Lab 5

Introduction to Malware analysis

Contents

5.1 Objectives

In this session, we will introduce some common approach to analyse a malware, so that you can turn malicious executable inside out to understand their inner-workings (technique known as reverse engineering). Knowing how to analyse a malware can bring an element of control into an otherwise chaotic environment that exists around a security incident. It is also a critical aspect of modern forensic analysis as it is frequent for investigators to discover malware on compromised systems.

The approach followed in this session is reverse-engineering that has worked for many analysts; it involves two key phases: behavioural analysis and code analysis. During behavioural analysis, we examine how the malware interacts with its environment. The code analysis phase allows us to learn about the specific capabilities by examining the code of the compromised program.

5.2 Laboratory environment

When performing malware analysis, it is convenient to stay in a virtualisation environment. Such tools typically simulate the underlying hardware, allowing you to run multiple instances of virtual machines simultaneously. For instance, you could use Windows 7 as your base OS, while having a separate instance of Windows XP running in another window, and a Linux instance running in another one.

Each virtual machine behaves mostly as *real* physical systems, in that it has its own set of I/O peripherals, RAM, network settings, and so on. All these aspects of the virtual machine are indeed virtualised.

The convenience of a virtualised lab comes, in part, from the flexibility of having multiple instances of various operating systems available to you within a single physical system. Virtualisation software can even emulate a network, so that your lab does not need to be connected to a physical network at all. Yet, the virtual machines will be able to communicate with each other over the simulated network, blissfully unaware that the network is not "real".

For this session we will use the VMware Player software. To do this, open the virtual machine called "malware" from the link provided in the presentation and start it. If the VMware Player asks, answer that "you copied the virtual machine". The password of the administrator user is **si2012**.

5.3 The malware: a trojan copy of a windows live messenger

The malicious executable which we will learn from in this session is shown in Figure 5.1 . It is a trojan copy of Windows Live Messenger - a fake instant messenger client that was being distributed to victims via email. Many such trojans have the capability of capturing the victims' logon credentials, and may have other *undocumented* features.

In this lab, we will assume that we have got the trojan executable file and the *msnsettings.dat* file from the victim's PC.

Let's see what capabilities are built into this malicious executable. First, let's introduce the tools and techniques that will help with the reverse-engineering process.

Note that in this example, as with the majority of malicious incidents you will probably encounter, we will be examining a compiled Windows executable for which we have no source code.

5.4 Behavioural analysis

Malware analysis typically starts examining a malicious executable with behavioural analysis as it is easier than code analysis and provides some hints for it.

When performing behavioural analysis, we are going to infect a laboratory system with the malware. Then we will observe how the malicious executable accesses the file system, the registry, and the network. As we learn about the program's expectations of its runtime environment, we will slightly adjust our analysis to evoke additional behaviour from the program. We will also attempt to interact with the program to discover additional characteristics it may exhibit.

Let's see this approach in action. Imagine you have a suspicious executable that you would like to analyse. You bring it into your lab, possible via a removable USB

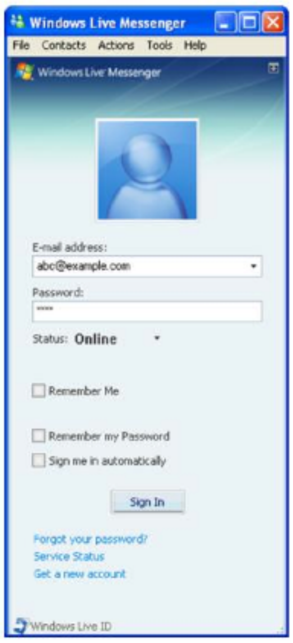


Figure 5.1: Windows live messenger snapshot.

disk and place it on the desktop of the virtual machine[1] you are about to infect. Now what?

First, take a snapshot of the state of the machine's file system and the registry. This will allow you to quickly see what major changes have occurred on the system after you infect it.

To do this we will use the free tool called **RegShot** (http://sourceforge.net/projects/regshot). RegShot is already installed in your VM. To use it, enable the "Scan dir1" option, and in the corresponding window type "C:\". This will allow the tool to scan the registry and the full C: drive. Click now "1st shot".

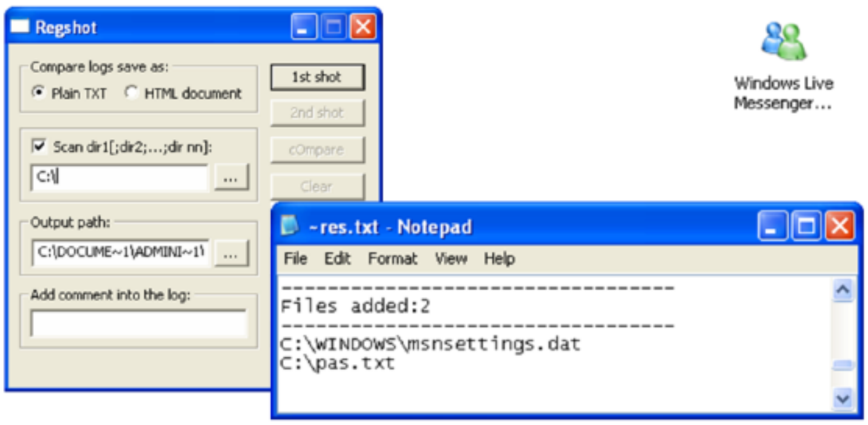
After RegShot takes the first snapshot, launch the malicious executable. Interact with it a bit (e.g., try logging into it). Then kill the process, if you can. Next, click the "2nd shot" button in RegShot, and click the "Compare" button. You will see a report that describes the major changes to the system's state. In this case, we see that, among other things, two files were added to the system.

The two files that appeared on the system after we infected it are *pas.txt* and *msnsettings.dat*. Take a look at them using notepad.

It looks like *pas.txt* has captured the login credentials we used when logging into the malicious executable. That makes sense, because we received reports that this executable is a trojan copy of Windows Live Messenger.

The *msnsettings.dat* file looks like a configuration file of some sort.

[1]The fake windows live messenger is already copied in your VM image



Compare logs save as: [✓ Plain TXT] HTML document

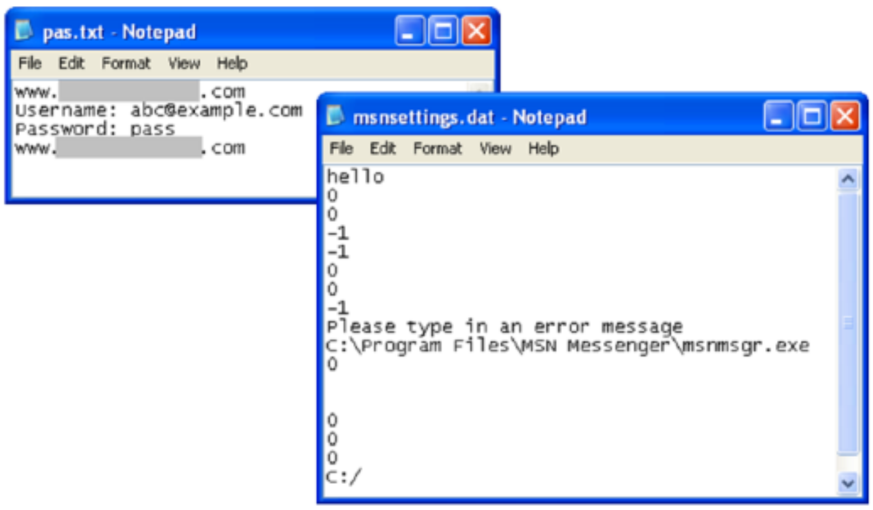
[✓] Scan dir1[;dir2;...;dir nn]: C:\

Output path: C:\DOCUME~1\ADMINI~1\

Add comment into the log:

```
----------------------------------
Files added:2
----------------------------------
C:\WINDOWS\msnsettings.dat
C:\pas.txt
```

Figure 5.2: The RegShot tool and its output.

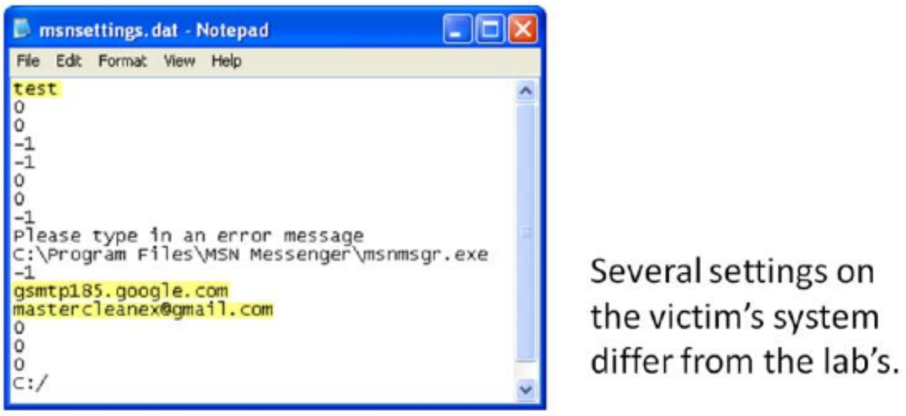

pas.txt - Notepad
File Edit Format View Help
msnsettings.dat - Notepad
File Edit Format View Help

```
www.[redacted].com
Username: abc@example.com
Password: pass
www.[redacted].com
```

```
hello
0
0
-1
-1
0
0
-1
Please type in an error message
C:\Program Files\MSN Messenger\msnmsgr.exe
0


0
0
0
C:/
```

Figure 5.3: The two files created by the Windows live messenger malware.


msnsettings.dat - Notepad
File Edit Format View Help

```
test
0
0
-1
-1
0
0
-1
Please type in an error message
C:\Program Files\MSN Messenger\msnmsgr.exe
-1
gsmtp185.google.com
mastercleanex@gmail.com
0
0
0
C:/
```

Several settings on the victim's system differ from the lab's.

Figure 5.4: Windows live messenger snapshot.

With the obtained information, reverse-engineering the malware can help you at incident response and forensic analysis. In our scenario, we have already discovered that Windows Live Messenger trojan makes use of the *msnsettings.dat* file. Now you know to look for it on the compromised system, even if you did not initially realise that this file was important.

Once you have a copy of *msnsettings.dat*, you can open it to see whether it reveals additional details about the program. It should appear like the one reported in Fig. 5.3. It seems that the malware uses *msnsettings.dat* as a configuration file.

Now, let's take a look at the *msnsettings.dat* inside the *live-messenger-malware* folder on the desktop. This file proceeds from the victim of the fake messenger malware. It is shown in Fig. 5.4, in which we have highlighted some lines.

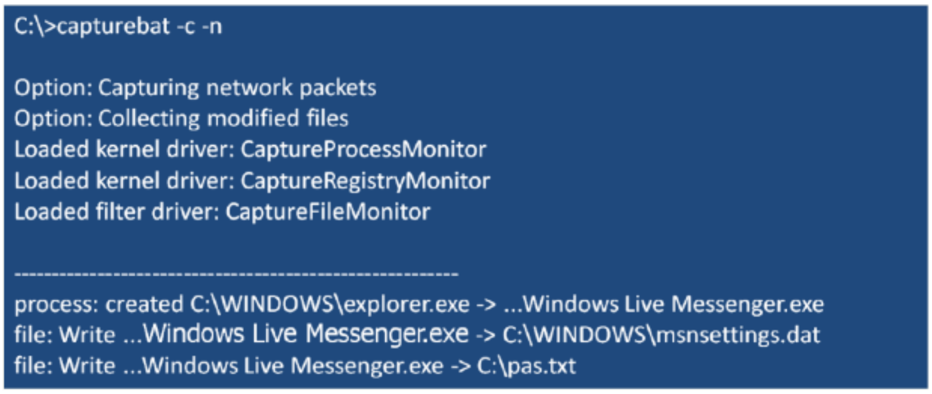
Let's close the fake messenger and **replace the *msnsettings.dat* in the Windows folder with the victim's one.**

In the first line there is a string "test", which we may be able to use later when trying to understand how the trojan processes the *msnsettings.dat* file. Another line, "gsmtp185.google.com" specifies an SMTP mail server; this suggests that the malware has the ability to send email. The file also includes an email address, "mastercleanex@gmail.com". This may be the recipient of the information that the trojan might attempt to send out. Of course, these are just theories at this point. We will need to confirm or deny them during subsequent analysis steps.

5.5 Network traffic analysis

Our next step is to confirm if the fake Microsoft MSN sends some SMTP messages. To do this, we have two options: **CaptureBAT** and **Wireshark**. You can launch the tool you prefer, both options will provide the same conclusions, and afterwards execute the malware again with the victim's *msnsettings.dat*.

Warning: before capturing network traffic, flush the DNS cache (in windows



```
C:\>capturebat -c -n

Option: Capturing network packets
Option: Collecting modified files
Loaded kernel driver: CaptureProcessMonitor
Loaded kernel driver: CaptureRegistryMonitor
Loaded filter driver: CaptureFileMonitor

--------------------------------------------------------
process: created C:\WINDOWS\explorer.exe -> ...Windows Live Messenger.exe
file: Write ...Windows Live Messenger.exe -> C:\WINDOWS\msnsettings.dat
file: Write ...Windows Live Messenger.exe -> C:\pas.txt
```

Figure 5.5: CaptureBat output snapshot.

command line mode, execute "ipconfig /flushdns"; you can also flush all the network related caches by going in the Control Panel -> Network settings -> Network connections-> double click on the network connection, select the Support tab and then click on Repair').

5.5.1 CaptureBAT

CaptureBat is a very useful and free tool available at http://www.honeynet.org/projects/ols/capture-bat.

CaptureBAT [2] records local processes' interactions with their environment. CaptureBAT's logs are quite easy to follow and understand because it comes with filters that eliminate the majority of standard, non-malicious activities from the logs. You can customise these filters to your liking, as they are text files located in the directory where you install CaptureBAT.

If you launch CaptureBAT [3] with the "-c" parameter, it will capture any files deleted in the background, allowing you to look at and restore even those files that the Windows Recycle Bin cannot capture.

Launching CaptureBAT with the "-n" parameter tells the tool to capture network traffic, like a sniffer would, saving the result into a local .cap file.

Warning! Do not terminate CaptureBAT with "ctrl+c", CaptureBAT dumps the data on the disk at the end of its execution. Use "enter" to terminate the CaptureBAT process.

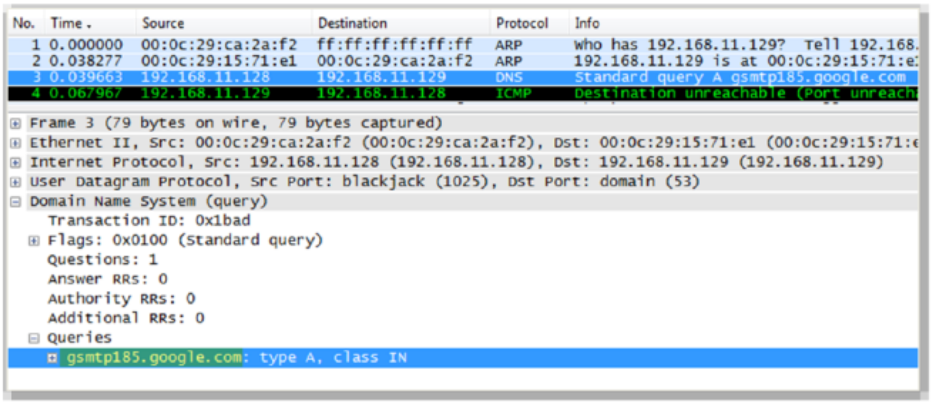
Open the .cap file in WireShark. As you can see on the next figure, CaptureBAT confirmed our earlier findings about the malware [4].

If you do not like using CaptureBAT, you could use Wireshark to capture the traffic sent by the malware.

[2]Install CaptureBat and reboot the system

[3]In command line mode, go to C:\Program Files\Capture

[4]If it does not show the DNS query we're looking for, make sure that you have substituted the *msnsettings.dat* in the Windows folder with the one that came in the Desktop folder



```
No.  Time .    Source             Destination        Protocol  Info
  1 0.000000   00:0c:29:ca:2a:f2  ff:ff:ff:ff:ff:ff  ARP       who has 192.168.11.129?  Tell 192.168.
  2 0.038277   00:0c:29:15:71:e1  00:0c:29:ca:2a:f2  ARP       192.168.11.129 is at 00:0c:29:15:71:e
  3 0.039663   192.168.11.128     192.168.11.129     DNS       Standard query A gsmtp185.google.com
  4 0.067967   192.168.11.129     192.168.11.128     ICMP      Destination unreachable (Port unreach

⊞ Frame 3 (79 bytes on wire, 79 bytes captured)
⊞ Ethernet II, Src: 00:0c:29:ca:2a:f2 (00:0c:29:ca:2a:f2), Dst: 00:0c:29:15:71:e1 (00:0c:29:15:71:e
⊞ Internet Protocol, Src: 192.168.11.128 (192.168.11.128), Dst: 192.168.11.129 (192.168.11.129)
⊞ User Datagram Protocol, Src Port: blackjack (1025), Dst Port: domain (53)
⊟ Domain Name System (query)
    Transaction ID: 0x1bad
  ⊞ Flags: 0x0100 (Standard query)
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⊟ Queries
    ⊞ gsmtp185.google.com: type A, class IN
```

The hostname suggests SMTP, but use DNS resolution to confirm.

Figure 5.6: WireShark output snapshot.

5.5.2 Wireshark

Wireshark is a full-feature network sniffer (`http://www.wireshark.org`). Instead of using CaptureBAT, you can directly open Wireshark, and run the sniffer over the network interface using the appropriate filters.

5.5.3 Analysis

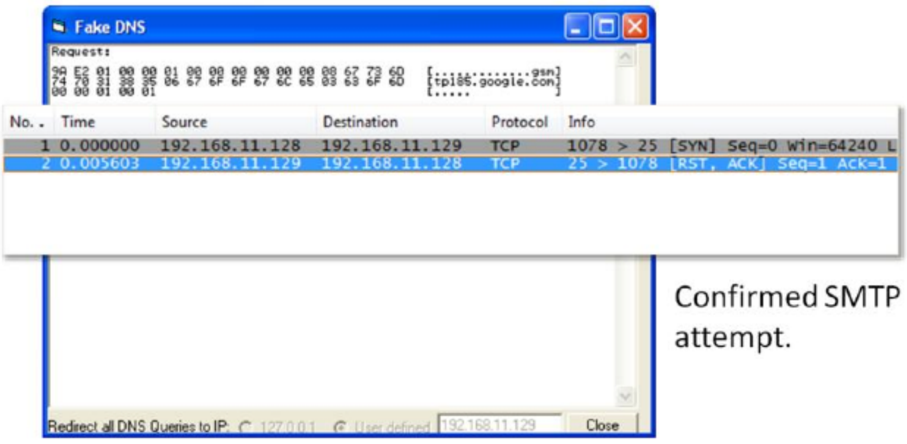
As you can see on the dump reported in the Fig. 5.6, the sniffer shows that the infected system has issued a DNS query, attempting to resolve the hostname "gsmtp185.google.com". The SMTP in the hostname suggests that the malware is looking for a mail server to connect to, reinforcing our earlier theory of how the trojan might use this hostname.

We will therefore try to intercept any e-mail that the fake messenger will possibly send to that SMTP server. To this end, we will

1. Redirect DNS queries to localhost: you can do this in the properties of the Windows Network Connection in the Control Panel; Go to TCP/IP properties and set 127.0.0.1 as the primary DNS server;

2. Capture any DNS request with **Fake DNS**. Fake DNS is a DNS server that you can configure to answer any DNS query with a single IP address of your choice[5]. Which IP address should you use? The easiest option is to use again "localhost". This will redirect any DNS query to localhost; therefore, a possible connection to "gsmtp185.google.com" on port 25 will be redirected to "localhost:25". Put Fake DNS to "Listen" mode.

[5]An alternative way, is to set up name resolution by inserting an entry for the hostname into the 'hosts' file on the infected system; Fake DNS is a faster alternative



```
Request:
9A E2 01 00 00 01 00 00 00 00 00 00 08 67 73 6D    [..............gsm]
74 70 31 38 35 06 67 6F 6F 67 6C 65 03 63 6F 6D    [tp185.google.com]
00 00 01 00 01                                     [.....           ]
```

```
No. . Time      Source          Destination     Protocol Info
   1 0.000000  192.168.11.128  192.168.11.129  TCP      1078 > 25 [SYN] Seq=0 Win=64240 L
   2 0.005603  192.168.11.129  192.168.11.128  TCP      25 > 1078 [RST, ACK] Seq=1 Ack=1
```

Redirect all DNS Queries to IP: ( ) 127.0.0.1 (•) User defined 192.168.11.129

Figure 5.7: Fake DNS output snapshot.

3. Set up an SMTP listener on port 25, capturing any mail will be sent. An easy way to do this is to use the **Mailpot** tool. Mailpot pretends to be a mail server, happily accepting SMTP messages from clients, but not sending them out. Instead, it stores the messages locally for your review. To use Mailpot, run it on the host to which you have redirected the SMTP requests (localhost) using Fake DNS and put it to 'Listen' mode.

Now that you have set up the DNS server as localhost (in Windows network connection settings), are replying the DNS request with localhost and are listening the SMTP connection on port 25, you are ready to try to login with the fake MSN and see what happens.

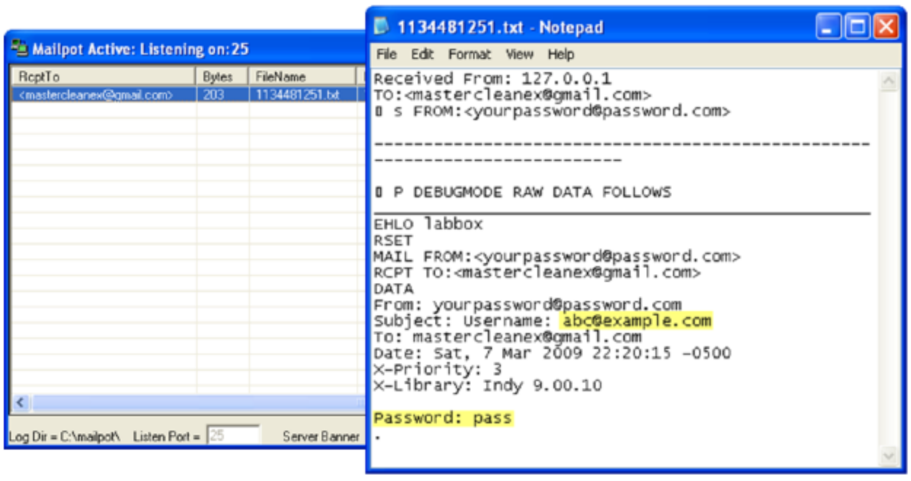
Now you should see both the DNS request and response in the Fake DNS window, and in the Mailpot window there should be the email that has been sent by the fake MSN and captured by Mailpot.

If you open the mail[6], you can see the contents of the message that the trojan is mailing to the attacker. As highlighted in the Fig. 5.8, the message includes the victim's Messenger username and password.

5.6 Code analysis

Behavioural analysis can be insightful and relatively fast. However, it will rarely tell you everything you need to know about malware of moderate and advanced complexity. That is where code analysis can be of help. It can help to reinforce

[6]If the mail does not open (by double clicking on it), you have to (1) close Mailpot, change the permissions of the C:\mailpot folder to writeable (remove the thick from "Read-only") and (3) start Mailpot again; you may have to repeat this procedure a couple of times; if it doesn't work yet, you can reboot the system and try again (remember to reactivate the Fake DNS)



RcptTo	Bytes	FileName
<mastercleanex@gmail.com>	203	1134481251.txt

```
Received From: 127.0.0.1
TO:<mastercleanex@gmail.com>
  s FROM:<yourpassword@password.com>

----------------------------------------------------
------------------------

  P DEBUGMODE RAW DATA FOLLOWS
____________________________________________________
EHLO labbox
RSET
MAIL FROM:<yourpassword@password.com>
RCPT TO:<mastercleanex@gmail.com>
DATA
From: yourpassword@password.com
Subject: Username: abc@example.com
To: mastercleanex@gmail.com
Date: Sat, 7 Mar 2009 22:20:15 -0500
X-Priority: 3
X-Library: Indy 9.00.10

Password: pass
.
```

Figure 5.8: Mailpot output snapshot.

your behavioural findings, and can shine a light on additional properties of the malware that you may not have discovered behaviourally.

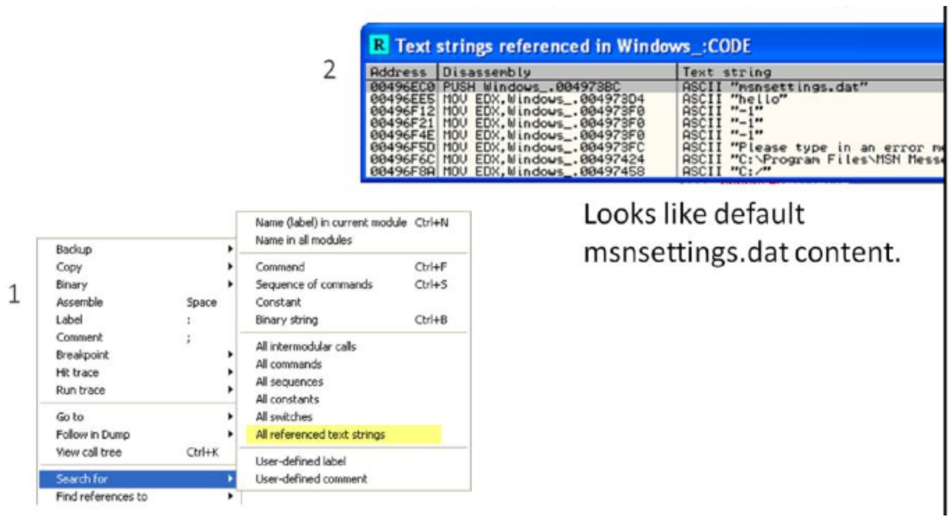
Code analysis can be tricky and time-consuming, because in the world of malware you almost never have the luxury of seeing the source code of the program you are analysing. Instead, you need to reverse-engineer the compiled executable's functionality by examining its code at the assembly level. A debugger and a disassembler can help you in this task. A disassembler converts the malware's instructions from their binary form into the human-readable assembly form. A debugger lets you step into the most interesting parts of the code, interacting with it and observing the effects of its instructions to understand their purpose.

We will use **OllyDbg** to perform code analysis. It is free, and includes both a disassembler and a debugger. You can download OllyDbg from: `http://www.ollydbg.de/`

A good way to start analysing the malware's code often involves looking at the strings embedded in its executable. To do this with OllyDbg, first load the malicious executable into OllyDbg via File -> Open. Then, right-click on the code you will see in the disassembler window, and select Search for -> All referenced text strings.

OllyDbg will then bring up a new window that will show the strings it discovered, as you can see on this slide. Notice that we have seen some of these strings during behavioural analysis! Some of them look like contents of the default *msnsettings.dat* file that our malware creates when infecting the system.

The reason we may be interested in looking at the embedded strings is because the string listing might include a reference to a malicious characteristic or a behavioural trait that we would like to understand. In this case, consider the following screenshot. We got here by highlighting one of the instances of *msnsettings.dat* strings, if you press Enter, OllyDbg shows us how the program makes use of this



```
Text strings referenced in Windows_:CODE
Address  Disassembly                        Text string
00496EC0 PUSH Windows_.004973BC             ASCII "msnsettings.dat"
00496EE5 MOV EDX,Windows_.004973D4          ASCII "hello"
00496F12 MOV EDX,Windows_.004973F0          ASCII "-1"
00496F21 MOV EDX,Windows_.004973F0          ASCII "-1"
00496F4E MOV EDX,Windows_.004973F0          ASCII "-1"
00496F5D MOV EDX,Windows_.004973FC          ASCII "Please type in an error m
00496F6C MOV EDX,Windows_.00497424          ASCII "C:\Program Files\MSN Mess
00496F8A MOV EDX,Windows_.00497458          ASCII "C:/"
```

Looks like default msnsettings.dat content.

Figure 5.9: A snapshot from OllyDbg.

string.

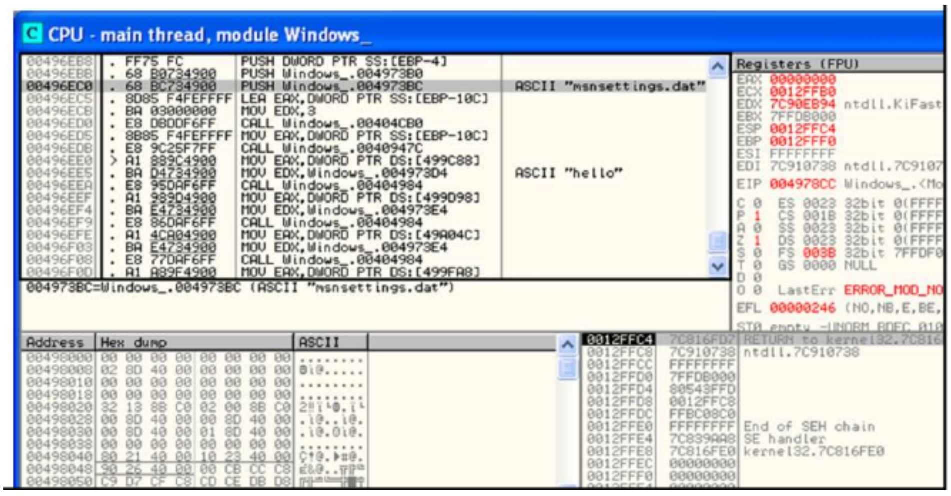
If we wanted to pursue this path of analysis further, we could now set a breakpoint on this command, run the trojan in the debugger, and see what it does. But, we are not going to investigate this particular aspect of the malicious program, because we will focus on another, more interesting technique.

You may recall that the version of *msnsetting.dat* on the victim's system was slightly different from the version that the trojan created on our laboratory system when we first ran it. Specifically, in our case, the file contained the string "hello", while the victim's version had the string 'test' instead. What's that about?

The string "test" is not visible anywhere within the body of the malicious executable when it's not running. That's probably because the trojan loads this string from *msnsettings.dat* during run time. To understand how the trojan uses the string "test", we will search for it in the memory of the running trojan.

Once we will have located the string in the trojan's memory (described further), we will set an access breakpoint there. A breakpoint is a condition that tells the debugger when to pause the normal execution of the debugged program. Once the execution is paused, the debugger will give us a chance to review the debugged program's run time environment to understand what it is doing. This is probably the most useful feature of a debugger in the context of malware reverse-engineering.

To make use of this technique, load the malicious program into OllyDbg, then run it into the debugger. Once the trojan is running, press "Alt+M" to bring up the memory map in OllyDbg. This shows the listing of the memory segments mapped and used by the currently-debugged executable. To search the executable's memory for a particular string, select the first line in the Memory Map window, and press "Ctrl+B"; then, enter your string as ASCII text field in the dialog box. Then press Enter.



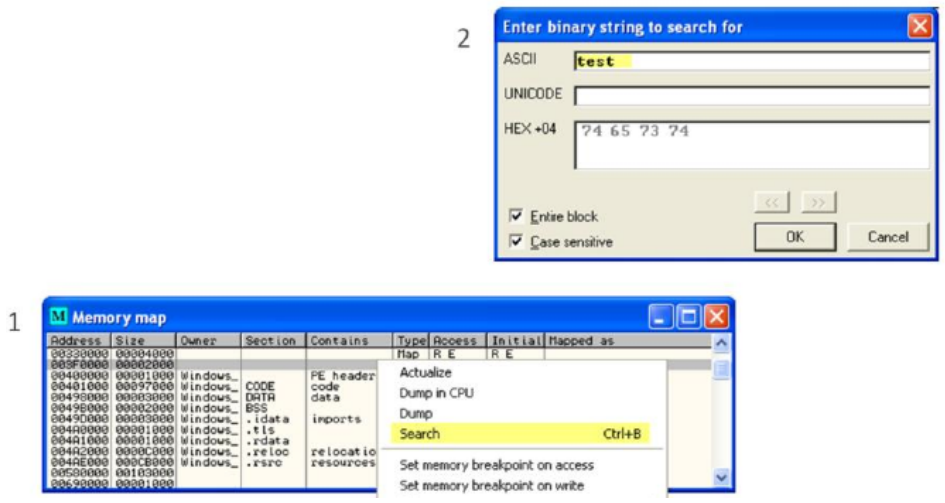
```
00496EB8 . FF75 FC           PUSH DWORD PTR SS:[EBP-4]
00496EBB . 68 B0734900       PUSH Windows_.004973B0
00496EC0 . 68 BC734900       PUSH Windows_.004973BC          ASCII "msnsettings.dat"
00496EC5 . 8D85 F4FEFFFF     LEA EAX,DWORD PTR SS:[EBP-10C]
00496ECB . BA 03000000       MOV EDX,3
00496ED0 . E8 DBDDF6FF       CALL Windows_.00404CB0
00496ED5 . 8B85 F4FEFFFF     MOV EAX,DWORD PTR SS:[EBP-10C]
00496EDB . E8 9C25F7FF       CALL Windows_.0040947C
00496EE0 > A1 889C4900       MOV EAX,DWORD PTR DS:[499C88]
00496EE5 . BA D4734900       MOV EDX,Windows_.004973D4       ASCII "hello"
00496EEA . E8 95DAF6FF       CALL Windows_.00404984
00496EEF . A1 989D4900       MOV EAX,DWORD PTR DS:[499D98]
00496EF4 . BA E4734900       MOV EDX,Windows_.004973E4
00496EF9 . E8 86DAF6FF       CALL Windows_.00404984
00496EFE . A1 4CA04900       MOV EAX,DWORD PTR DS:[49A04C]
00496F03 . BA E4734900       MOV EDX,Windows_.004973E4
00496F08 . E8 77DAF6FF       CALL Windows_.00404984
00496F0D . A1 A89F4900       MOV EAX,DWORD PTR DS:[499FA8]
004973BC=Windows_.004973BC (ASCII "msnsettings.dat")
```

```
Registers (FPU)
EAX 00000000
ECX 0012FFB0
EDX 7C90EB94 ntdll.KiFast
EBX 7FFDB000
ESP 0012FFC4
EBP 0012FFF0
ESI FFFFFFFF
EDI 7C910738 ntdll.7C9107
EIP 004978CC Windows_.<Mo
C 0  ES 0023 32bit 0(FFFF
P 1  CS 001B 32bit 0(FFFF
A 0  SS 0023 32bit 0(FFFF
Z 1  DS 0023 32bit 0(FFFF
S 0  FS 003B 32bit 7FFDF0
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_MOD_NO
EFL 00000246 (NO,NB,E,BE,
ST0 empty -UNORM BDEC 010
```

```
Address  Hex dump                ASCII
00498000 00 00 00 00 00 00 00 00 ........
00498008 02 8D 40 00 00 00 00 00 ☻ì@.....
00498010 00 00 00 00 00 00 00 00 ........
00498018 00 00 00 00 00 00 00 00 ........
00498020 32 13 8B C0 02 00 8B C0 2‼ï└☻.ï└
00498028 00 8D 40 00 00 8D 40 00 .ì@..ì@.
00498030 00 8D 40 00 01 8D 40 00 .ì@.☺ì@.
00498038 00 00 00 00 00 00 00 00 ........
00498040 80 21 40 00 10 23 40 00 Ç!@.►#@.
00498048 90 26 40 00 00 CB CC C8 É&@..╦╠╚
00498050 C9 D7 CF C8 CD CE DB D8 ╔╫╧╚═╬█╪
```

```
0012FFC4  7C816FD7  RETURN to kernel32.7C816
0012FFC8  7C910738  ntdll.7C910738
0012FFCC  FFFFFFFF
0012FFD0  7FFDB000
0012FFD4  80543FFD
0012FFD8  0012FFC8
0012FFDC  FFBC08C0
0012FFE0  FFFFFFFF  End of SEH chain
0012FFE4  7C839AA8  SE handler
0012FFE8  7C816FE0  kernel32.7C816FE0
0012FFEC  00000000
0012FFF0  00000000
```

Figure 5.10: A snapshot from OllyDbg.


2
Enter binary string to search for
1
Memory map
Actualize
Dump in CPU
Dump
Search Ctrl+B
Set memory breakpoint on access
Set memory breakpoint on write
OK
Cancel

ASCII: test
UNICODE:
HEX +04: 74 65 73 74
[✓] Entire block
[✓] Case sensitive

```
Address  Size     Owner    Section Contains   Type Access Initial Mapped as
00330000 00004000                             Map  R E    R E
003F0000 00002000
00400000 00001000 Windows_         PE header
00401000 00097000 Windows_ CODE    code
00498000 00003000 Windows_ DATA    data
0049B000 00002000 Windows_ BSS
0049D000 00003000 Windows_ .idata  imports
004A0000 00001000 Windows_ .tls
004A1000 00001000 Windows_ .rdata
004A2000 0000C000 Windows_ .reloc  relocatio
004AE000 000CB000 Windows_ .rsrc   resources
00580000 00103000
00690000 00001000
```

Figure 5.11: A snapshot from OllyDbg.

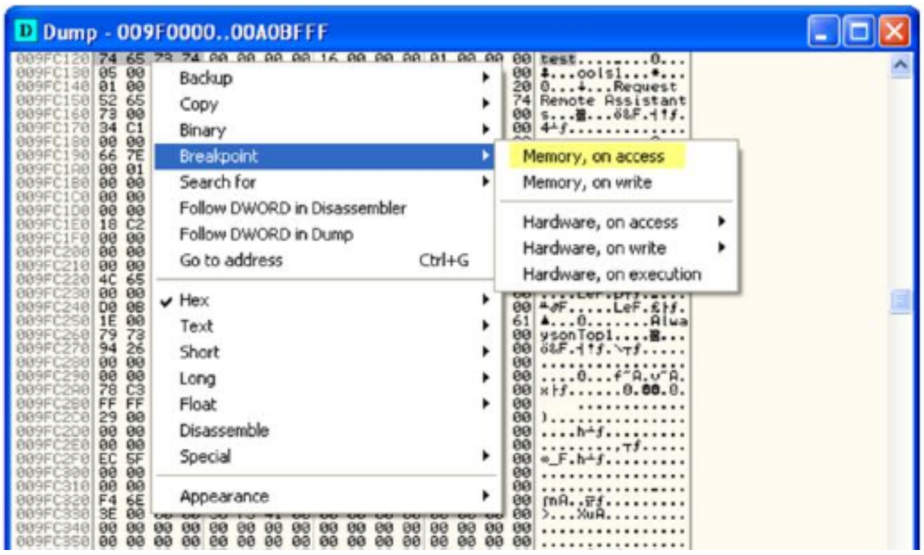


Figure 5.12: A snapshot from OllyDbg.

It is possible that your string will be located in several memory areas. The one you are interested in won't necessarily be the fist one. To repeat your search, click on the memory map window, then press "Ctrl+L" (do not forget to click on the memory map window!).

In the case of our example, we will need to perform the initial search via "Ctrl+B". This will find us an instance of 'test' that is not promising. We will repeat the search by pressing "Ctrl+L" once.

Now that we have located the exact string "test" in the trojan's memory, we can set a breakpoint there. In this case we will be setting a memory access breakpoint, so that OllyDbg pauses the program's execution whenever it attempts to access this particular memory area. Effectively, this will allow us to catch the trojan while it is attempting to use the "test" string; we will then be able to see how it makes use of the string.

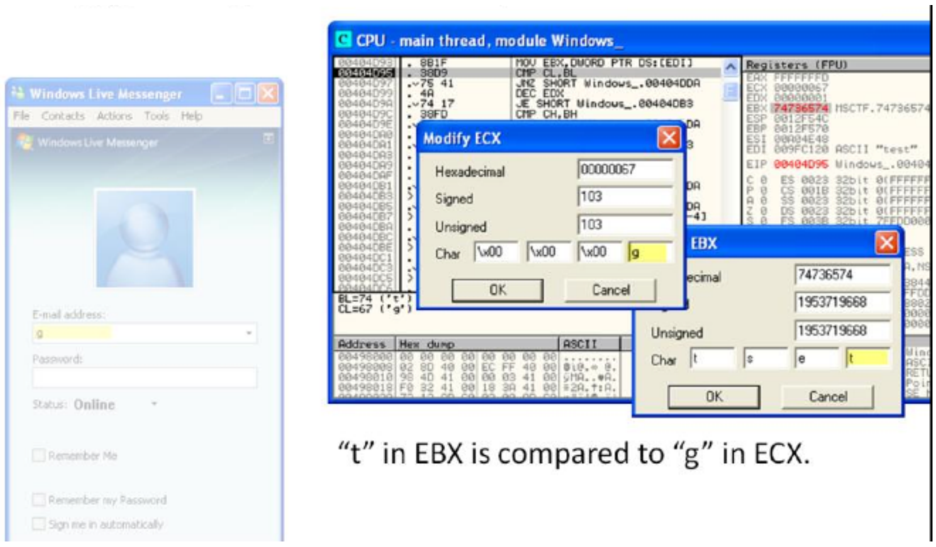
To set the breakpoint, highlight the exact characters of the string "test", then right-click and click "Breakpoint" -> "Memory, on access"[7].

The trojan will continue to run. Now we can either wait for it to try using the string, or attempt interacting with the program with the hope it will use the string.

We can try interacting with the trojan by typing some text into its first field, the one labeled "E- mail address". If you type any character there after setting our memory breakpoint, you will immediately trigger the breakpoint, as you can see on the next figure.

We entered a character into the field (picked a letter at random: "g"). Right

[7]Note that this kind of breakpoint will not appear in the list of breakpoints of OllyDbg; don't worry, it is OK



```
00404D93  .  8B1F       MOV EBX,DWORD PTR DS:[EDI]
00404D95  .  38D9       CMP CL,BL
00404D97  .v75 41       JNZ SHORT Windows_.00404DDA
00404D99  .  4A         DEC EDX
00404D9A  .v74 17       JE SHORT Windows_.00404DB3
00404D9C  .  38FD       CMP CH,BH
```

```
Registers (FPU)
EAX FFFFFFFD
ECX 00000067
EDX 00000001
EBX 74736574 MSCTF.74736574
ESP 0012F54C
EBP 0012F570
ESI 00A04E48
EDI 009FC120 ASCII "test"
EIP 00404D95 Windows_.00404
```

Modify ECX: Hexadecimal 00000067; Signed 103; Unsigned 103; Char \x00 \x00 \x00 g

EBX: Hexadecimal 74736574; Signed 1953719668; Unsigned 1953719668; Char t s e t

"t" in EBX is compared to "g" in ECX.

Figure 5.13: A snapshot from OllyDbg.

away, OllyDbg comes to the foreground, because we just triggered an attempt by the trojan to somehow use the string "test". You can now interact with the code, looking at its environment, and even running it as slowly as one instruction at a time.

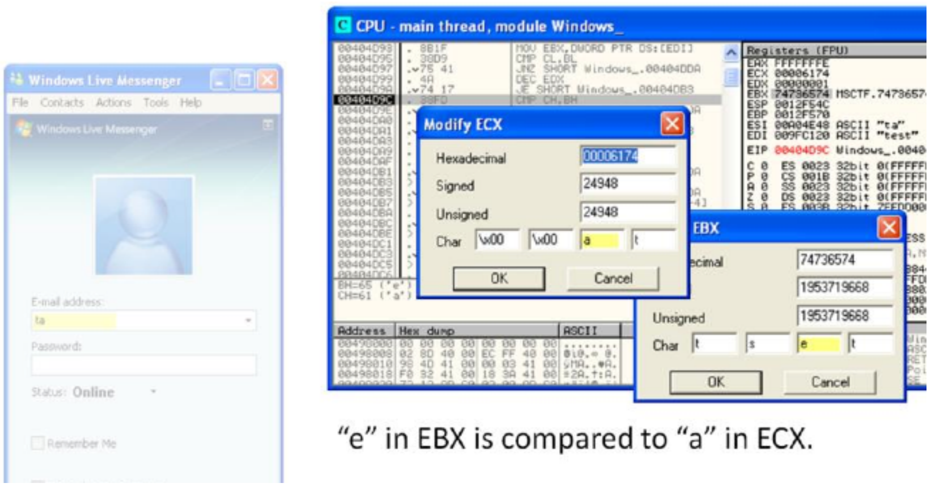
To execute one instruction, press F8. To examine the run-time environment of the program, look at its registers in the top right corner of the OllyDbg window. A register is a specialised and very fast location on the CPU that can store data.

What is going on in this part of the code? Do not worry if you do not understand much of the assembly code you see there: this is just an introduction to malware analysis, so we will walk you through the most important parts. OllyDbg has highlighted the instruction that will be executed next by the program, "CMP CL, BL". This compares contents of two registers, CL and BL. CL points to the lowest byte of ECX; BL points to the lowest byte of EBX, so it is an efficient way of comparing parts of ECX and EBX registers.

Double-click the registers to see their contents. ECX contains the character we entered, "g". EBX contains the string that our input is being compared to, "test" (it is stored backwards).

Press F9 to continue executing the trojan. Delete the "g" character you have entered previously. This time, let the program match the first character of the "test" string, and see how it compares the second character. To do this, enter "ta" in the "E- mail address" box. If you keep triggering the breakpoint, press F9 to continue. You want to pause right after you have had a chance to type "ta".

Press F8 to execute one instruction after you have triggered the breakpoint, just like you did previously. Keep on pressing F8 carefully, once at a time, until you reach "CMP CH,BH" some lines below. This time, if you look at contents of ECX


Windows Live Messenger
File Contacts Actions Tools Help
CPU - main thread, module Windows_
Modify ECX
EBX
OK
Cancel

E-mail address: ta
Status: Online
Remember Me

```
00404D93  . 8B1F      MOV EBX,DWORD PTR DS:[EDI]
00404D95  . 38D9      CMP CL,BL
00404D97  .v75 41     JNZ SHORT Windows_.00404DDA
00404D99  . 4A        DEC EDX
00404D9A  .v74 17     JE SHORT Windows_.00404DB3
00404D9C  . 38FD      CMP CH,BH
BH=65 ('e')
CH=61 ('a')
```

```
Registers (FPU)
EAX FFFFFFFE
ECX 00006174
EDX 00000001
EBX 74736574 MSCTF.7473657
ESP 0012F54C
EBP 0012F570
ESI 00A04E48 ASCII "ta"
EDI 009FC120 ASCII "test"
EIP 00404D9C Windows_.0040
```

Modify ECX — Hexadecimal: 00006174; Signed: 24948; Unsigned: 24948; Char: \x00 \x00 a t

EBX — Hexadecimal: 74736574; Signed: 1953719668; Unsigned: 1953719668; Char: t s e t

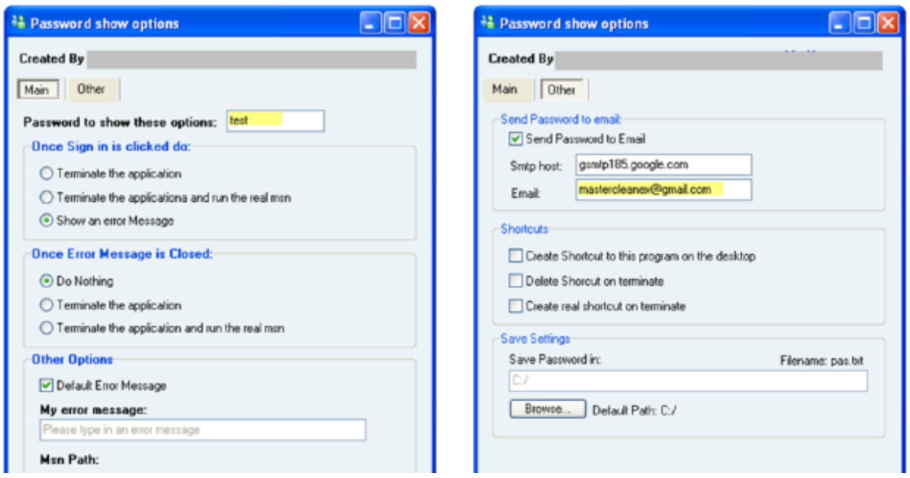
"e" in EBX is compared to "a" in ECX.

Figure 5.14: A snapshot from OllyDbg.

and EBX registers, you will notice that the trojan is comparing the character 'a' that we entered to the character 'e' that it seems to expect. That is because the CH register points to the second lowest byte of ECX; the BH register points to the second lowest byte of EBX.

So, the trojan seems to be looking for the string "test" in the "E-mail address" field. Exit the debugger, launch the trojan by itself, and enter "test" to see what happens.

Voila! When you enter "test", the trojan brings you to a brand new screen that seems to allow you to configure the trojan's operation. As you can see on this slide, the configuration options let you define the passphrase to activate this string, the address where the trojan will send captured login credentials, etc.



Password show options

Created By [redacted]

Main | Other

Password to show these options: test

Once Sign in is clicked do:

- ○ Terminate the application
- ○ Terminate the applicationa and run the real msn
- ● Show an error Message

Once Error Message is Closed:

- ● Do Nothing
- ○ Terminate the application
- ○ Terminate the application and run the real msn

Other Options

- ☑ Default Error Message

My error message: Please type in an error message

Msn Path:

Password show options

Created By [redacted]

Main | Other

Send Password to email:

- ☑ Send Password to Email

Smtp host: gsmtp185.google.com

Email: mastercleanex@gmail.com

Shortcuts

- ☐ Create Shortcut to this program on the desktop
- ☐ Delete Shorcut on terminate
- ☐ Create real shortcut on terminate

Save Settings

Save Password in: C:/ Filename: pas.txt

Default Path: C:/

Figure 5.15: The configuration panels of the malware.