**TASK 2**

**Malware Analysis**

d9c37b937ffde812ae15de885913e101

| File Name | f188abc33d351c2254d794b525c5a8b79ea78acd3050cd8d27d3ecfc568c2936 |
| --- | --- |
| File type | Win32 Executable |
| File Size | 328 KB |
| MD5 | d9c37b937ffde812ae15de885913e101 |
| SHA1 | ed1cd9e086923797fe2e5fe8ff19685bd2a40072 |
| SHA265 | f188abc33d351c2254d794b525c5a8b79ea78acd3050cd8d27d3ecfc568c2936 |
| Sample Origin | Downloaded from LumniNUS <https://luminus.nus.edu.sg/modules/c0629e16-d7d2-40c0-8fc2-01e6d2e7184f/files/e3039acb-8e33-463a-a7ab-6903ab3ab9e4> |
| Date of Analysis | 22/11/2021 13:37 |
| Type of Analysis | Static and Dynamic |
| Packed | .rsrc section likely to be packed or obfuscated |
| Compilation Date | 2017/12/27 11:44:35 UTC |
| #DLLs | 3 |
| #FNs | 82 |
| Entropy .text | 6.508244893979196 |
| Entropy .rdata | 4.90962531442763 |
| Entropy .data | 2.2202882533697186 |
| Entropy .pdata | 4.758069810234056 |
| Entropy .gfids | 1.6206175832411014 |
| Entropy .rsrc | 7.996551078065988 |
| Entropy .reloc | 4.849914007496848 |

**Basic Static Analysis**

Entropy Analysis

.rsrc section has very high entropy values and is likely to be packed or obfuscated

Graphical user interface, text, website

Description automatically generated

VirusTotal Summary

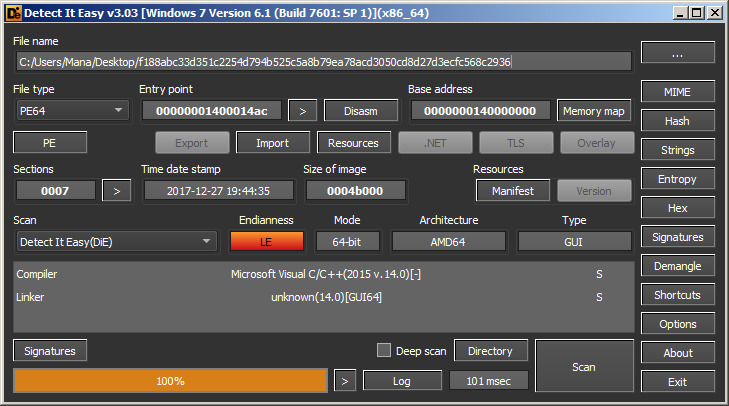
Matches 55/69 of existing antivirus definitions.

Graphical user interface, application, Teams

Description automatically generated

Packing Summary

Main program does not appear to be packed



PeView Summary

Compilation Date: 2017/12/27 11:44:35 UTC

Graphical user interface, text, application

Description automatically generated

DependencyWalker

3 DLLS: KERNEL32.DLL, ADVAPI32.DLL, SHELL32.DLL

Basic Functionality: Manipulates files, connects to some network, and uses shell to execute commands.

Graphical user interface, text, application, email

Description automatically generated

Resource Check

File contains resources :

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Strings Analysis

The malware uses Windows' own software. It checks whether the DLLs are missing or ineffective and see if malware has access to specific applications to use the OS.

Text

Description automatically generated

Malware has privilege escalation.

Graphical user interface, text, application

Description automatically generated

Error detection used by malware

Text

Description automatically generated

Interesting string to take note of:

Text, letter

Description automatically generated

Interesting DLL to take note of that is not present in the dependency walker: mscoree.dll.

Text

Description automatically generated

Import Analysis

The malware dynamically imports external resources (GetProcAddress, LoadLibraryA)

Graphical user interface, text, application

Description automatically generated

Malware has the ability to operate files, both read and write



Malware has a specific way to be executed, either as a file or through command line argument.

Graphical user interface, text

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence

The Malware may allocate memory and use it in some fashion. (VirtualAlloc, HeapAlloc, VirtualFree, HeapFree)

Graphical user interface, application

Description automatically generated with medium confidence

Malware attempts for privilege escalation

Graphical user interface, website

Description automatically generated

Malware looks for a file (FindFirstFileExW, FindNextFIleW), loads some argument into the buffer (FreeEnvironmentStringW, SetStdHandle), then executes the files (WriteConsoleW, CommandLineToArgvW)Table

Description automatically generated with medium confidence

**Advanced Static Analysis via IDA Pro**

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Defence Evasion | Obfuscated Files or Information | Encoding::XOR | 0x140011320  0x1400113D0 |

Purpose: to clear the register, and prep for inserting information.

Graphical user interface, text

Description automatically generated

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Defence Evasion | Obfuscated Files or Information | Obfuscated Files or Information::Encoding-Standard Algorithm   1. AES | 0x140010AD0  0x140010C80  0x140010EE0 |

Purpose: the payload is compressed and encrypted in order to avoid detection.

Graphical user interface, text

Description automatically generated with medium confidence

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Defence Evasion | Obfuscated Files or Information | Obfuscated Files or Information::Encoding-Standard Algorithm   1. Hash MD5 | 0x140012E60 |

Purpose: the payload is compressed and encrypted in order to avoid detection.

Graphical user interface

Description automatically generated with medium confidence

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Defence Evasion | Obfuscated Files or Information | Uses a resource file | resource (.rsrc) section |
|  |  | extract resource via kernel32 functions | 0x140013220 |
|  |  | accept command line arguments | 0x14000A4A8  0x1400135D0  0x140013630 |
|  |  | Queries environment variables | 0x14000A4D4 |

Purpose: The resource section is the resources required by the malware to execute the program. It is separated in order to avoid immediate detection.

Resource section

Text

Description automatically generated

extract resource via kernel32 functions

A picture containing table

Description automatically generated

accept command line arguments Text

Description automatically generated with medium confidence

Graphical user interface

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated

Queries environment variables

Text

Description automatically generated

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Discovery | File and Directory Discovery | enumerate files via kernel32 functions | 0x14000979C |

Purpose: to enumerate files and directories for certain information within a file system

0x14000979C

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Discovery | File and Directory Discovery | read file | 0x140007AAC  0x14000802C  0x14000825C  0x140008674 |

Purpose: to enumerate files and directories for certain information within a file system

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Execution | Command and Scripting Interpreter | Write file | 0x14000C000  0x14000C208  0x14000C310  0x14000C42C  0x14000C68C  0x140013630 |

Purpose: Environment variables are used to evade signature based detections and application control mechanisms.

0x14000C000

Graphical user interface, text

Description automatically generated

Kernel32 function used: WriteFile

Text

Description automatically generated

0x14000C208

Text

Description automatically generated with medium confidence

Kernel32 function used: WriteFile

A picture containing text

Description automatically generated

0x14000C310

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

0x14000C68C

Text

Description automatically generated with medium confidence

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Execution | Command and Scripting Interpreter | allocate thread local storage | 0x140004C2C  0x14000A70C |

Purpose: To allocate memory for the malware to run in.

Graphical user interface, text, application

Description automatically generated

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Privilege Escalation | Access Token Manipulation | modify access privileges | 0x140013040 |

Purpose: Used to modify access tokens to operate and perform actions that bypasses access controls.

A picture containing graphical user interface

Description automatically generated

Key Points in Graph View:

Kernel32 functions used: GetCurrentProcess, OpenProcessToken, GetTokenInformation, AdjustTokenPrivileges

Text

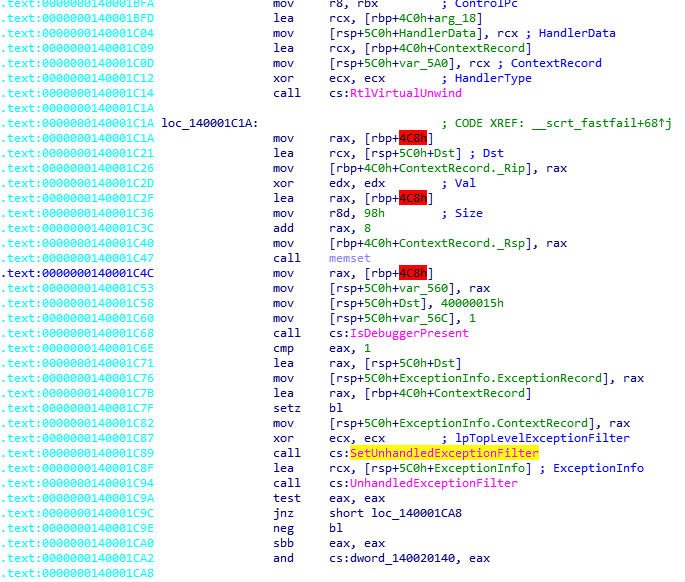
Description automatically generated

Graphical user interface, text, application

Description automatically generated

| Malware Tactic | Attack Technique | Behaviour | Function Location |
| --- | --- | --- | --- |
| Defence Evasion | Anti-debugging | Detect debugger and register top level exception handle | text (.text) section |

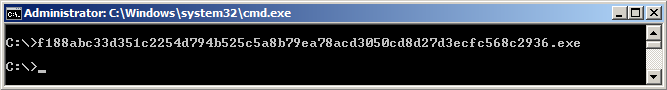
The malware seems to be able to detect the existence of a debugger with ‘IsDebuggerPresent’ and register top level exception handle with ‘SetUnhandledExceptionFilter’ and ‘UnhandledExceptionFilter’. This is a commonly used anti-debugger technique.



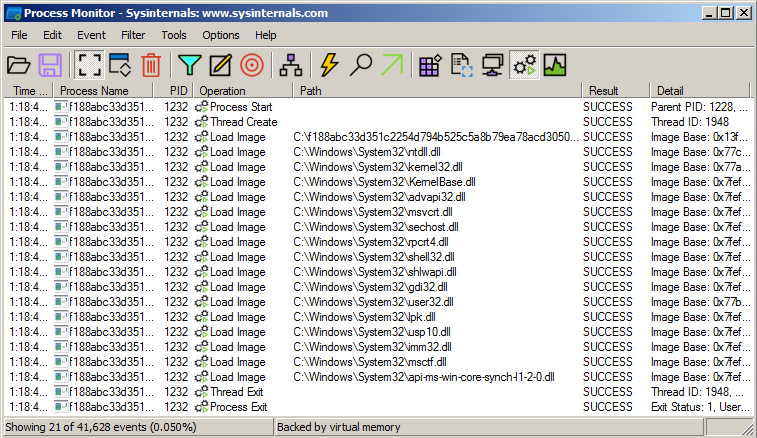
**Advanced Dynamic Analysis**

**Core Functionality**

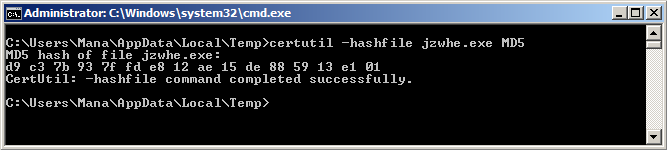
By executing the sample normally, nothing seems to happen and just terminates immediately.



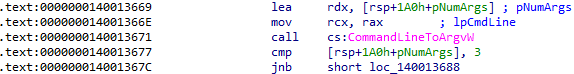
Inspecting the events in Process Monitor, the sample seems to load several DLLs upon launch, but does not do anything significant and terminates shortly after.



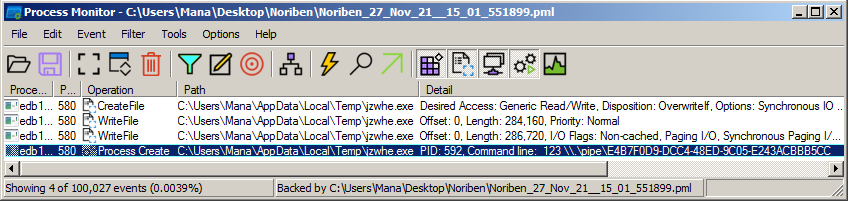
However, upon further observation, the MD5 hash value of this sample matches one of the 5 executables that is created by sample **edb1ff2521fb4bf748111f92786d260d40407a2e8463dcd24bb09f908ee13eb9** upon launched as referenced in Task 1-3.



It is also hinted at **0x140013677** that the sample takes in 3 arguments in the static analysis.

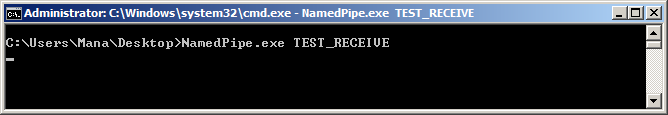


Observing the process calls from sample **edb1ff2521fb4bf748111f92786d260d40407a2e8463dcd24bb09f908ee13eb9** in Process Monitor, this particular sample takes in 2 arguments, a consistent value **123** and a named pipe that randomizes its name upon every launch.



In order to analyse the contents of the named pipe, we will need a server that is capable of reading and printing values from the named pipe, such as using the NamedPipe server from <https://github.com/nu774/namedpipe>.

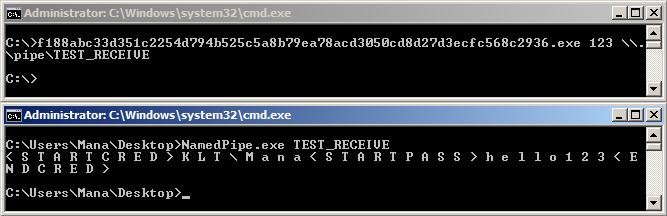
To utilise this server, we will need to specify a pipe name as an argument to the NamedPipe server. In this example, we will be using the pipe name **TEST\_RECEIVE**.



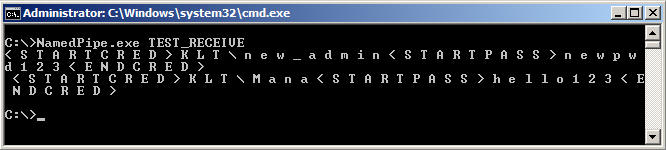
Next, we will need to specify our server’s pipe name to the sample using the command:

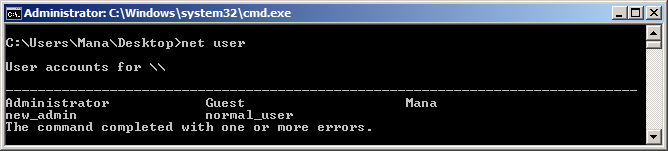
f188abc33d351c2254d794b525c5a8b79ea78acd3050cd8d27d3ecfc568c2936.exe **123** **\\.\pipe\TEST\_RECEIVE**

After running the above command, we can see that the details of our login credentials are being relayed in the form of <STARTCRED>**COMPUTERNAME\USERNAME**<STARTPASS>**PASSWORD**<ENDCRED>. In this instance, the username used is **KLT\Mana** and the password used is **hello123**.

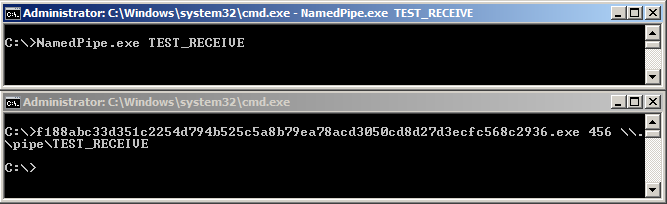


By logging in to another user in the same session (username: **new\_admin**, password: **newpwd123**), we can see that the sample manages to retrieve the credentials of only the currently logged in users, and not all the users that exist within the system.

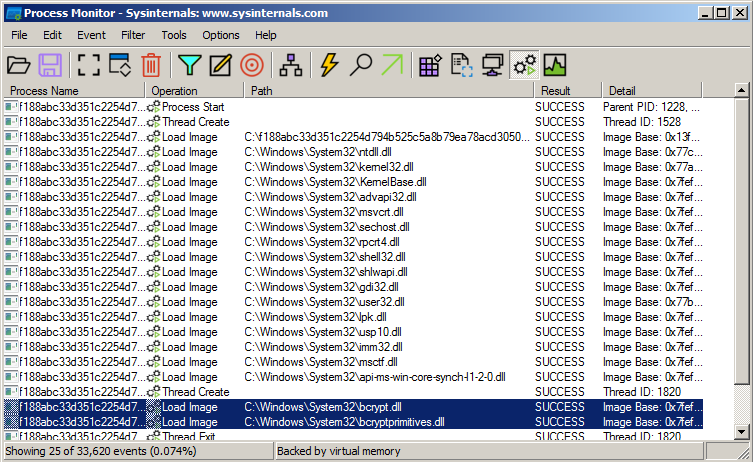




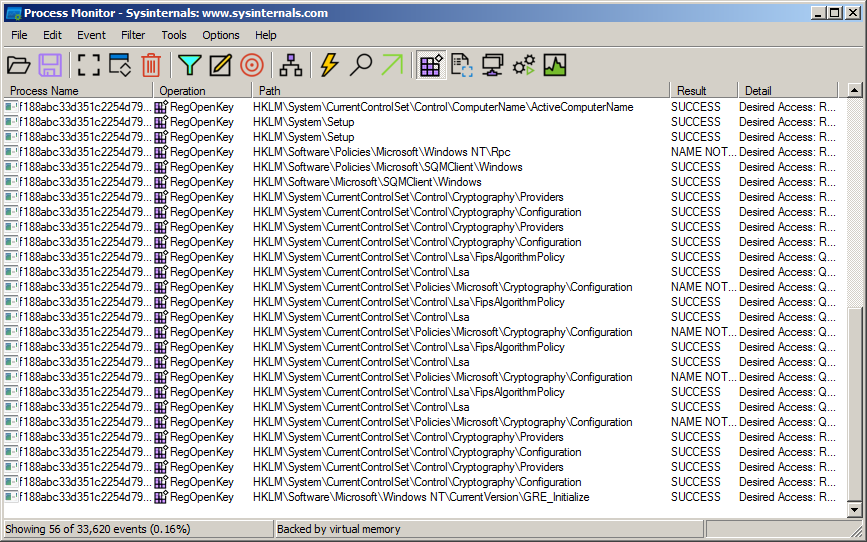
Additionally, it is also noted that by changing the 2nd argument **123** to any other value such as **456**, the server will not receive any data from the sample.



Inspecting the events in Process Monitor again, we can see additional DLLs being loaded now, which are **bcrypt.dll** and **bcryptprimitives.dll**.



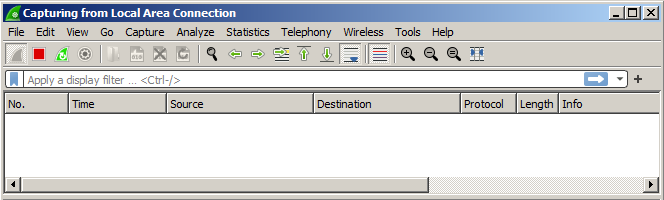
Inspecting the Registry activities, we can also see that many security and cryptography related keys are being read and queried by the sample.

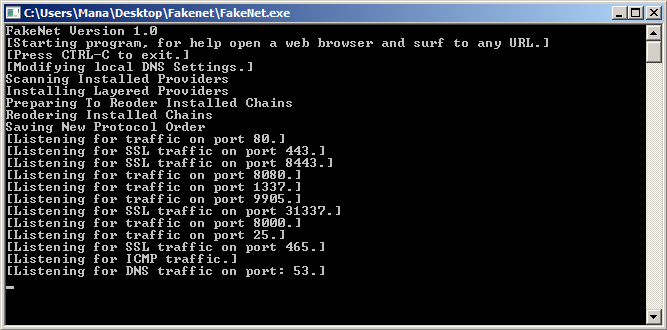


Thus, we can confirm that this sample attempts to retrieve the Windows login credentials of all logged in users in plaintext and relay it back to the main application that invokes this command.

**Network Analysis**

The sample does not attempt to establish any network connections, even when executed with the supplied arguments, as shown from the network capture with Wireshark and the simulated fake network with FakeNet.

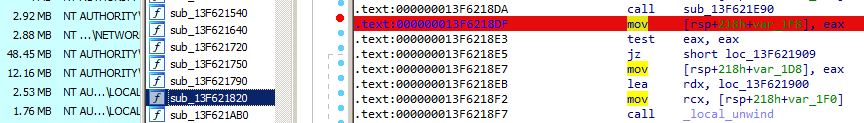
****

****

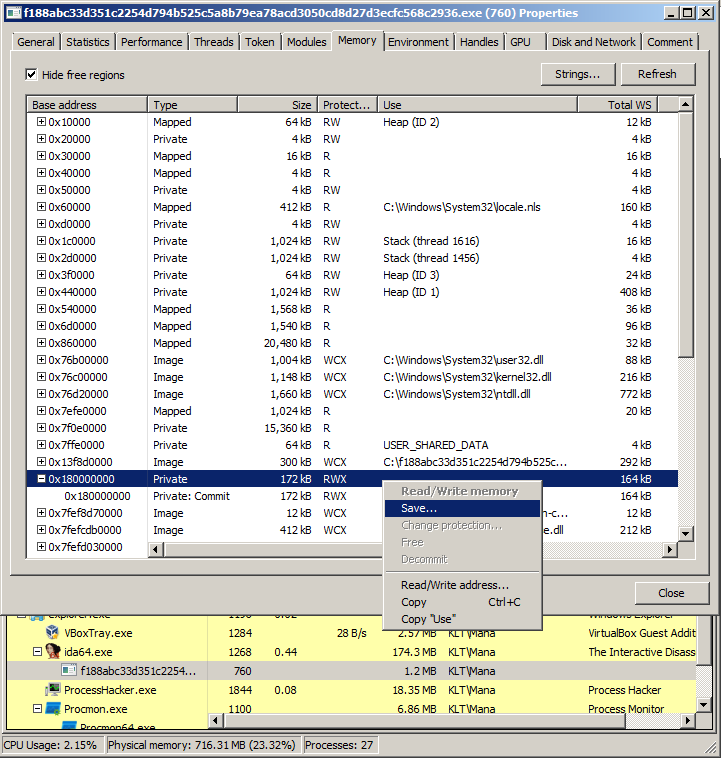
**Reading from the memory**

In this section, we will attempt to retrieve some information about the loaded resource from the memory while the sample is running. Here, we will make use of IDA and WinDbg to set-up breakpoints to pause the execution of the sample, and Process Hacker to extract the resource from the memory.

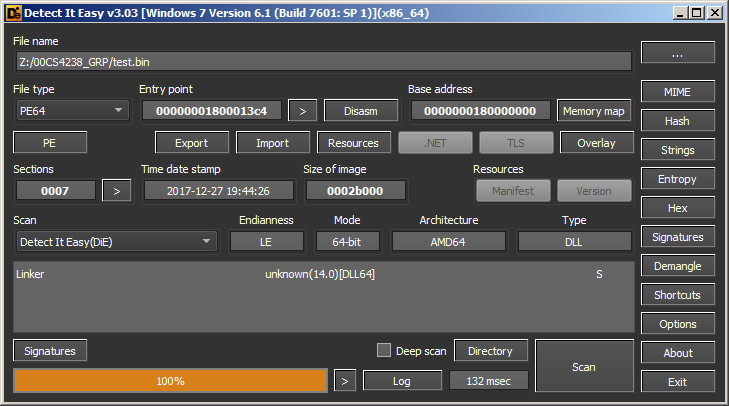
In IDA, we will set a breakpoint at **0x13F6218DF**, which is right after calling subroutine **0x13F621E90** that is responsible for loading the resource into the memory.



Next, we will switch over to Process Hacker and save the data from the memory that is loaded right after the sample executable. In this example, the data is located at base address **0x180000000**.



**Static analysis on extracted data from memory**

By loading the extracted file in Detect It Easy, it was identified as a DLL file with a compiled date of 2017/12/27 19:44:26.  
  


Loading the file in IDA however resulted in several errors as the extracted DLL does not seem to be complete and contains other “noise” from the memory. Instead, we will focus on some interesting string analysis extracted from Ghidra.

Within the strings, we can see several references to security related functions, DLLs and executables such as **lsasrv.dll**, **lsass.exe**, **kdcsvc.dll** and **kerberos.dll**. These functions and DLLs are likely to be accessed by the sample in order to obtain the login credentials of the users in plaintext. The format for the output of the sample are also found within these strings, such as **<ENDCRED>**, **<STARTPASS>**

and **<STARTCRED>**.

Additionally, we also found an additional DLL name **kiwi64.dll** that was not seen in the previous static analysis, and is very likely to be the actual name of the resource that is being loaded in the memory by the sample.

| kiwi64.dll  %ls\n  lsasrv  lsasrv.dll  LsaICancelNotification  LsaIRegisterNotification  bcrypt  BCryptOpenAlgorithmProvider  BCryptSetProperty  BCryptGetProperty  BCryptGenerateSymmetricKey  tBCryptEncrypt  BCryptDecrypt  BCryptDestroyKey  tBCryptCloseAlgorithmProvider  3DES  ChainingModeCBC  ChainingMode  ObjectLength  ChainingModeCFB  ChainingMode  ObjectLength  kdcsvc.dll  lsass.exe  UndefinedLogonType  Unknown !  Interactive  Network  Batch  Service  Proxy  Unlock  NetworkCleartext  NewCredentials  RemoteInteractive  CachedInteractive  CachedRemoteInteractive  CachedUnlock  %ls%wZ\\%wZ%ls%.\*s%ls%ls  <ENDCRED>  <STARTPASS>  <STARTCRED>  %ls%wZ\\%wZ%ls%wZ%ls%ls  credman  lsasrv.dll  \n\t [%08x]  kerberos  kerberos.dll  Ticket Granting Service  Client Ticket ?  Ticket Granting Ticket  livessp  livessp.dll  Primary  CredentialKeys  lsasrv.dll  msv1\_0.dll  \n\t [%08x]  tspkg  tspkg.dll  wdigest  wdigest.dll  %02x  0x%02x,  \\x%02x  \nBYTE data[] = {\n\t  \n};\n |
| --- |