a new malware is found, its signature is written to the database of signatures, so that all systems infected with this malware can be easily fixed.

In addition it is very important to understand the functionality of the malware. In order to remove the malware, it is not sufficient to remove the binary from the Windows environment; all the registry settings affected by the malware should be restored.

Generally the malware analysis is conducted by allowing the malware program to be executed in a restricted environment and observe the actions. Then the actions of the program are analyzed (usually a debugger is used). This manual analysis is a time taking and tedious process. Thus there is a need for automated analysis programs. This automation is generally achieved by executing the affected program in a virtual environment and recording the actions of the programs and finally sending the recorded actions to the human analyst [Andreas, Ulrich 2006].

The existing automated malicious code analyzers have shortcomings. One of the very important aspect among them being the failure of the analyzers due to the presence of detection routines within the malware. The detection routines allow the malware to detect if the program is running in a virtual environment. If so, the malware program acts in a different way, thus hiding its existence. Some malware have the capability to check the existence of both hardware and software breakpoints, which can be used to detect the existence of a malware.

Other problems with the automated malware analysis include the incapability of the tool to detect the complete interaction of the program with the system.

### 1.4.1. Static Analysis

Static analysis is the process of analyzing the malware without actually executing it. In this technique the binary code is converted into corresponding assembler level instructions. After the transformation, control flow and data flow analysis techniques are implemented to draw a conclusion about the programs functionality. The following figure shows a static analyzer. [Feng M, Gupta R]

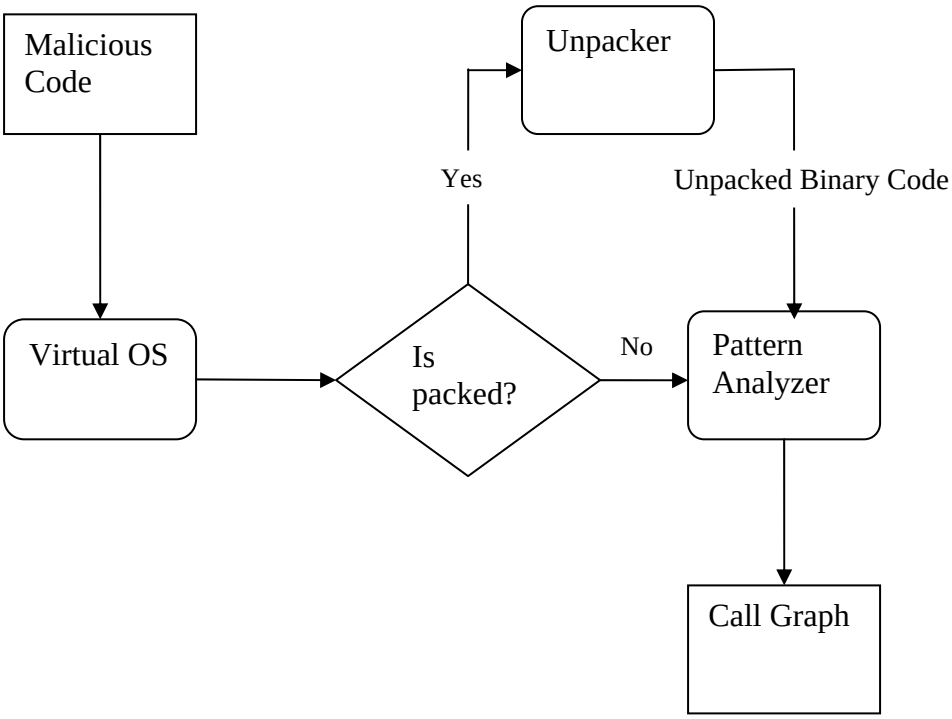


Figure 1.1 A Static Analyzer

Static analysis is faster in performance than dynamic analysis. One of the major disadvantages of static analysis is the ability of the malware to make use of binary obfuscation, which can be used to safely play with the control flow and data flow analysis. The binary obfuscation preserves the programs functionality while at the same time making the parsing of the program difficult. The malware can also make use of code

obfuscation which makes it difficult to perform the data flow and control flow analysis. These obfuscation techniques are implemented with the help of opaque predicates and opaque constants. Opaque predicates are defined as “Boolean valued expressions whose values are known to the obfuscator but difficult to determine for an automatic deobfuscator”. Opaque constants are similar to Opaque predicates but they hold integer values [Christodorescu,M., Jha, S].

It’s not necessary that the code analyzed by the static analyzer is the code that will be actually executed. This is true in particular for the self-modifying programs that make use of polymorphism to hide their actual form.

### 1.4.2. Dynamic Analysis

In contrast to static analysis, the dynamic analyzers analyze the code when it is being executed. The most important advantage of dynamic analyzer is that the instructions that are analyzed are the ones that are executed. These tools provide security against the obfuscation techniques. The following figure shows a dynamic analyzer.

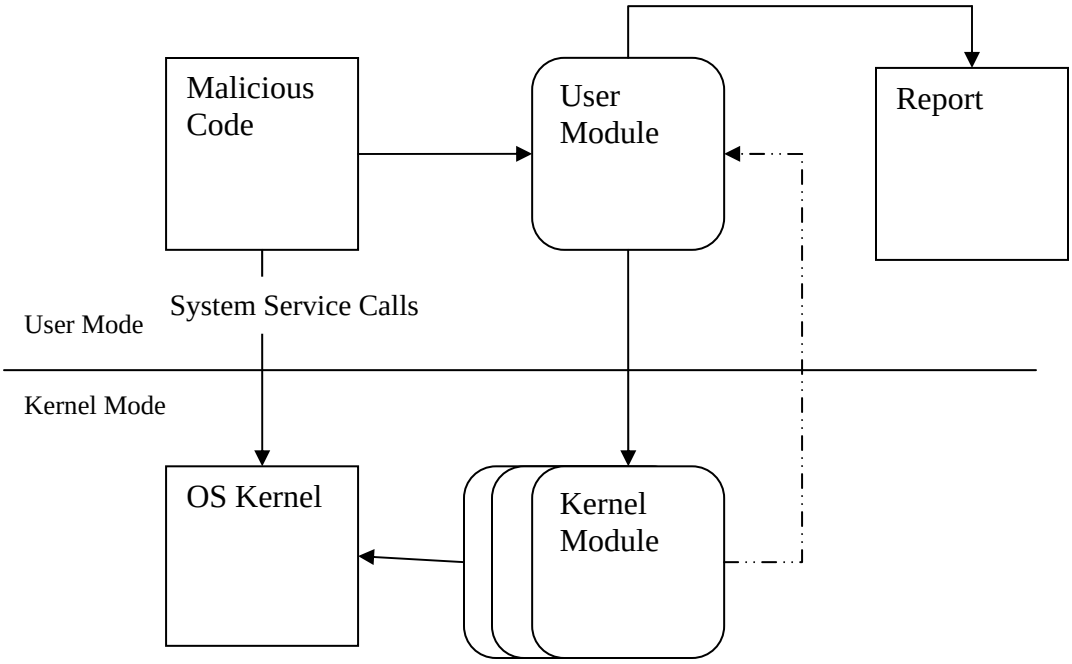


Figure 1.2 A Dynamic Analyzer

Generally the analysis is conducted in a virtual environment. Thus the risk of system being damaged is reduced, because the virtual environment image could be replaced with a new one. [Feng M, Gupta R]

One of the significant drawbacks of conducting the analysis in a virtual environment is that the malware could determine that it is running in a virtual environment and may change its behavior accordingly.

Virtual environment detection tools are easily available. These tools make use of CPU instructions to determine the existence of a virtual environment. For example the following sample code can be used to test the existence of a virtual environment:

```
int swallow_redpill () {
      unsigned char m[2+4], rpill[] =
      "\x0f\x01\x0d\x00\x00\x00\x00\xc3";
      *(unsigned*)&rpill[3]) = (unsigned)m;
       ((void(*)())&rpill)();
       return (m[5]>0xd0) ? 1 : 0;
     }
```

The heart of this code is actually the SIDT (store interrupt descriptor table) instruction (encoded as 0F010D [*addr*]), which stores the contents of the interrupt descriptor table register (IDTR) in the destination operand, which is actually a memory location. One very interesting thing about the SIDT instruction is that, it can be executed in non privileged mode but it returns the contents of the sensitive register, used internally by the operating system.

Because there is only one IDTR register, but there are at least two OS running concurrently (i.e. the host and the guest OS), VMM needs to relocate the guest's IDTR in a safe place, so that it will not conflict with a host's one. Unfortunately, VMM cannot know if (and when) the process running in guest OS executes SIDT instruction, since it is not privileged (and it doesn't generate an exception). Thus the process gets the relocated address of the IDT table. It was observed that on VMWare, the relocated address of IDT is at address 0xff*XXXXXX*, whereas on Virtual PC it is 0xe8*XXXXXX*. This was tested on VMWare Workstation 4 and Virtual PC 2004, both running on Windows XP host OS [Quist D, Val Smith].

# 2. Minifilter Driver

## 2.1 Analyzing a Executable

The ultimate goal of this project is to analyze a given executable and generate a report of the changes made to the system by the executable. Then analysis of the report is performed to decide if the executable is a malware or a normal program.

The executable is tested in a virtual environment. The analysis tool has two main components. They are:

1. **Minifilter Driver**: The minifilter driver is used to dynamically monitor the activities of the program that is being tested. This driver can track the Windows API calls made by the program.
2. **Analysis Tool**: This tool works along with the driver. It makes use of the track report of the Windows API calls made by the program and generates a report that is understandable by an analyst. The report includes all the file and registry operations made by the program.

The minifilter driver is created using Windows programming with Windows Driver Kit (WDK) [Microsoft-7]. The analysis component is developed using C and C++ programming with Visual C++. Based on the reports generated by the analysis tool, the changes made by the program are taken into consideration, analysis is performed and a decision is made as malicious or normal program.

## 2.2 Malicious Code Detection

Malicious code detection is the process of detecting various malware that can cause potential damage to the system. Any defense technology can be separated into two components – a technical component and an analytical component. In reality, these

components may not be clearly separable at the module or algorithm level within every malicious program. However, in terms of function, their differences are significant and important. The technical component is a collection of program functions and algorithms that selects the data that will be analyzed by the analytical component. This data may be anything – from text strings within a file, to a specific action the program performs, to a full sequence of actions that the program performs, and more.

The analytical component serves as the decision-making system. It assesses the data provided by the technical component using one or more algorithms and then issues a verdict about the data. The security program will then use the verdict to take action on the malicious program according to the security policy that has been set in the security program. For example, a few of the possible actions that could occur based upon the verdict might be –

a. Notifying the user

b. Requesting further instructions from the user

c. Placing a file in the quarantine

d. Blocking unauthorized program actions

Consider the following example, which is extracted from the assembly code of Bagle, which is a widespread email-based virus. For ease of presentation and understanding, some simplifications of the original code have been performed.

```
1 lea edi, ptr [ebp+0x4025]   // edi = mem[ebp+...]
2 mov edx, 0xef4013a0         // edx = 0xef4013a0
3 mov ecx, 0x3ec5             // ecx = 0x3ec5

   loop:

4 mov al, byte ptr ds[edi]    // al = mem[ds+edi]
5 sub al, dl                  // al = al - dl
6 sub al, dh                  // al = al - dh
```

```
7 xor al, cl                    // al = al . cl
8 rol edx, cl                   // rotate edx by cl bits

9 mov byte ptr ds[edi], al      // mem[ds+edi] = al

10 inc edi
11 dec ecx

12 jnz loop                     // jump
13 push edi                     // push args into stack
14 call 0x7c92a950              // call a lib function
```

The key part of the sample code is a loop formed by instructions 4 through 12. The instructions preceding the loop (i.e., instructions 1 through 3) initialize the loop counter (*ecx*), starting address (*edi*), and another variable (*edx*). During each iteration, the loop fetches a value from the data segment, performs a calculation based upon that value, and then finally puts the new computed value back into the data segment. Following the loop, the program calls a library function that uses the newly computed values. The use of these values triggers the actions of the virus. Many different kinds of obfuscation transformations can be applied to this piece of code to affect mutations of the Bagle virus [Feng M, Gupta R].

## 2.3 Minifilter Driver

### 2.3.1 Minifilter

The filter manager is a kernel-mode driver that performs in accordance to a standard file system filter model. The ultimate goal of a filter manager is to provide the generic functionality that is required by file system filter drivers. This functionality is very useful for the third party driver developer to develop and write minifilters for the user applications [Microsoft-3].

The minifilter technique is very simple and easy to develop than the corresponding file system filter drivers. By taking advantage of this functionality, third-party developers can write minifilter drivers, which are simpler to develop than legacy file system filter drivers. This process shortens the driver production process with far superior quality. The applications developed by minifilters are more robust and versatile [Microsoft-1].

### 2.3.2 Filter Manager Concepts

All Windows operating systems have a Filter manager installed on them. But the filter manager is turned into active mode only when a minifilter driver is loaded. The filter manager works by attaching itself to the file system stack and thus acquiring a place in the target volume. On the other hand minifilter driver has an indirect attachment to the target volume by registering itself with the filter manager. The minifilter driver can register itself with the filter manager to perform filtering of a chosen set of I/O operations.

#### 2.3.2.1 Load Order Groups

The “load order group” determines the position of a legacy filter driver position in the file system I/O stack relative to other filter drivers. This can be better explained by the following example. An antivirus filter driver should always be at a higher position in a file system I/O stack than replication filter driver. This position of antivirus filter driver is required in order to detect viruses and disinfect files before they can cause further damage to the other machines. Thus, filter drivers located in the FSFilter Anti-Virus load order group are higher in position and are loaded before filter drivers located in the

FSFilter Replication group. Every filter driver in the file system has a corresponding system-defined class. There is also a GUID specified in the .inf file for the filter driver [Microsoft-4].

#### 2.3.2.2 Altitude of a Minifilter

Similar to the legacy filter drivers, all the minifilter drivers attach to the file system stack in a particular order. A unique identifier called *altitude* determines the order of attachment of a minifilter driver. The altitude of a minifilter is the characteristic that identifies the position of a minifilter relative to other minifilters in the I/O stack when the minifilters are loaded. The altitude is an infinite-precision string interpreted as a decimal number. Lower the altitude value, earlier the filter loaded. It means a minifilter that has a low numerical altitude is loaded below a minifilter that has a higher numerical value in the I/O stack [Microsoft-4].

#### 2.3.2.3 Instance of a Minifilter

The attachment of a minifilter driver at a particular altitude on the file system stack is called an *instance* of the minifilter driver. The altitude of a minifilter driver ensures the order of loading instances of minifilter drivers is appropriate. The altitude also determines the order in which the minifilter drivers are called by the filter manager to handle I/O. The allocation of altitudes to the instance of minifilter drivers is managed by Microsoft. The following figure shows a simplified I/O stack with the filter manager and three minifilter drivers. The following figure shows a single filter manager on a I/O stack.

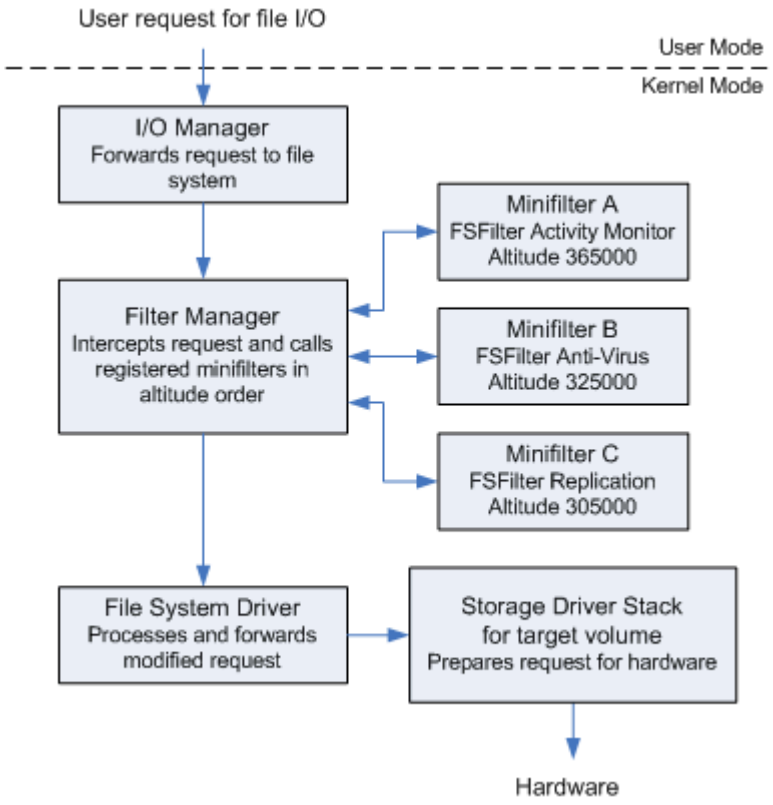


Figure 2.1 I/O stack with single filter manager and three minifilter drivers [Microsoft-2].

#### 2.3.2.4 Callback Routines of a Minifilter

A minifilter has the capability to filter all the three major I/O based operations. The three I/O operations are IRP-based I/O operations; fast I/O and file system filter (FSFilter) callback operations. The minifilter can register with a preoperation call back routine, a post operation callback routine, or both for filtering each of the above three I/O operations. The filter manager makes a call to the appropriate callback routine for each minifilter driver whenever they need to handle an I/O operation. It can call a callback routine only when it is registered. When the callback routine returns, the filter

manager makes a call to other callback routine registered for other minifilter drivers for the same I/O operation [Microsoft-2].

The call to the callback routines by the filter manager are in order of altitude from highest to lowest (A, B, C). The I/O request is then forwarded to the next-lower driver for further processing. Once a request for I/O operation complete is receive by the filter manager, a call to post operation callback routines of each of the minifilter driver is made in reverse order, from lowest to highest (C, B, A) [Microsoft-5].

Generally there is a need for interoperability between the minifilter drivers and the legacy filter drivers. To achieve this interoperability the filter manager attaches filter device objects to a file system I/O stack in multiple location. Each of the filter manager's filter device objects is called a *frame*. The legacy filter driver perceives each filter manager frame as just another legacy filter driver.

Every filter manager frame makes use of band of altitudes. The filter manager is robust enough, so that it can create a new frame or modify an existing frame to attach to the file system at the correct location.

It’s always important to verify the interoperability among the legacy filter drivers and the minifilter drivers. If there are issues with the interoperability, there is a need to replace legacy filters with minifilters. If a minifilter driver is unload operation is performed and then a reload operation is called, the minifilter is reloaded at the same altitude in the same frame from which it was unloaded.

The following figure shows a simplified I/O stack with a two filter manager frames, minifilter driver instances, and a legacy filter driver. The following figure shows multiple filter managers on a I/O stack.

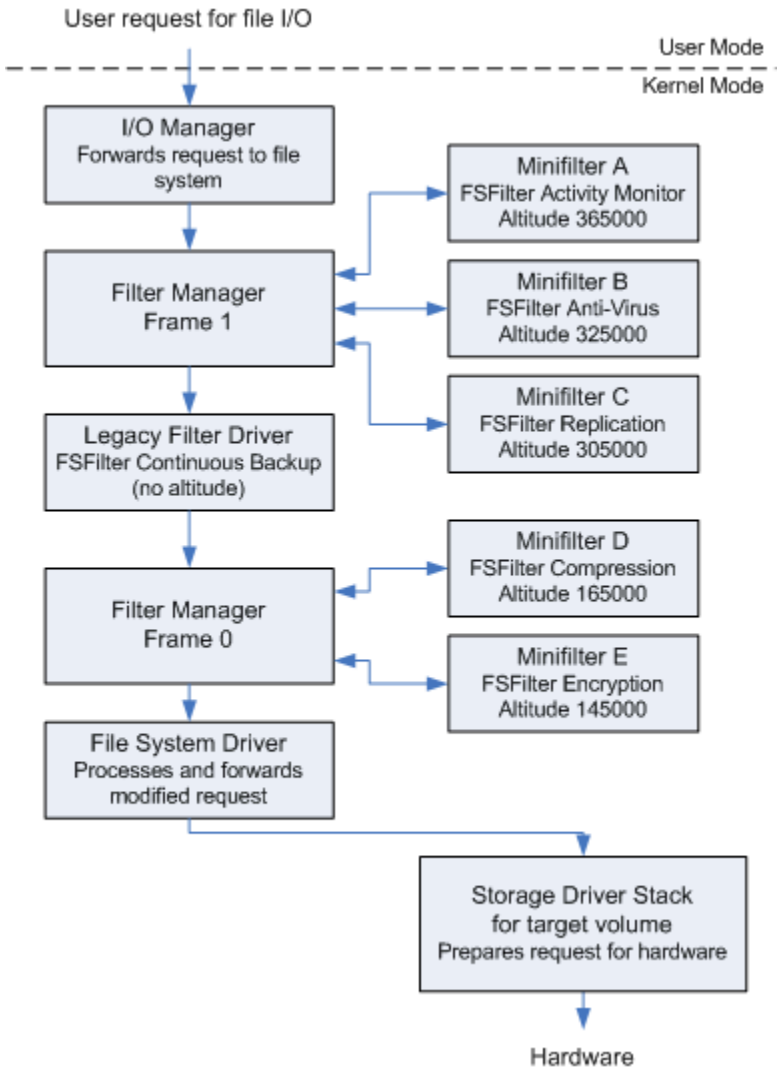


Figure 2.2 I/O stack with two filter manager frames, minifilter driver instances, and a legacy filter driver [Microsoft-2]

## 2.4 Advantages of Minifilter Drivers

The minifilter model offers the following advantages over the existing legacy filter driver model:

1. Filter load order control is easy
2. Unloading a minifilter while a system running is possible
3. Ability to process only necessary operations
4. Kernel stack is used more efficiently
5. Code redundancy is reduced
6. Less complexity
7. New operations can be easily added
8. Can support multiple platforms easily
9. Better support for user-mode applications

## 2.5 Analyzing the reports and decision making

Once a report is generated by the tool, analysis can be performed by the analyst. Based on the operations performed by the program as listed in report, the analyst can classify a program as a malicious program or a normal program. For example, the following is the analysis of a sample report. The report is from Symantec Antivirus:

**Name**: *Auraax.c*

**Type**: Worm

**Infection Length**: 27,136 bytes

**Systems Affected**: Windows 98, Windows 95, Windows XP, Windows Me, Windows Vista, Windows NT, Windows Server 2003, and Windows 2000.

Whenever the worm executes, it replicates itself and infects all the machines that are prone to it. It copies into the system files of a machine as follows:

***%ProgramFiles%\Microsoft Common\wuauclt.exe***

Once a machine is infected, it then creates several files on the infected machine. For example the following files are created:

1. ***%Windir%\Temp\rld[SINGLE NUMBER].tmp***
2. ***%System%\config\systemprofile\Local Settings\History\desktop.ini***
3. ***%System%\config\systemprofile\Cookies\index.dat***
4. ***%System%config\systemprofile\Local Settings\Temporary Internet Files\desktop.ini***

After the creation of files, the worm alters the following system processes:

1. ***svchost.exe***
2. ***explorer.exe***

Then the worm makes few new entries into the windows registry. These entries run every time the system boots.

The entries are as follows:

***HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Image File Execution Options\explorer.exe\"Debugger" = "%ProgramFiles%\Microsoft Common\wuauclt.exe"***

The worm also modifies few existing registry entries, which are loaded every time the system boots. The entries are as follows:

***HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\"Userinit" = "%System%\userinit.exe, ProgramFiles%\Microsoft Common\wuauclt.exe"***

The worm also creates registry entries that have the capability to bypass the Windows firewall. They are:

***HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\AuthorizedApplications\List\"%ProgramFiles%\MicrosoftCommon\wuauclt.exe"="%ProgramFiles%\Microsoft Common\wuauclt.exe:*:Enabled:EMOTIONS_EXECUTABLE"***

The worm also searches the kernel drivers for .sys extension files in the %System%\DRIVERS\ folder so that it can overwrite these files. These files are generally overwritten with a root kit so that the worm can hide itself.

The worm also modifies the host machine files, so that the host machine is prevented from accessing the following websites.

1. 127.0.0.1 downloads.microsoft.com
2. 127.0.0.1 downloads1.kaspersky-labs.com
3. 127.0.0.1 downloads1.kaspersky-labs.com
4. 127.0.0.1 downloads1.kaspersky-labs.com
5. 127.0.0.1 downloads2.kaspersky-labs.com
6. 127.0.0.1 downloads3.kaspersky-labs.com [Symantec 2008].

# 3. SYSTEM DESIGN

## 3.1 Overview of the system

The use of a dedicated standalone system for testing the malware is not an efficient solution. The dedicated system can be reinstalled after each dynamic test run is performed, but this induces very high cost. In order to overcome the disadvantages of a standalone system, the tool run the tests in a virtual environment, thus limiting the effect of the malware only to the virtual machine but not the real system.

In case of a virtual system, the infected virtual image is replaced with a new one. Thus there is no need to reinstall software on the machine, thereby reducing the overhead. Virtual machines are very fast and similar to the real one in-term's of the execution speed. A major drawback of the virtual solution is that the malware may detect that the environment in which it is running is the virtual environment and may change its behavior accordingly.

The alternative solution to the above problem is the use of an emulator. A PC emulator is a piece of software that emulates the functionality of a real system including all the real time resources of a system. There is a subtle difference between an emulator and a virtual machine. Virtual system executes a statically dominant set of instructions directly on a real system, whereas an emulator simulates all the instructions in software [Duan H, Guan Y, Zhang J].

But there is a problem with software emulators. They are not easily available for Windows based Operating System. Hence there is a need for a tool that can work with a virtual Operating system and still detect malware.

Also there is a very important difference between the speed of execution on a real time system and the speed of execution on a virtual Operating System. This difference can be used by the malware to judge whether it is being run on a virtual Operating System or a real system. This disadvantage can be overcome by using a minifilter driver which can redirect operations and makes a malware believe that it is not being run in a test environment.

## 3.2 Information Analyzed

It is possible to classify the types of information that is captured during the analysis phase of the system. Many systems concentrate on the communication between the application and the operating system. This includes intercepting system calls and hooking API calls.

There are tools that can be used to list all the windows processes running in a system. Also it is possible to log the windows registry and the file system activity. Generally these tools are implemented as operating system drivers that can intercept the native windows calls. Thus they are invisible to the application that is currently on analysis. There are also tools that can intercept arbitrary user functions, including all windows API calls. This requires some rewriting of the target function. This rewriting could be detected by a malware and thus it may act differently to overcome the detection.

In order to overcome the above problem, virtual Operating System is used along with a minifilter tool. The minifilter tool has complete control over the system. It has the capability of analyzing both the native windows calls as well as the windows API calls, at the same time being unidentified by the malicious code. Because of the use of the minifilter which has complete control over the system, the analysis being performed is

more fine grain. This functionality is similar to the debugger but the technique does not make use of break points, which are known to create problems when used for analyzing malicious code. The reason being the software break points can be detected using code integrity checks and the malware could act accordingly [Kirda E, Ulrich 2006].

The minifilter tool is used for analyzing windows executables especially the files corresponding to the PE file format. In this technique the program is tested in a virtual environment and the valid native windows calls and windows API are logged for analysis.

## 3.3 Testing Environment

The testing environment is very simple. The tool has two major parts. They are:

1. **Minifilter Driver**
2. **Analysis Tool**

The whole testing is performed on a Virtual System. First the minifilter is installed on the virtual operating system using Windows Installer. Then the analysis tool service is started. The service can be started using the windows executable *sc.exe* which is used to start, suspend and stop services on the windows operating system. The tool is started using a command prompt. Once the minifilter is installed and the service is running, the program can be tested [OSR 2010]. The following chart shows the designed tool architecture.

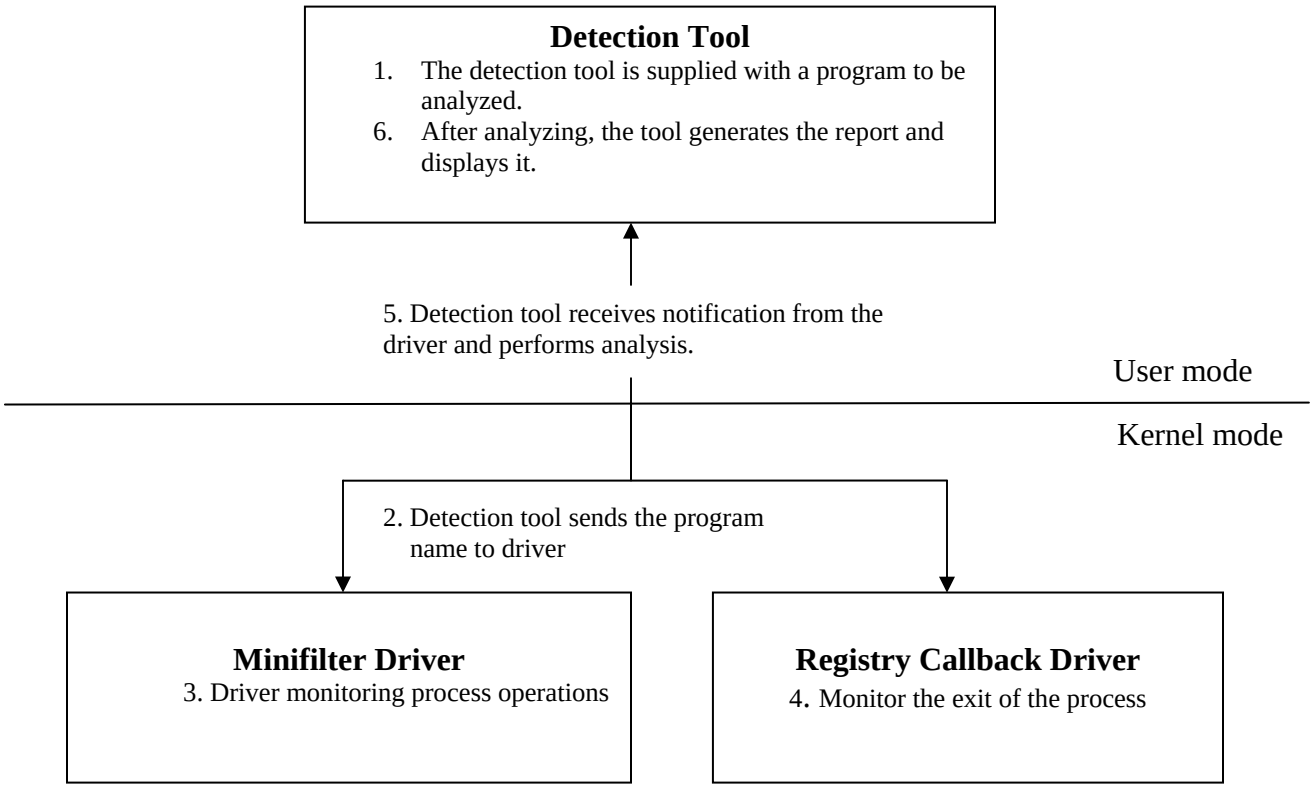


Figure 3.1 Architecture of the Malicious Code Detection Tool

The testing is performed in the following steps:

1. The detection tool is supplied with a program to be analyzed.
2. The program name is sent to the driver
3. Driver monitoring process operations
    a. driver monitors the process creation
    b. drivers monitors the activity and redirect any operation
4. Driver monitor the exit of the process
5. Detection tool receives notification from the driver and performs analysis
6. After analyzing, the tool generates the report and displays it.

### 3.3.1 Redirection using the detection tool

Redirection explains how the detection tool controls the malware by having greater control over the system. The following redirection schemes are used:

**1**. Suppose program wants to read an existing registry key:

***HKLM\Software\Microsoft\Windows\TaskManager***

Analyzer will let the program read the value and return to malware program.

**2**. Suppose malware wants to create a registry key

***HKLM\Software\Microsoft\Windows\MalwareXXX***

Analyzer will create a registry key as:

***HKLM\Software\Analyzer Sandbox\Malware***

– > Sub Key will be:

***HKLM\Software\Microsoft\Windows\MalwareXXX***

The tool makes the malware believe that it is not detected by sending wrong registry keys. It monitors the creation and deletion of registry keys.

**3**. Whenever malware tries to read a registry key, Analyzer will first check the sandboxed key and if found will return the value from there, otherwise will let the operation go as usual.

**4**. What happens if malware modifies any registry key?

This operation will be considered as same to creation of key

**5**. What happens when file operation is requested?

It treats as if registry names are some file names and file paths.

## 3.4 Installing a Minifilter Driver

All the Windows Operating systems including Windows XP and latter, minifilter drivers are installed by using an INF (Setup Information file) and an installation application. In all the previous operating systems, minifilter drivers were installed by the Service Control Manager [Microsoft-5].

The "INF-based installation" means that the use of INF file is to copy files and to store information in the registry. The system does not depend on a single INF file for the whole installation.

### 3.4.1 Creating an INF File for a Minifilter Driver

An INF file for a file system minifilter driver is very important for installing the minifilter drivers. An INF file generally contains the following sections:

1. Version (required)
2. DestinationDirs (optional but recommended)
3. DefaultInstall (required)
4. DefaultInstall.Services (required)
5. ServiceInstall (required)
6. AddRegistry (required)
7. DefaultUninstall (optional)
8. DefaultUninstall.Services (optional)
9. Strings (required)

#### 3.4.1.1 Version Section (required)

The **Version section** specifies a class and GUID. These class and GUID characteristics are used to determine the type of minifilter driver. This can be shown in the following code example.

```
[Version]
Signature           = "$WINDOWS NT$"
Class               = "ActivityMonitor"
ClassGuid           = {b86dff51-a31e-4bac-b3cf-e8cfe75c9fc2}
Provider            = %v@t%
DriverVer           = 10/09/2001,1.0.0.0
```

| Entry | Value |
|---|---|
| **Signature** | "$WINDOWS NT$" |
| **Class** | Specifies the class name for the minifilter driver |
| **ClassGuid** | Specifies the GUID for the minifilter driver |
| **Provider** | The name of the software company which is developing the drivers |
| **DriverVer** | Specifies the driver version |
| **CatalogFile** | This field is empty for non-antivirus minifilter drivers.<br><br>This filed contains the name of a WHQL-supplied catalog file for Antivirus minifilter drivers which are signed. |

Table 3.1 Table describing the contents of the version section in a INF file [Microsoft-3].

Table 3.1 describes the version section of a INF file.

#### 3.4.1.2 DestinationDirs Section (optional but recommended)

This section specifies the directories where minifilter driver and application files will be copied.

In both *DestinationDirs Section* and *ServiceInstall* section, well-known system directories can be specified by system-defined numeric values. In the following code example, the value 12 refers to the Drivers directory (*%windir%\system32\drivers* on Windows NT-based platforms), and the value 10 refers to the Windows directory (*%windir%*) [Microsoft-3].

```
[DestinationDirs]
DefaultDestDir          = 12
Mafiltertool.DriverFiles    = 12
Mafiltertool.UserFiles      = 10,FltMgr
```

#### 3.4.1.3 DefaultInstall Section (required)

The *DefaultInstall section* makes use of a *CopyFiles directive.* This directive can be used to copy the minifilter driver's driver files and user-application files to the targets directories that are enlisted in the *DestinationDirs* section [Microsoft-3].

#### 3.4.1.4 Strings Section (required)

The **Strings section** defines each %*strkey*% token that is used in the INF file. The following code example shows a typical **Strings** section [Microsoft-3].

```
[Strings]
V@t                     = "Vinay@TAMUCC"
ServiceDescription      = "Malprober Mini-Filter Driver"
ServiceName             = "Malprober"
DriverName              = "Malprober"
DiskId1                 = "PassThrough Device Installation Disk"
```

**RegInstancesSubkeyName** = "Instances"
**RegAltitudeValueName** = "Altitude"
**RegFlagsValueName** = "Flags"
**DefaultInstance** = " Mafiltertool - Top Instance"
**Instance1.Name** = " Mafiltertool - Middle Instance"
**Instance1.Altitude** = "370000"
**Instance1.Flags** = 0x1 ; Suppress automatic attachments
**Instance2.Name** = " Mafiltertool - Bottom Instance"
**Instance2.Altitude** = "365000"
**Instance2.Flags** = 0x1 ; Suppress automatic attachments
**Instance3.Name** = " Mafiltertool - Top Instance"
**Instance3.Altitude** = "385000"
**Instance3.Flags** = 0x1 ; Suppress automatic attachments

## 3.5 Analysis Process

The analysis process is started by allowing the given program to execute in an emulated environment. When the program executes, all the operating system services that are requested by the program are noted. Every action that involves communication with the environment, requires some operating system services, it cannot directly interact with the hardware.

In a windows operating system environment, the application cannot directly interact with the windows native API. They are supposed to make use of the functions provided by the operating system to interact with the operating system services.

Malware writers make direct use of these native API to avoid any kind of DLL dependencies or confuse the virus scanners. This tool takes care of both Windows API function calls by an application and native API calls of an application, thus making the probability of a malware escaping the analysis very low.

The tool is supposed to track which operating system services are used by a program. This tracking requires us to solve two problems:

1. We must be able to track the execution of a malware process and also distinguish between the instructions executed by a malware process and the instructions executed by a normal process. This is very important because the emulated environment does not only run the instructions of the malware process, but also the native operating system instructions and instructions of the other supporting processes.
2. We need to make sure that the native API call or a windows API call is invoked without any kind of modification to the malware sample.

The PDBR (Page Directory Base Register) can be used to track the execution of the instructions. This project creates a tool that makes use of virtual OS and minifilter drivers to detect malicious code. The following figure shows the overview of the malicious code detection tool.

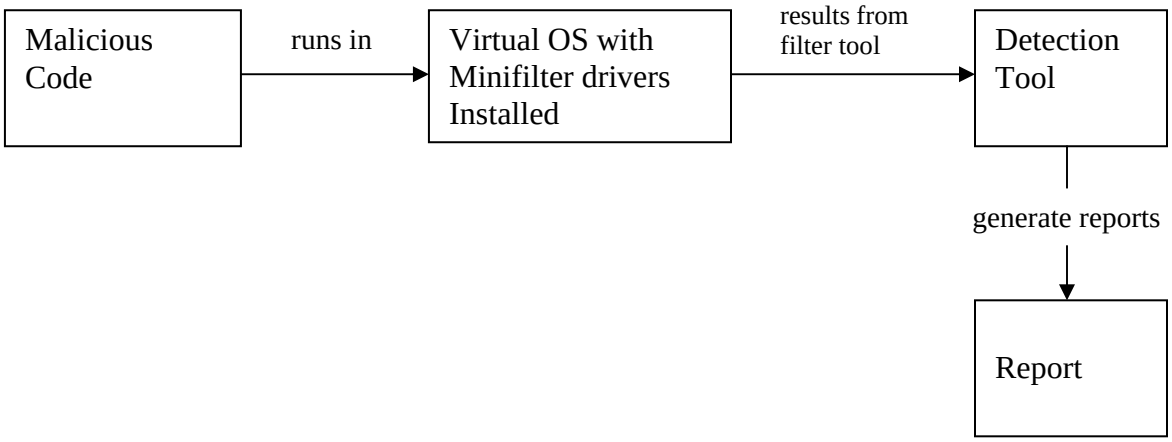


Figure 3.2 Overview of the Malicious Code Detection Tool

# 4. EVALUATION AND RESULTS

To demonstrate the capability of the malware tool in successfully monitoring the actions of malicious code, the tool runs dynamic tests on current malware samples. Then the results of the tool are compared to the solutions provided by various anti-virus providers. The ultimate goal of the evaluation is to determine till what extent our analysis results match the characterizations provided by this well-known anti-virus vendor.

For the selection of our test subjects, we make use of Symantec's list of the most prevalent malware samples that are published. Unfortunately, it is not possible to obtain samples for all entries on these lists. However, we select some set of different malware programs that represent a good mix of different malicious code variants currently popular on the Internet. Some of these samples may be packed using different executable packer programs; others may not be recognized as valid Windows PE executables. From this pool, we choose one working sample for each malware type. Then, we scan all samples for our experiments by the online virus scanner provided by Symantec and make sure that they were all recognized correctly. There can be differences between the results of our tool and the virus descriptions of the various anti-virus providers [Kirda E, Ulrich 2006].

The detection tool was also able to recognize many viruses that are enlisted neither on Symantec's anti-virus list nor on Kaspersky's anti-virus list. The technical detail of malware includes the files, registry, processes and services affected by the malware. Generally these changes by a particular virus are not the same on different computers. The probable reason can be that the malicious code chooses random file names or a name from a list of options that are not exhaustively covered by the malware description. Another possible reason for the variation in output can be analysis of a

malware variant rather than the malware about which the virus scanner has published the technical details.

The following table shows some malwares that were analyzed by the tool and compared to the results of the Symantec anti-virus list.

| Malware Name | File | Registry | Process | Service |
|---|---|---|---|---|
| **W32.Storm.Worm** | Yes | P | P | P |
| **W32.HLLW.Doomsjuice** | Yes | P | P | P |
| **W32.Sality.AE** | Yes | P | P | P |
| **W32.Qquzlzb.exe** | No | No | No | No |
| **W32.Srvcp.exe** | No | No | No | No |

Table 4.1 MalProber Test Results

In the above figure ‘P’ refers to partial matches. The partial matches occur due to the malware dependency on the target system. This dependency occurs due to the systems execution environment. Generally files are also dependent on the target system but the core files that are created or affected by the malware are always the same.

Also there are few malware samples listed above whose virus definitions were not found on the Symantec’s anti-virus list. These viruses were detected by our tool.

Whenever some normal process is analyzed, the tool displays the changes made by the program to the system. This tool can also be handy for people who want to study the nature of their processes.

The following steps describe the process of analyzing a malware or a normal process.

Step 1: Installing the Malprober Driver

1. Go to the project folder
2. Open the directory /Malware Analyzer/Driver
3. Install the minifilter drivers by right clicking the Malprober.inf and selecting "install".

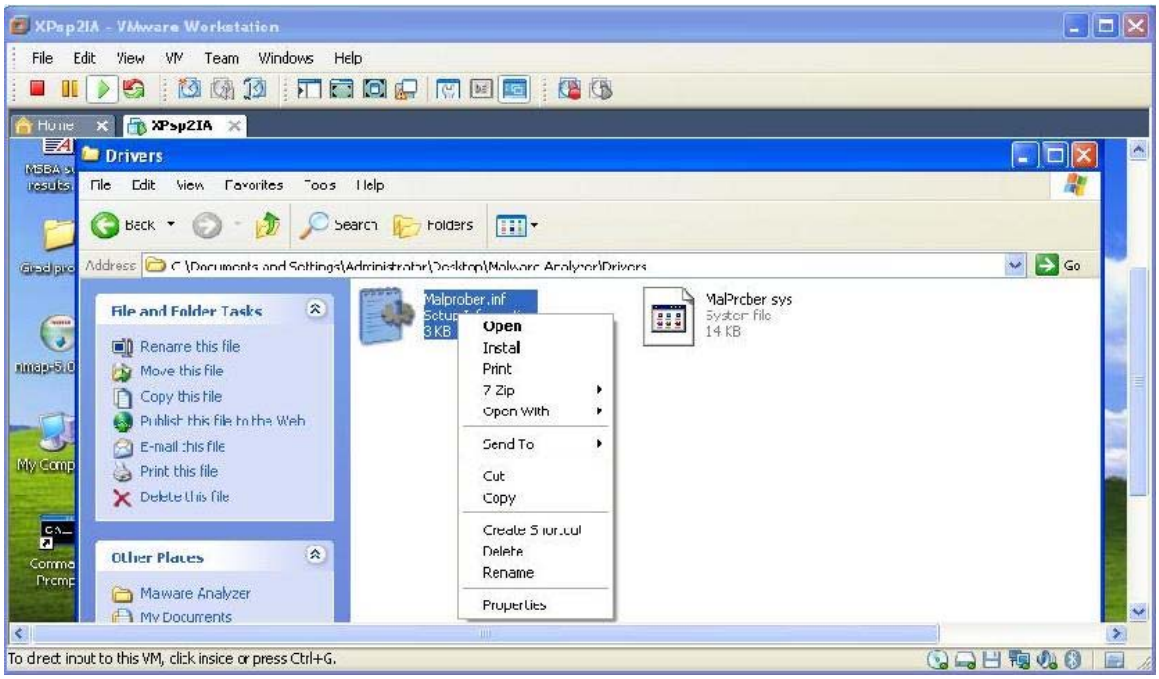


Figure 4.1 Installing Malprober Driver

The above figure shows how to install the Malprober Minifilter Driver.

Step 2: Starting the Malprober Service

1. Open a command prompt on the virtual Operating System.
2. Type the following command to start the tool

Command: sc start malprober

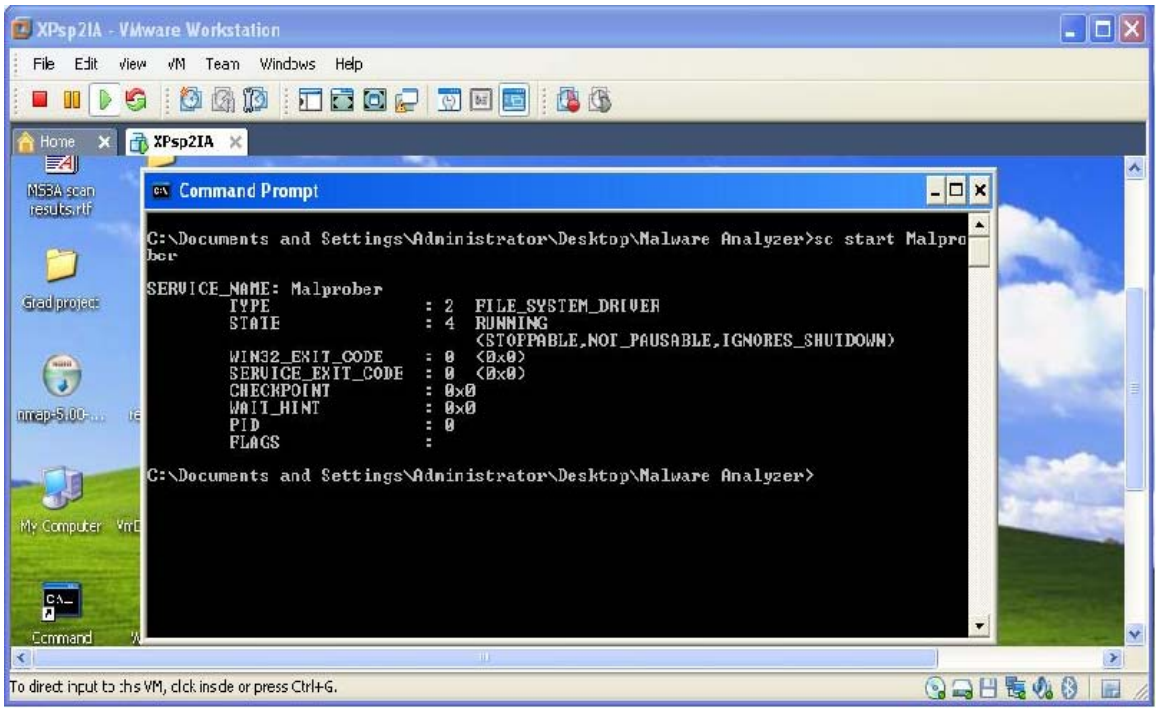


Figure 4.2 Starting the Malprober service

The above figure shows how to start the Malprober Service.

Step 3: Testing a program with Malprober

1. Type the following command for testing the sample program and generating reports.

   Command: u_malanalyze.exe usbview.exe

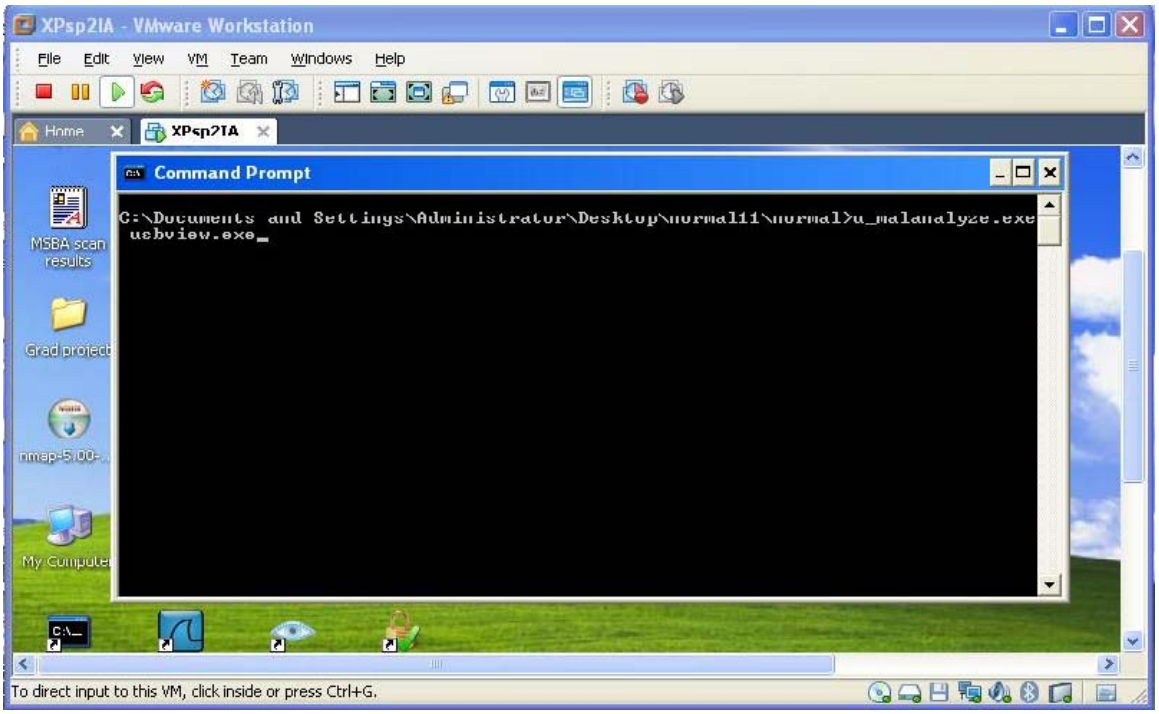


Figure 4.3 Testing usbview.exe with Malprober Tool

The above figure shows how a program named “usbview.exe” is tested with the Malprober. Once we run the tool, the tool generates a report for the binary. The generic report contains all the file operations, registry and timestamp changes made by the binary under analysis. At the end of the report, there is an automated analysis of the program behavior.

## 4.1 Sample Reports

### Sample#1

### A genuine program. The following is the report for usbview.exe.

1. The program created the following new files
---
/*=========================================*/

2. The program opened the following files
---
/*=========================================*/

3. These files are read by the program
---
/*=========================================*/

4. These files are written into by the program
---
/*=========================================*/

5. The following files security permissions are changed
---
/*=========================================*/

6. These files are deleted by the program
---
/*=========================================*/

7. These files are sent ioctl command
---
/*=========================================*/

8. The timestamps are changed for the following files
---
/*=========================================*/

9. The following registry keys are created
---
\Registry\Machine\SOFTWARE\Microsoft\Cryptography\RNG
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\SystemCertificates\My
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Control\DeviceClasses
/*==========*/

10. The following registry key values set
---
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Cryptography\RNG Seed
/*==========*/

11. The following registry keys are deleted
---
/*==========*/

12. The following processes are created
---
/*==========*/

/*==========*/

Automated analysis of the program behavior

/*==========*/

*********
##### The program seems to be a genuine program#####
**********

## Sample#2

## A little suspicious program. The following is the report for msrll.exe.

1. The program created the following new files
---
\Device\HarddiskVolume1\WINDOWS\system32\mfm\msrll.exe
/*======================================*/

2. The program opened the following files
---
\Device\HarddiskVolume1\WINDOWS\system32\ws2_32.dll
\Device\HarddiskVolume1\WINDOWS\system32\ws2help.dll
\Device\HarddiskVolume1\WINDOWS\system32\shell32.dll
\Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9\comctl32.dll
\Device\HarddiskVolume1\WINDOWS\WindowsShell.Manifest
\Device\HarddiskVolume1\WINDOWS\system32\wininet.dll
\Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\freemalwares\msrll.exe
\Device\HarddiskVolume1\WINDOWS\system32\mfm\msrll.exe
/*======================================*/

3. These files are read by the program
---
/*======================================*/

4. These files are written into by the program
---
/*======================================*/

5. The following files security permissions are changed
---
/*======================================*/

6. These files are deleted by the program
---
\Device\HarddiskVolume1\WINDOWS\system32\mfm\msrll.exe
/*======================================*/

7. These files are sent ioctl command
---
/*======================================*/

8. The timestamps are changed for the following files
---
/*=========================================*/

9. The following registry keys are created
---
\Registry\Machine\SOFTWARE\Microsoft\Cryptography\RNG
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings
/*=========================================*/

10. The following registry key values set
---
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Cryptography\RNG Seed
/*=========================================*/

11. The following registry keys are deleted
---
/*=========================================*/

12. The following processes are created
---
/*=========================================*/

/*=============================================*/

Automated analysis of the program behavior

/*=============================================*/

The program deletes itself

*********
##### The program is a little suspicious program#####
**********

## Sample#3

## A quite suspicious program. The following is the report for DoomJuice2.exe.

1. The program created the following new files
---
\Device\HarddiskVolume1\WINDOWS\system32\intrenat.exe
\Device\HarddiskVolume1\sync-src-1.00.tbz
\Device\HarddiskVolume1\WINDOWS\sync-src-1.00.tbz
\Device\HarddiskVolume1\WINDOWS\system32\sync-src-1.00.tbz
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temp\sync-src-1.00.tbz
\Device\HarddiskVolume1\Documents and Settings\Administrator\sync-src-1.00.tbz
/*====================================*/

2. The program opened the following files
---
\Device\HarddiskVolume1\WINDOWS\system32\ws2_32.dll
\Device\HarddiskVolume1\WINDOWS\system32\ws2help.dll
\Device\HarddiskVolume1\WINDOWS\system32\intrenat.exe
\Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\freemalwares\DoomJuice2.exe
\Device\HarddiskVolume1\WINDOWS\system32\wininet.dll
\Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9\comctl32.dll
\Device\HarddiskVolume1\WINDOWS\WindowsShell.Manifest
\Device\HarddiskVolume1\WINDOWS\system32\shell32.dll
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
\Device\HarddiskVolume1\Documents and Settings\Administrator\Cookies
\Device\HarddiskVolume1\Documents and Settings\Administrator\Cookies\index.dat
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History\History.IE5
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History\History.IE5\index.dat
\Device\HarddiskVolume1\WINDOWS\system32\rasapi32.dll
\Device\HarddiskVolume1\WINDOWS\system32\rasman.dll
\Device\HarddiskVolume1\WINDOWS\system32\tapi32.dll
\Device\HarddiskVolume1\WINDOWS\system32\rtutils.dll
\Device\HarddiskVolume1\WINDOWS\system32\winmm.dll
\Device\HarddiskVolume1\WINDOWS\system32\sensapi.dll
\Device\HarddiskVolume1\AUTOEXEC.BAT
\Device\HarddiskVolume1\WINDOWS\system32\mswsock.dll
\Device\HarddiskVolume1\WINDOWS\system32\dnsapi.dll
\Device\HarddiskVolume1\WINDOWS\system32\winrnr.dll
\Device\HarddiskVolume1\WINDOWS\system32\rasadhlp.dll
\Device\HarddiskVolume1\WINDOWS\system32\hnetcfg.dll
\Device\HarddiskVolume1\WINDOWS\system32\wshtcpip.dll
/*====================================*/

3. These files are read by the program
---
\Device\HarddiskVolume1\AUTOEXEC.BAT
/*=======================================*/

4. These files are written into by the program
---
\Device\HarddiskVolume1\WINDOWS\system32\intrenat.exe
\Device\HarddiskVolume1\sync-src-1.00.tbz
\Device\HarddiskVolume1\WINDOWS\sync-src-1.00.tbz
\Device\HarddiskVolume1\WINDOWS\system32\sync-src-1.00.tbz
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temp\sync-src-1.00.tbz
\Device\HarddiskVolume1\Documents and Settings\Administrator\sync-src-1.00.tbz
/*=======================================*/

5. The following files security permissions are changed
---
/*=======================================*/

6. These files are deleted by the program
---
\Device\HarddiskVolume1\WINDOWS\system32\intrenat.exe
/*=======================================*/

7. These files are sent ioctl command
---
/*=======================================*/

8. The timestamps are changed for the following files
---
\Device\HarddiskVolume1\WINDOWS\system32\intrenat.exe
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
\Device\HarddiskVolume1\Documents and Settings\Administrator\Cookies
\Device\HarddiskVolume1\Documents and Settings\Administrator\Cookies\index.dat
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History\History.IE5
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History\History.IE5\index.dat
/*=======================================*/

9. The following registry keys are created
---
\Registry\Machine\SOFTWARE\Microsoft\Cryptography\RNG
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Tracing
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Hardware
Profiles\0001\Software\Microsoft\windows\CurrentVersion\Internet Settings
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters
/*=====================================*/

10. The following registry key values set
---
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Gremlin
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Cryptography\RNG Seed
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Cache
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths
Directory
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths
Paths
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache\Paths\path1 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache\Paths\path2 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache\Paths\path3 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache\Paths\path4 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache\Paths\path1 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache\Paths\path2 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache\Paths\path3 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet
Settings\Cache\Paths\path4 CacheLimit
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Cookies
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders History
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
Common AppData
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders AppData

\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings MigrateProxy
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings ProxyEnable
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Hardware
Profiles\0001\Software\Microsoft\windows\CurrentVersion\Internet Settings ProxyEnable
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections SavedLegacySettings
/*====================================*/

11. The following registry keys are deleted
---
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings ProxyServer
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings ProxyOverride
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-
500\Software\Microsoft\Windows\CurrentVersion\Internet Settings AutoConfigURL
/*====================================*/

12. The following processes are created
---
/*====================================*/

/*============================================*/

Automated analysis of the program behavior

/*============================================*/

The program adds a program which will run automatically on next login

The program changes the internet settings

The program writes to executables

*********
##### The program is a quite suspicious program#####
**********

## Sample#4

## A very very suspicious program. The following is the report for qquzlzb.exe

1. The program created the following new files
---
\Device\HarddiskVolume1\WINDOWS\qquzlzb.exe
/*=====================================*/

2. The program opened the following files
---
\Device\HarddiskVolume1\WINDOWS\system32\crtdll.dll
\Device\HarddiskVolume1\WINDOWS\system32\wsock32.dll
\Device\HarddiskVolume1\WINDOWS\system32\ws2_32.dll
\Device\HarddiskVolume1\WINDOWS\system32\ws2help.dll
\Device\HarddiskVolume1\WINDOWS\system32\wininet.dll
\Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b64144ccf1df_6.0.2600.2180_x-ww_a84f1ff9\comctl32.dll
\Device\HarddiskVolume1\WINDOWS\WindowsShell.Manifest
\Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\freemalwares\qquzlzb.exe
\Device\HarddiskVolume1\WINDOWS\qquzlzb.exe
\Device\HarddiskVolume1\WINDOWS\Prefetch\QQUZLZB.EXE-1E41BD18.pf
\Device\HarddiskVolume1
\Device\HarddiskVolume1\WINDOWS\system32\ntdll.dll
\Device\HarddiskVolume1\WINDOWS\system32\kernel32.dll
\Device\HarddiskVolume1\WINDOWS\system32\unicode.nls
\Device\HarddiskVolume1\WINDOWS\system32\locale.nls
\Device\HarddiskVolume1\WINDOWS\system32\sorttbls.nls
\Device\HarddiskVolume1\WINDOWS\system32\advapi32.dll
\Device\HarddiskVolume1\WINDOWS\system32\rpcrt4.dll
\Device\HarddiskVolume1\WINDOWS\system32\secur32.dll
\Device\HarddiskVolume1\WINDOWS\system32\user32.dll
\Device\HarddiskVolume1\WINDOWS\system32\gdi32.dll
\Device\HarddiskVolume1\WINDOWS\system32\crypt32.dll
\Device\HarddiskVolume1\WINDOWS\system32\msvcrt.dll
\Device\HarddiskVolume1\WINDOWS\system32\msasn1.dll
\Device\HarddiskVolume1\WINDOWS\system32\oleaut32.dll
\Device\HarddiskVolume1\WINDOWS\system32\ole32.dll
\Device\HarddiskVolume1\WINDOWS\system32\shlwapi.dll
\Device\HarddiskVolume1\WINDOWS\system32\sortkey.nls
\Device\HarddiskVolume1\WINDOWS\system32\ctype.nls
\Device\HarddiskVolume1\WINDOWS\WINDOWSSHELL.MANIFEST
\Device\HarddiskVolume1\WINDOWS\system32\shell32.dll
\Device\HarddiskVolume1\WINDOWS\system32\comctl32.dll
\Device\HarddiskVolume1\WINDOWS\system32\mswsock.dll
\Device\HarddiskVolume1\WINDOWS\system32\hnetcfg.dll
\Device\HarddiskVolume1\WINDOWS\system32\wshtcpip.dll
\Device\HarddiskVolume1\WINDOWS\system32\dnsapi.dll
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
\Device\HarddiskVolume1\WINDOWS\system32\winrnr.dll
\Device\HarddiskVolume1\WINDOWS\system32\wldap32.dll
\Device\HarddiskVolume1\Documents and Settings\Administrator\Cookies\index.dat

\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History\History.IE5\index.dat
\Device\HarddiskVolume1\WINDOWS\system32\urlmon.dll
\Device\HarddiskVolume1\WINDOWS\system32\version.dll
\Device\HarddiskVolume1\WINDOWS\system32\rasapi32.dll
\Device\HarddiskVolume1\WINDOWS\system32\rasman.dll
\Device\HarddiskVolume1\WINDOWS\system32\netapi32.dll
\Device\HarddiskVolume1\WINDOWS\system32\tapi32.dll
\Device\HarddiskVolume1\WINDOWS\system32\rtutils.dll
\Device\HarddiskVolume1\WINDOWS\system32\winmm.dll
\Device\HarddiskVolume1\WINDOWS\system32\sensapi.dll
\Device\HarddiskVolume1\WINDOWS\system32\userenv.dll
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5
\Device\HarddiskVolume1\Documents and Settings\Administrator\Cookies
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History\History.IE5
\Device\HarddiskVolume1\AUTOEXEC.BAT
\Device\HarddiskVolume1\WINDOWS\system32\rasadhlp.dll
/*========================================*/

3. These files are read by the program
---
\Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\freemalwares\qquzlzb.exe
\Device\HarddiskVolume1\WINDOWS\Prefetch\QQUZLZB.EXE-1E41BD18.pf
\Device\HarddiskVolume1\WINDOWS\qquzlzb.exe
\Device\HarddiskVolume1\AUTOEXEC.BAT
/*========================================*/

4. These files are written into by the program
---
/*========================================*/

5. The following files security permissions are changed
---
/*========================================*/

6. These files are deleted by the program
---
/*========================================*/

7. These files are sent ioctl command
---
/*========================================*/

8. The timestamps are changed for the following files
---
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temporary Internet Files\Content.IE5\index.dat
\Device\HarddiskVolume1\Documents and Settings\Administrator\Cookies
\Device\HarddiskVolume1\Documents and Settings\Administrator\Cookies\index.dat
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History\History.IE5
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\History\History.IE5\index.dat
/*=====================================*/

9. The following registry keys are created
---
\Registry\Machine\SOFTWARE\Microsoft\Cryptography\RNG
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Tracing
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows NT\CurrentVersion\Winlogon
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Hardware Profiles\0001\Software\Microsoft\windows\CurrentVersion\Internet Settings
/*=====================================*/

10. The following registry key values set
---
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Cryptography\RNG Seed
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Update
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Cache
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths Directory
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths Paths
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path1 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path2 CachePath

\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path3 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path4 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path1 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path2 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path3 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path4 CacheLimit
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Cookies
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders History
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Common AppData
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders AppData
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings MigrateProxy
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings ProxyEnable
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Hardware Profiles\0001\Software\Microsoft\windows\CurrentVersion\Internet Settings ProxyEnable
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections SavedLegacySettings
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap ProxyBypass
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap IntranetName
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap UNCAsIntranet
/*=======================================*/

11. The following registry keys are deleted
---
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings ProxyServer
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings ProxyOverride
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings AutoConfigURL
/*=======================================*/

12. The following processes are created
---
most probably creates a short lived or hidden process: pid = 1464
/*=======================================*/

/*==========================================*/

Automated analysis of the program behavior

/*==========================================*/

The program adds a program which will run automatically on next login

The program changes the internet settings

*********
##### The program is a very very suspicious program#####
**********

## Sample#5

## A quite suspicious program. The following is the report for msnbot.exe

1. The program created the following new files
---
/*=======================================*/

2. The program opened the following files
---
/*=======================================*/

3. These files are read by the program
---
/*=======================================*/

4. These files are written into by the program
---
/*=======================================*/

5. The following files security permissions are changed
---
/*=======================================*/

6. These files are deleted by the program
---
/*=======================================*/

7. These files are sent ioctl command
---
/*=======================================*/

8. The timestamps are changed for the following files
---
/*=======================================*/

9. The following registry keys are created
---
\Registry\Machine\SOFTWARE\Microsoft\Cryptography\RNG

\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services\Tcpip\Parameters
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{edcc221d-ac40-11de-a14b-806d6172696f}
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{2b820c76-8d25-11df-80c4-000c29f8a390}
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{9bca3804-8d21-11df-80c3-806d6172696f}
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{edcc221a-ac40-11de-a14b-806d6172696f}
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\WinTrust\Trust Providers\Software Publishing
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
/*=======================================*/

10. The following registry key values set
---
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Cryptography\RNG Seed
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Cache
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths Directory
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths Paths
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path1 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path2 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path3 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path4 CachePath
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path1 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path2 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path3 CacheLimit
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths\path4 CacheLimit
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Cookies

\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders History
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Personal
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{edcc221d-ac40-11de-a14b-806d6172696f} BaseClass
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{2b820c76-8d25-11df-80c4-000c29f8a390} BaseClass
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{9bca3804-8d21-11df-80c3-806d6172696f} BaseClass
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2\{edcc221a-ac40-11de-a14b-806d6172696f} BaseClass
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Common Documents
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Desktop
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders Common Desktop
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap ProxyBypass
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap IntranetName
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap UNCAsIntranet
\REGISTRY\USER\S-1-5-21-854245398-113007714-682003330-500\Software\Microsoft\Windows\ShellNoRoam\MUICache c:\a.bat
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run Windows Taskmanager
/*==========*/

11. The following registry keys are deleted
---
/*==========*/

12. The following processes are created
---
most probably creates a short lived process: pid = 4864
most probably creates a short lived process: pid = 4872
/*==========*/

/*==========*/

Automated analysis of the program behavior

/*==========================================*/

The program changes the internet settings

The program adds a program which will run automatically on next login

*********
##### The program is a quite suspicious program#####
**********

**Sample#6**

A quite suspicious program. The following is the report for 440acfc139f1ea0b8e879 bf8990a0f92.exe

1. The program created the following new files
---
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temp\~DF2FEF.tmp
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\csrss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\smss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\lsass.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\services.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\winlogon.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\Paraysutki_VM_Communit
y
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\msvbvm60.dll
\Device\HarddiskVolume1\Documents and Settings\Administrator\Local Settings\Temp\~DF3B27.tmp
/*====================================*/

2. The program opened the following files
---
\Device\HarddiskVolume1\WINDOWS\system32\msvbvm60.dll
\Device\HarddiskVolume1\WINDOWS\system32\rpcss.dll
\Device\HarddiskVolume1\WINDOWS\system32\uxtheme.dll
\Device\HarddiskVolume1\WINDOWS\system32\clbcatq.dll
\Device\HarddiskVolume1\WINDOWS\system32\comres.dll
\Device\HarddiskVolume1\WINDOWS\system32\scrrun.dll
\Device\HarddiskVolume1\WINDOWS\system32\mfc42.dll
\Device\HarddiskVolume1\WINDOWS\system32\sxs.dll
\Device\HarddiskVolume1\Documents and Settings\Administrator\Desktop\malwares\440acfc139f1ea0b8e879bf8990a0f92.exe\malware.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\csrss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\smss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\lsass.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\services.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\winlogon.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\Paraysutki_VM_Communit
y
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\msvbvm60.dll
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~
\Device\HarddiskVolume1\WINDOWS\system32\apphelp.dll
\Device\HarddiskVolume1\WINDOWS\AppPatch\sysmain.sdb
/*====================================*/

3. These files are read by the program
---
\Device\HarddiskVolume1\WINDOWS\system32\scrrun.dll
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\csrss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\smss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\lsass.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\services.exe

\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\winlogon.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\Paraysutki_VM_Communit
y
\Device\HarddiskVolume1\WINDOWS\system32\msvbvm60.dll
/*====================================*/

4. These files are written into by the program
---
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\csrss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\smss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\lsass.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\services.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\winlogon.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\Paraysutki_VM_Communit
y
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\msvbvm60.dll
/*====================================*/

5. The following files security permissions are changed
---
/*====================================*/

6. These files are deleted by the program
---
/*====================================*/

7. These files are sent ioctl command
---
/*====================================*/

8. The timestamps are changed for the following files
---
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\csrss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\smss.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\lsass.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\services.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\winlogon.exe
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\Paraysutki_VM_Communit
y
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~\msvbvm60.dll
\Device\HarddiskVolume1\WINDOWS\system32\~A~m~B~u~R~a~D~u~L~
/*====================================*/

9. The following registry keys are created

---
\Registry\Machine\SOFTWARE\Microsoft\Cryptography\RNG
/*=======================================*/

10. The following registry key values set
---
\REGISTRY\MACHINE\SOFTWARE\Microsoft\Cryptography\RNG Seed
/*=======================================*/

11. The following registry keys are deleted
---
/*=======================================*/

12. The following processes are created
---
most probably creates a short lived or hidden process: pid = 400
/*=======================================*/

/*=========================================*/

Automated analysis of the program behavior

/*=========================================*/

The program writes to executables

*********
##### The program is a quite suspicious program#####
**********

# 5. CONCLUSION AND FUTUREWORK

Because of the time gap between the vulnerability that comes into existence due to the appearance of new malware and the point where a solution for the new malware is generated by the anti-virus provider, every new malware poses a serious threat to computer systems. This dynamic tool analyzes the behavior of an unknown program by executing the code in a virtual environment.

The ultimate goal of the dynamic analysis tool is to gain a quick understanding of the malicious activity performed by malicious code with the central target of minimizing the time frame between the creation of the vulnerability and generation of solution to the malware attack. This dynamic tool has many advantages. One of the main advantage is the report generated by the tool is simple and easy to understand by an analyst. All the events are listed in separate paragraph, which makes it easy to understand. Because the analysis is performed in a virtual environment, the overhead is less.

Some sophisticated malware can detect the presence of a virtual environment. In order to overcome this disadvantage, thwarting virtual environment detection techniques are implemented. These techniques work by hiding the VME (Virtual Machine Environment) artifacts in memory and VME specific processor instructions from malware.

This dynamic tool makes use of a minifilter driver for tracking the security related operating system events including Windows API functions and native kernel calls. Once a minifilter driver is provided with a program name, it notifies the tool about the activities of the program.

This dynamic tool generates reports for the program tested based on the input parameters and the execution environment. Generally all programs have some decision making branches in it, so there can be many ways a program can be executed. The path of execution depends on the input parameters and the execution environment. This tool analyses the input program for a specific path of execution. But the program can be executed in many different ways, so there is a need for extending this tool to analyze all the paths of execution. This thing can be achieved by making copies of the binaries of the program that is analyzed.

In the future this tool can be improved to include more classified and detailed reports. The tool is run on windows command prompt, it can be improved to a GUI (Graphical User Interface) based tool. Also there is a scope of creating a database to store the signatures of detected viruses and use this database whenever necessary.

## BIBLIOGRAPHY AND REFERENCES


[Andreas, Ulrich 2006] *Dynamic analysis of malicious code.* Journal of Computer Viruses, (2006) 2:67–77.

[Bellard, F] *Qemu,a fast and portable dynamic translator*. In: Usenix Annual Technical Conference, 2005.

[Christodorescu,M., Jha, S] *Static analysis of executables to detect malicious patterns.* In: Usenix Security Symposium, 2003.

[Duan H, Guan Y, Zhang J] *AMCAS: An Automatic Malicious Code Analysis System.* The Ninth International Conference on Web-Age Information Management.

[Feng M, Gupta R] *Detecting Virus Mutations Via Dynamic Matching.* ICSM 2009, Edmonton, Canada.

[Kent K, Mell P, Nusbaum J] *Guide to Malware Incident Prevention and Handling.* Special Publication 800-83, NIST.

[Kirda E, Ulrich 2006] *TTAnalyze: A Tool for Analyzing Malware.* Ikarus Software & Technical University of Vienna.

[Microsoft-1] Microsoft Filter Drivers, Available from http://www.microsoft.com/whdc/driver/filterdrv/default.mspx

[Microsoft-2] Microsoft Filter Manager, Available from

[Microsoft-3] Microsoft INF file, Available from http://msdn.microsoft.com/en-us/library/ms924764.aspx

[Microsoft-4] Microsoft Load Order Groups, Available from http://www.microsoft.com/whdc/driver/filterdrv/alt-range.mspx

[Microsoft-5] Microsoft Minifilter Architecture, Available from http://msdn.microsoft.com/en-us/library/ff541613(v=VS.85).aspx

[Microsoft-6] MSDN Simrep Filters, Available from http://msdn.microsoft.com/en-us/library/ff556746(VS.85).aspx

[Microsoft-7] Microsoft WDK, Available from http://www.microsoft.com/whdc/DevTools/WDK/WDKpkg.mspx

[Microsoft-8] Microsoft Windows Service, Available from http://support.microsoft.com/kb/251192

[OSR 2010] OSR Driver Development, Available from http://www.osronline.com/custom.cfm?name=login_joinok.cfm

[Offensive Computing 2009] *Offensive Computing*. http://www.offensivecomputing.net

[Quist D, Val Smith] *Detecting the Presence of Virtual Machines Using the Local Data Table.* Offensive Computing http://www.offensivecomputing.net/

[Redpill] *Redpill* Available from http://www.invisiblethings.org/papers/redpill.html

[Rothman M, Zimmer V] *Virus Scanning of Input/Output Traffic of a Computer System.* Unites States Patent Application Publication, Pub No. US 2005/0216759 A1.

[Symantec 2008] Symantec Auraax, Available from http://www.symantec.com/security_response/writeup.jsp?docid=2008-092409-4704-99&tabid=2

[Wiki 2009] Wikipedia, Available from www.wikipedia.com/

# APPENDIX A. STARTING A VIRTUAL OPERATING SYSTEM

1. Start ⟶ All Programs⟶ VMWare⟶ VMWare⟶ Workstation

The figure below shows how to start a virtual operating system.

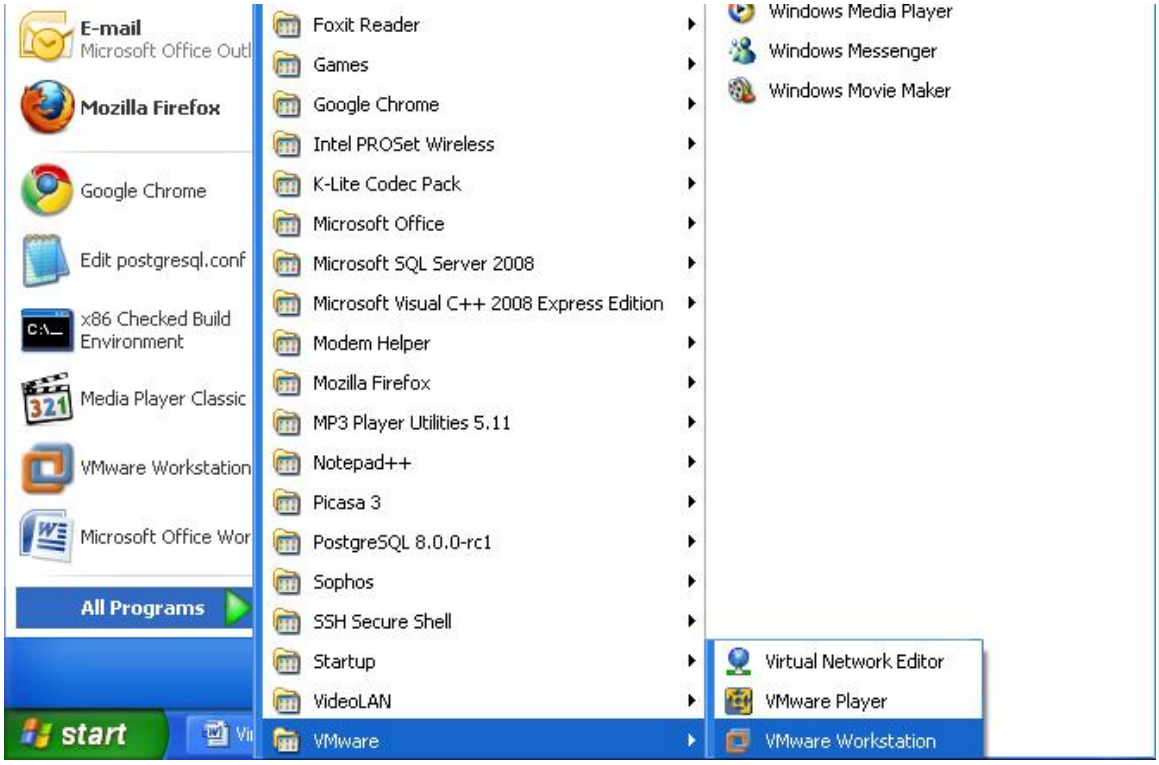


Figure A.1 Starting Virtual Operating System

2. File⟶ Open and then select a virtual operating system which is to be loaded

3. Select “Power on this Virtual Machine”

   The figure below shows how to select a particular virtual machine.

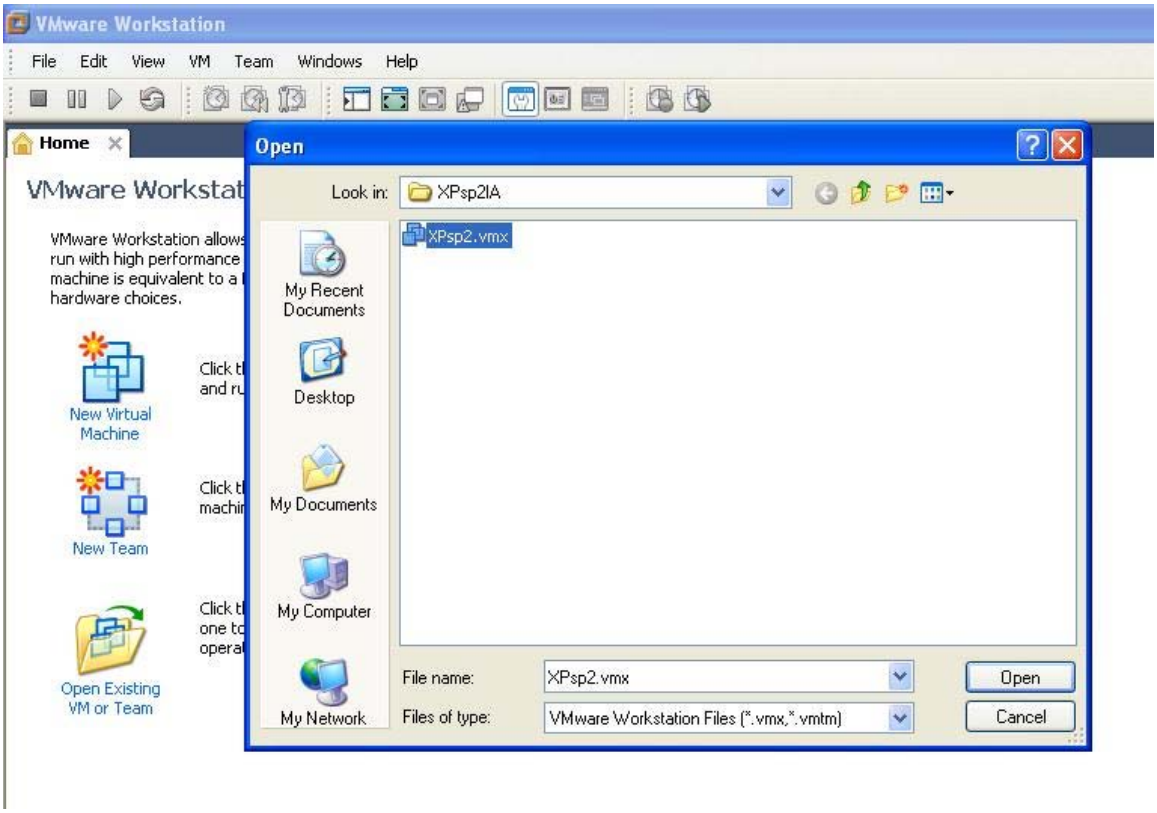


Figure A.2 Selecting a Virtual Disk

4. This start the Virtual Machine after the login credentials are specified

## APPENDIX B. INSTALLING MINIFILTER

1. Go to the project folder
2. Open the directory /Malware Analyzer/Driver
3. Install the minifilter drivers by right clicking the matool.inf and selecting “install” This step installs the drivers and places files required by the project in the correct destinations. The figure below shows how to install a mini filter driver.

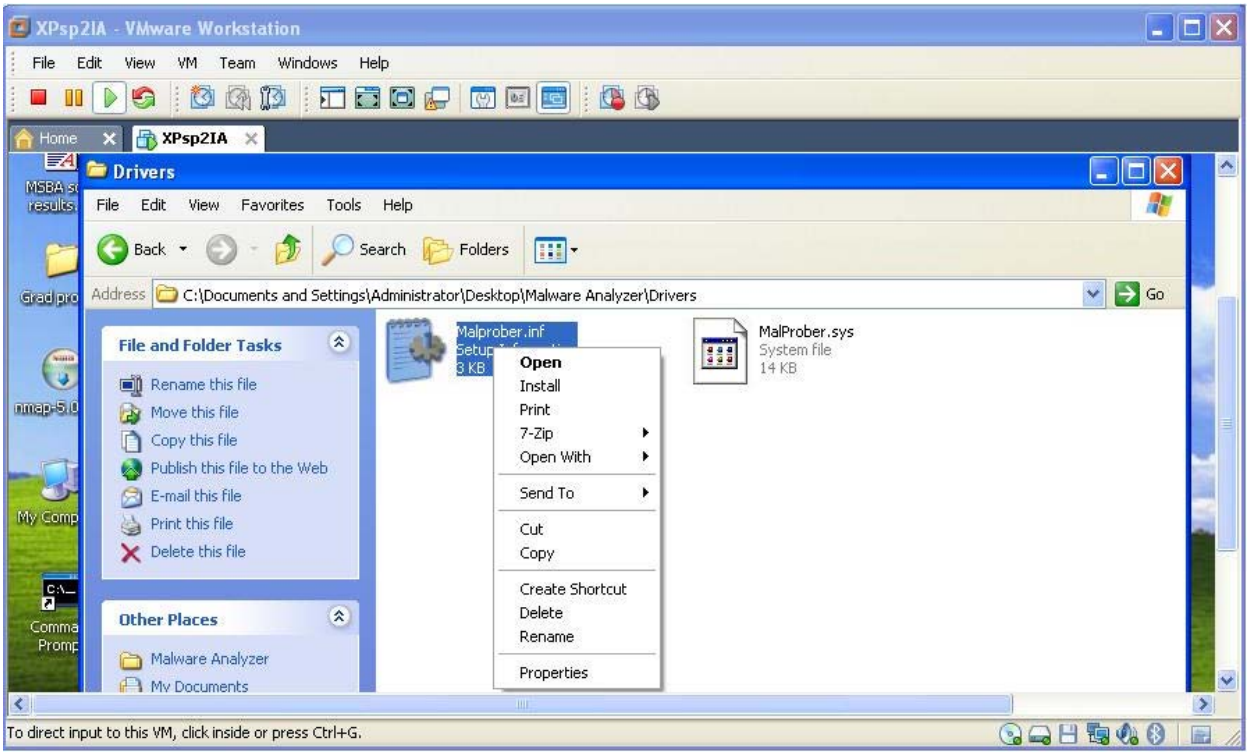


Figure B.1 Installing Minifilter Driver

## APPENDIX C. STARTING THE TOOL

**1.** Open a command prompt on the virtual Operating System.

**2.** Type the following command to start the tool

Command: sc start malprober

This command makes use of the sc service of Windows. This service can be used to start services on the Windows Operating System [Microsoft-8]. The figure below shows that a minifilter service is started.

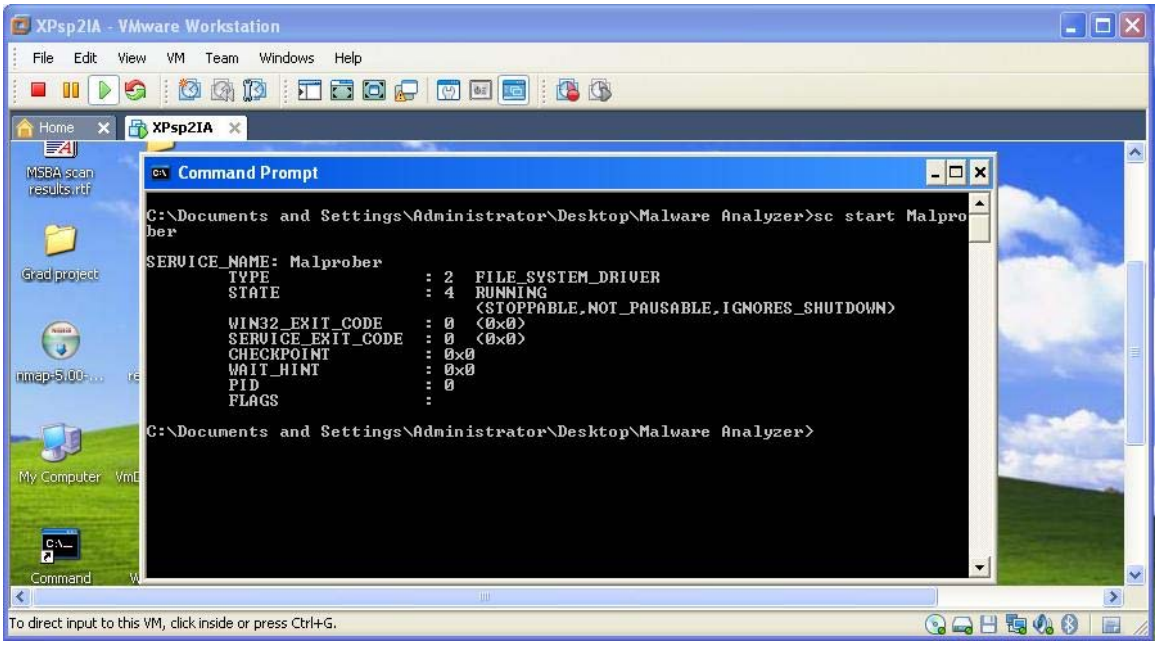


Figure C.1 Starting the Malprober Service

## APPENDIX D. TESTING A SAMPLE PROGRAM

1. After starting the tool, navigate to the project folder that has the u_malanalyze.exe file. This is the main executable of the tool.
2. Place the sample program to be tested in the same folder as u_malanalyze.exe is in.
3. Type the following command for testing the sample program and generating reports.

    Command: u_malanalyze.exe <sample program executable>

Sample program is the executable that is to be analyzed.

These reports can be analyzed to classify the nature of the given sample program as malicious or a normal program

## APPENDIX E. INF FILE FOR THE MINIFILTER

```
;;;
;;; PassThrough
;;;
;;;
;;; Copyright (c) 1999 - 2001, Microsoft Corporation
;;;
[Version]
Signature       = "$Windows NT$"
Class           = "ActivityMonitor"
;This is determined by the work this filter driver does
ClassGuid       = {b86dff51-a31e-4bac-b3cf-e8cfe75c9fc2}
;This value is determined by the Class
Provider        = %vat%
DriverVer       = 06/16/2007,1.0.0.1
CatalogFile     = passthrough.cat


[DestinationDirs]
DefaultDestDir         = 12
MiniFilter.DriverFiles = 12            ;%windir%\system32\drivers
;;
;; Default install sections
;;
```

```
[DefaultInstall]
OptionDesc          = %ServiceDescription%
CopyFiles           = MiniFilter.DriverFiles

[DefaultInstall.Services]
AddService          = %ServiceName%,,MiniFilter.Service
;;
;; Default uninstall sections
;;

[DefaultUninstall]
DelFiles   = MiniFilter.DriverFiles

[DefaultUninstall.Services]
DelService = %ServiceName%,0x200
;Ensure service is stopped before deleting
;
; Services Section
;

[MiniFilter.Service]
DisplayName           = %ServiceName%
Description           = %ServiceDescription%
ServiceBinary         =%12%\%DriverName%.sys
;%windir%\system32\drivers\
Dependencies          = "FltMgr"
```

```
ServiceType     = 2                  ;SERVICE_FILE_SYSTEM_DRIVER
StartType       = 3                  ;SERVICE_DEMAND_START
ErrorControl    = 1                 ;SERVICE_ERROR_NORMAL
LoadOrderGroup  = "FSFilter Activity Monitor"
AddReg          = MiniFilter.AddRegistry
;
; Registry Modifications
;

[MiniFilter.AddRegistry]
HKR,,"DebugFlags",0x00010001 ,0x0
HKR,"Instances","DefaultInstance",0x00000000,%DefaultInstance%
HKR,"Instances\"%Instance1.Name%,"Altitude",0x00000000,%Instance1
.Altitude%
HKR,"Instances\"%Instance1.Name%,"Flags",0x00010001,%Instance1.Fl
ags%
;
; Copy Files
;

[MiniFilter.DriverFiles]
%DriverName%.sys
```

```
[SourceDisksFiles]
passthrough.sys = 1,,

[SourceDisksNames]
1 = %DiskId1%,,,
;;
;; String Section
;;

[Strings]
vat                     = "Vinay@TAMUCC"
ServiceDescription      = "Malprober Mini-Filter Driver"
ServiceName             = "Malprober"
DriverName              = "Malprober"
DiskId1                 = "PassThrough Device Installation Disk"

;Instances specific information.
DefaultInstance         = "PassThrough Instance"
Instance1.Name          = "PassThrough Instance"
Instance1.Altitude      = "370030"
Instance1.Flags         = 0x0          ; Allow all attachments
```