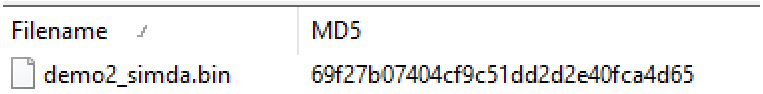
Here is my write-up for the file “Simba.bin”. The objective here is to find indicators of compromise/evidence that we are dealing with a malicious file.

Tools used during this analysis: Hashmyfiles, TrIDNET, DetectitEasy, PEStudio, x32dbg, ProcessHacker, bstrings, Regshot, VisualStudioCode, Procmon

---

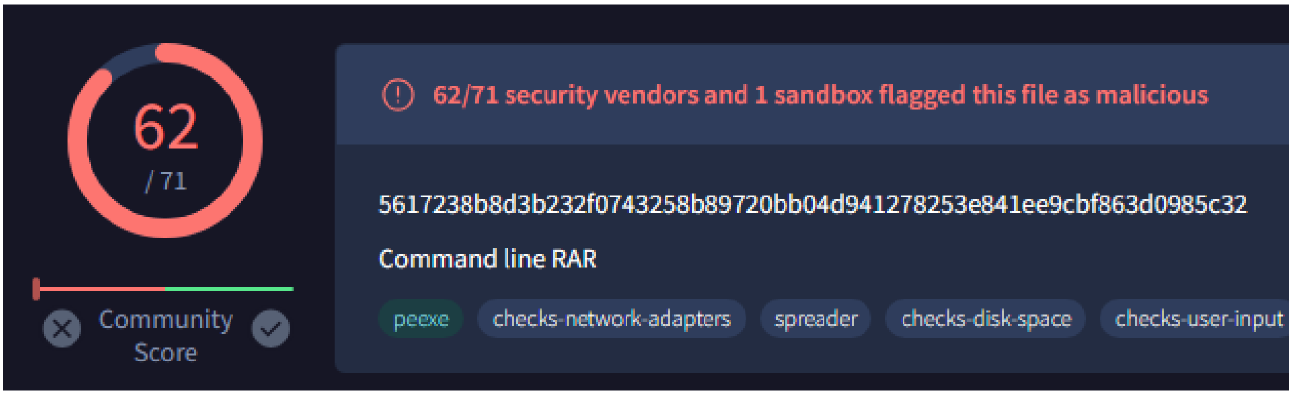
Static Analysis

We start off by performing hash analysis. We use hashmyfiles to create a hash of the simba.bin file.



Filename	MD5
demo2_simda.bin	69f27b07404cf9c51dd2d2e40fca4d65

Then we test this hash on virustotal.



62/71 security vendors and 1 sandbox flagged this file as malicious

5617238b8d3b232f0743258b89720bb04d941278253e841ee9cbf863d0985c32

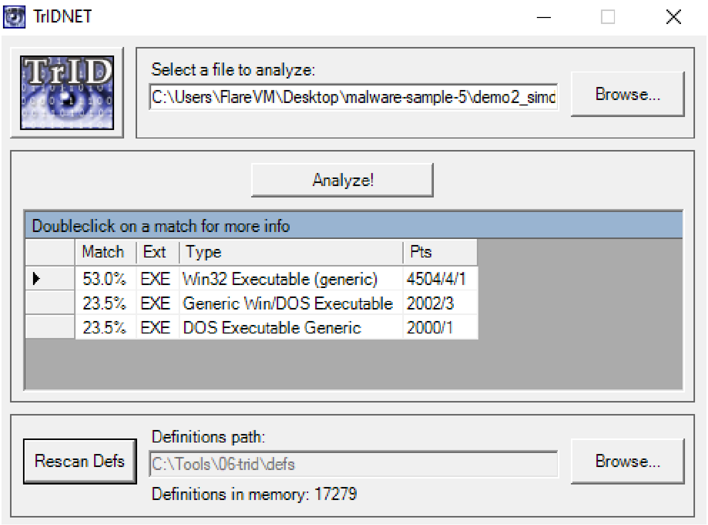
Command line RAR

peexe checks-network-adapters spreader checks-disk-space checks-user-input

Virustotal gives us a 62/71 for Trojan, a clear indicator that we are indeed dealing with malware.

Next we want to know more about the nature of this file, what kind of file we are dealing with and if the content is packed or not.

We use TrIDNET for file type identification. We see that this is an .exe file.



Select a file to analyze: C:\Users\FlareVM\Desktop\malware-sample-5\demo2_simc

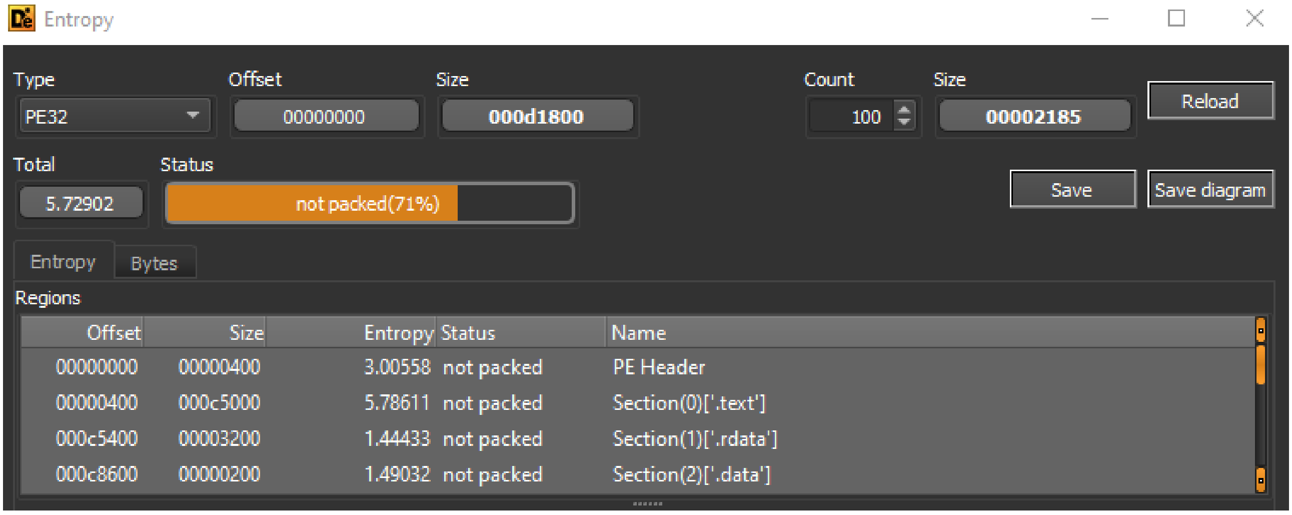
Doubleclick on a match for more info

Match	Ext	Type	Pts
53.0%	EXE	Win32 Executable (generic)	4504/4/1
23.5%	EXE	Generic Win/DOS Executable	2002/3
23.5%	EXE	DOS Executable Generic	2000/1

Definitions path: C:\Tools\06-trid\defs

Definitions in memory: 17279

We use Detect it Easy to see if the file content is packed or not.


Entropy

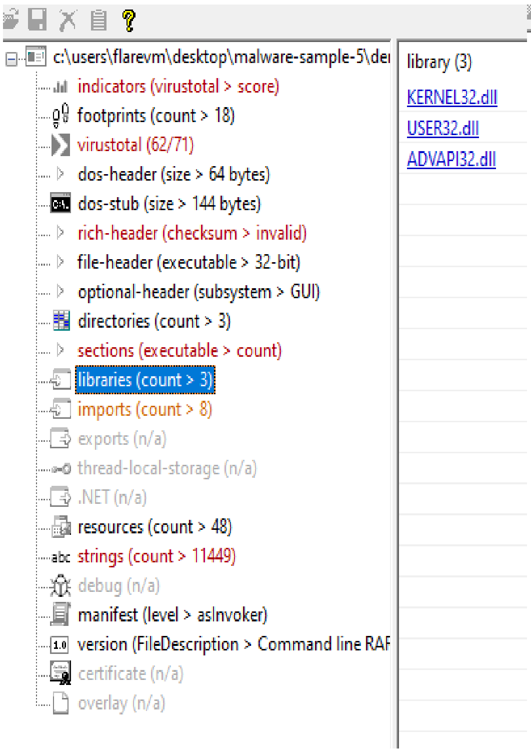
Type: PE32
Offset: 00000000
Size: 000d1800
Count: 100
Size: 00002185
Total: 5.72902
Status: not packed(71%)

Regions

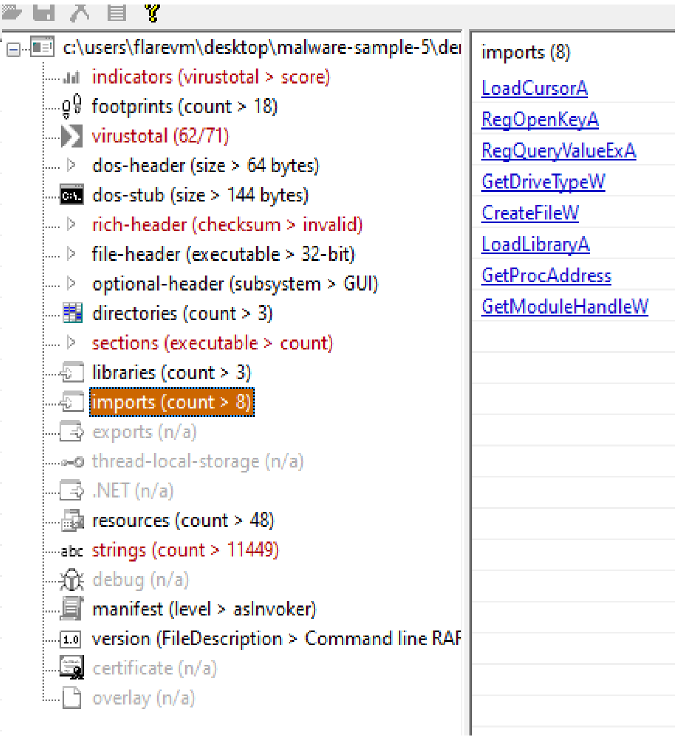
Offset	Size	Entropy	Status	Name
00000000	00000400	3.00558	not packed	PE Header
00000400	000c5000	5.78611	not packed	Section(0)['.text']
000c5400	00003200	1.44433	not packed	Section(1)['.rdata']
000c8600	00000200	1.49032	not packed	Section(2)['.data']

Detect it Easy shows that the content of this file is not packed so we can continue with our other tooling for string analysis.

We use PEStudio to investigate more on this file but we notice some strange things about it, there is not a lot of data available. There are only 3 libraries, 8 imports and 1 flagged string. A normal program will have around 100 imports making me believe that the real payload is still hidden.

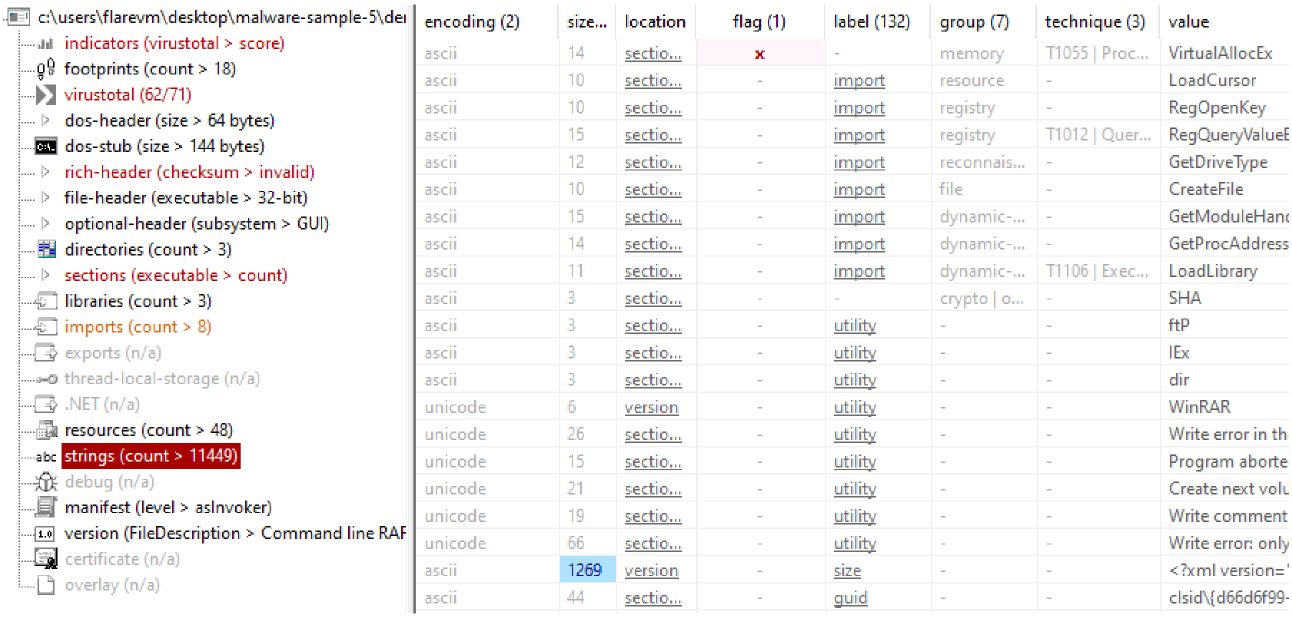

c:\users\flarevm\desktop\malware-sample-5\de
indicators (virustotal > score)
footprints (count > 18)
virustotal (62/71)
dos-header (size > 64 bytes)
dos-stub (size > 144 bytes)
rich-header (checksum > invalid)
file-header (executable > 32-bit)
optional-header (subsystem > GUI)
directories (count > 3)
sections (executable > count)
libraries (count > 3)
imports (count > 8)
exports (n/a)
thread-local-storage (n/a)
.NET (n/a)
resources (count > 48)
strings (count > 11449)
debug (n/a)
manifest (level > asInvoker)
version (FileDescription > Command line RAF
certificate (n/a)
overlay (n/a)

library (3)
KERNEL32.dll
USER32.dll
ADVAPI32.dll


c:\users\flarevm\desktop\malware-sample-5\de
indicators (virustotal > score)
footprints (count > 18)
virustotal (62/71)
dos-header (size > 64 bytes)
dos-stub (size > 144 bytes)
rich-header (checksum > invalid)
file-header (executable > 32-bit)
optional-header (subsystem > GUI)
directories (count > 3)
sections (executable > count)
libraries (count > 3)
imports (count > 8)
exports (n/a)
thread-local-storage (n/a)
.NET (n/a)
resources (count > 48)
strings (count > 11449)
debug (n/a)
manifest (level > asInvoker)
version (FileDescription > Command line RAF
certificate (n/a)
overlay (n/a)

imports (8)
LoadCursorA
RegOpenKeyA
RegQueryValueExA
GetDriveTypeW
CreateFileW
LoadLibraryA
GetProcAddress
GetModuleHandleW

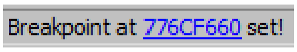
The 1 string that is shown as flagged is VirtualAllocEx, we need to run this program in a debugger and extract the real payload out of the memory that is allocated to it.



```
c:\users\flarevm\desktop\malware-sample-5\de
  indicators (virustotal > score)
  footprints (count > 18)
  virustotal (62/71)
  dos-header (size > 64 bytes)
  dos-stub (size > 144 bytes)
  rich-header (checksum > invalid)
  file-header (executable > 32-bit)
  optional-header (subsystem > GUI)
  directories (count > 3)
  sections (executable > count)
  libraries (count > 3)
  imports (count > 8)
  exports (n/a)
  thread-local-storage (n/a)
  .NET (n/a)
  resources (count > 48)
  strings (count > 11449)
  debug (n/a)
  manifest (level > asInvoker)
  version (FileDescription > Command line RAF
  certificate (n/a)
  overlay (n/a)
```

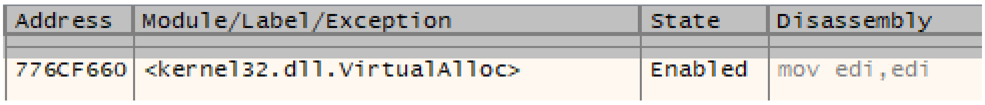
encoding (2)	size...	location	flag (1)	label (132)	group (7)	technique (3)	value
ascii	14	sectio...	x	-	memory	T1055 | Proc...	VirtualAllocEx
ascii	10	sectio...	-	import	resource	-	LoadCursor
ascii	10	sectio...	-	import	registry	-	RegOpenKey
ascii	15	sectio...	-	import	registry	T1012 | Quer...	RegQueryValueE
ascii	12	sectio...	-	import	reconnais...	-	GetDriveType
ascii	10	sectio...	-	import	file	-	CreateFile
ascii	15	sectio...	-	import	dynamic-...	-	GetModuleHanc
ascii	14	sectio...	-	import	dynamic-...	-	GetProcAddress
ascii	11	sectio...	-	import	dynamic-...	T1106 | Exec...	LoadLibrary
ascii	3	sectio...	-	-	crypto | o...	-	SHA
ascii	3	sectio...	-	utility	-	-	ftP
ascii	3	sectio...	-	utility	-	-	IEx
ascii	3	sectio...	-	utility	-	-	dir
unicode	6	version	-	utility	-	-	WinRAR
unicode	26	sectio...	-	utility	-	-	Write error in th
unicode	15	sectio...	-	utility	-	-	Program aborte
unicode	21	sectio...	-	utility	-	-	Create next volu
unicode	19	sectio...	-	utility	-	-	Write comment
unicode	66	sectio...	-	utility	-	-	Write error: only
ascii	1269	version	-	size	-	-	<?xml version='
ascii	44	sectio...	-	guid	-	-	clsid\{d66d6f99-

We use x32dbg as our chosen debugger, we type in "bp virtualalloc" to create a breakpoint at the location where the memory allocation is done.



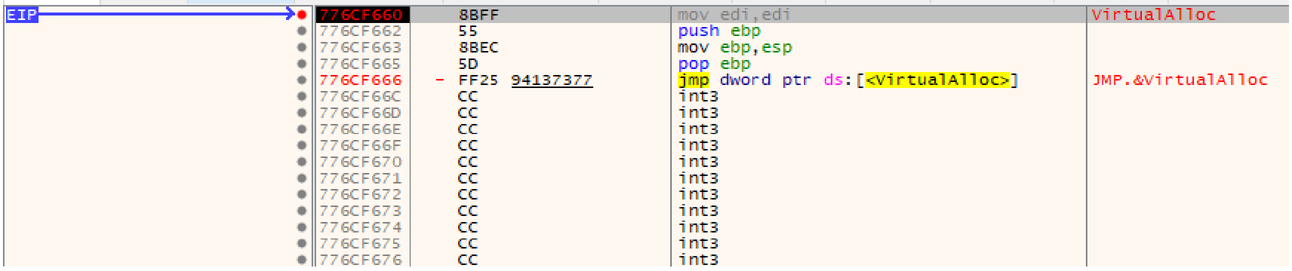
Breakpoint at 776CF660 set!

And check if it's set correctly under the tab "Breakpoints"



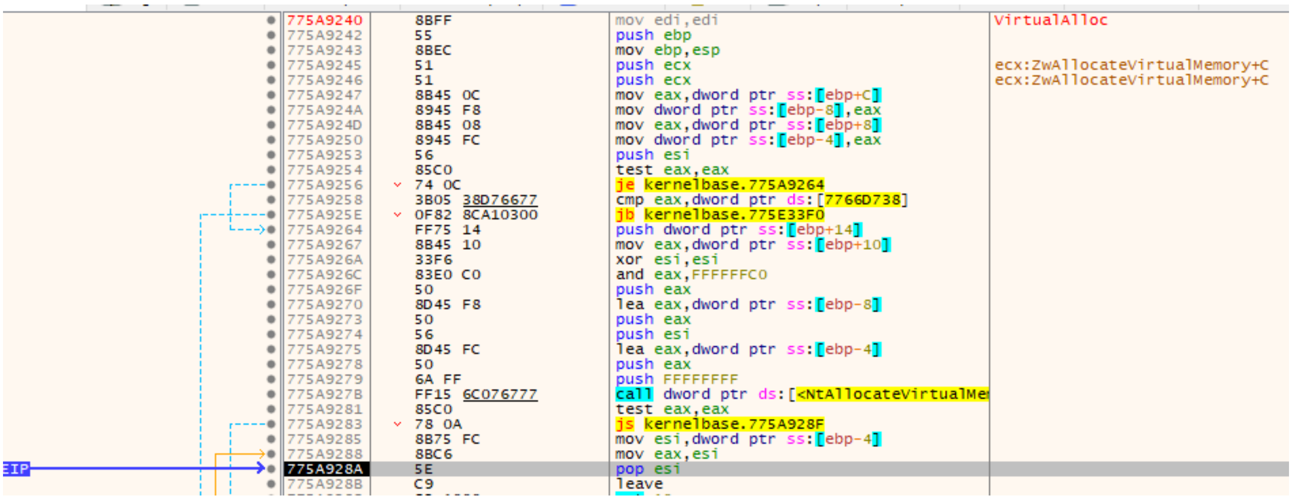
Address	Module/Label/Exception	State	Disassembly
776CF660	<kernel32.dll.VirtualAlloc>	Enabled	mov edi,edi

We click on the "Run" button to run the program in the debugger. The debugger will stop at the breakpoint that we just set.



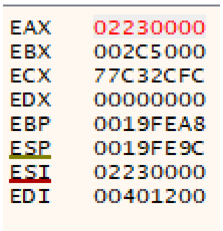
```
EIP  776CF660     8BFF            mov edi,edi                          VirtualAlloc
     776CF662     55              push ebp
     776CF663     8BEC            mov ebp,esp
     776CF665     5D              pop ebp
     776CF666   - FF25 94137377   jmp dword ptr ds:[<VirtualAlloc>]    JMP.&VirtualAlloc
     776CF66C     CC              int3
     776CF66D     CC              int3
     776CF66E     CC              int3
     776CF66F     CC              int3
     776CF670     CC              int3
     776CF671     CC              int3
     776CF672     CC              int3
     776CF673     CC              int3
     776CF674     CC              int3
     776CF675     CC              int3
     776CF676     CC              int3
```

Then we use the "Step Over" button until we reach the pop esi field under the call of virtual memory allocation.



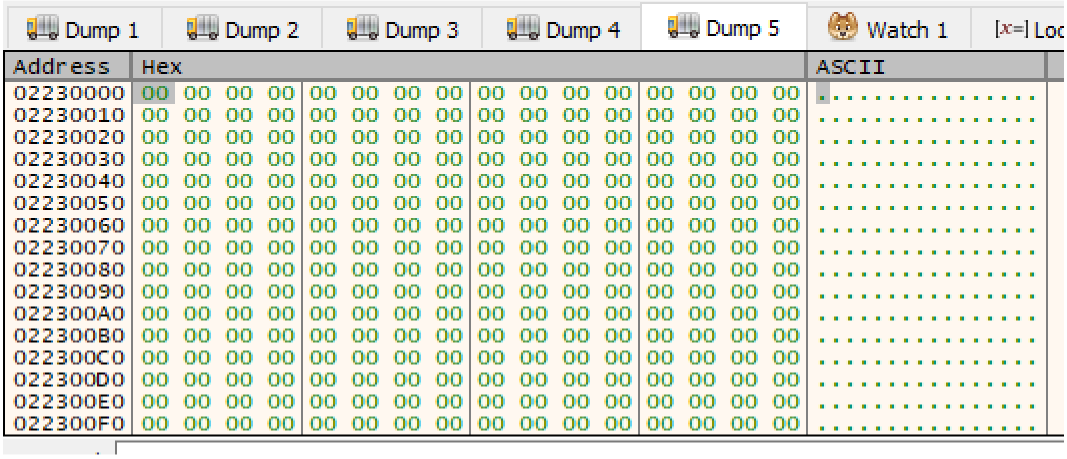
```
775A9240   8BFF            mov edi,edi                              VirtualAlloc
775A9242   55              push ebp
775A9243   8BEC            mov ebp,esp
775A9245   51              push ecx                                 ecx:ZwAllocateVirtualMemory+C
775A9246   51              push ecx                                 ecx:ZwAllocateVirtualMemory+C
775A9247   8B45 0C         mov eax,dword ptr ss:[ebp+C]
775A924A   8945 F8         mov dword ptr ss:[ebp-8],eax
775A924D   8B45 08         mov eax,dword ptr ss:[ebp+8]
775A9250   8945 FC         mov dword ptr ss:[ebp-4],eax
775A9253   56              push esi
775A9254   85C0            test eax,eax
775A9256   74 0C           je kernelbase.775A9264
775A9258   3B05 38D76677   cmp eax,dword ptr ds:[7766D738]
775A925E   0F82 8CA10300   jb kernelbase.775E33F0
775A9264   FF75 14         push dword ptr ss:[ebp+14]
775A9267   8B45 10         mov eax,dword ptr ss:[ebp+10]
775A926A   33F6            xor esi,esi
775A926C   83E0 C0         and eax,FFFFFFC0
775A926F   50              push eax
775A9270   8D45 F8         lea eax,dword ptr ss:[ebp-8]
775A9273   50              push eax
775A9274   56              push esi
775A9275   8D45 FC         lea eax,dword ptr ss:[ebp-4]
775A9278   50              push eax
775A9279   6A FF           push FFFFFFFF
775A927B   FF15 6C076777   call dword ptr ds:[<NtAllocateVirtualMe
775A9281   85C0            test eax,eax
775A9283   78 0A           js kernelbase.775A928F
775A9285   8B75 FC         mov esi,dword ptr ss:[ebp-4]
775A9288   8BC6            mov eax,esi
EIP  775A928A   5E         pop esi
775A928B   C9              leave
```

This will set a location in memory, 02230000.



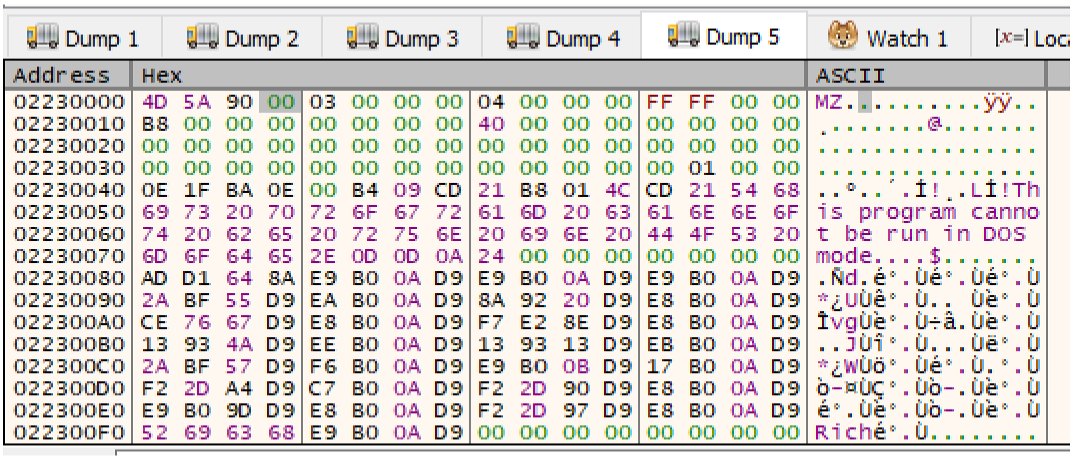
```
EAX   02230000
EBX   002C5000
ECX   77C32CFC
EDX   00000000
EBP   0019FEA8
ESP   0019FE9C
ESI   02230000
EDI   00401200
```

We need to watch this memory location as it will be filled with the malicious payload. We dump the memory to any available dump location and it will be shown as empty until we run the program.



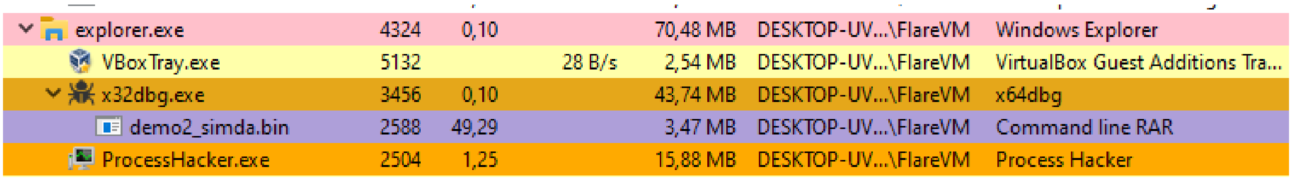
```
Address   Hex                                               ASCII
02230000  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230010  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
022300A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
022300B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
022300C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
022300D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
022300E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
022300F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

When we run the program the memory location will be filed with the payload, notice the value "MZ" at the beginning of the ASCII field, this means that the payload is an executable.



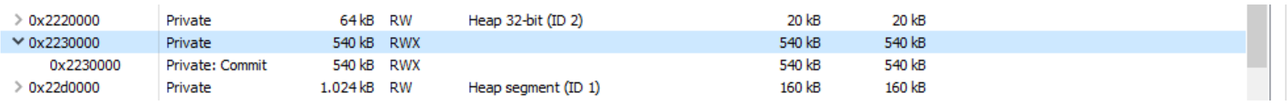
```
Address   Hex                                              ASCII
02230000  4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
02230010  B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
02230020  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
02230030  00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00  ................
02230040  0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
02230050  69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
02230060  74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
02230070  6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
02230080  AD D1 64 8A E9 B0 0A D9 E9 B0 0A D9 E9 B0 0A D9  .Ñd.é°.Ùé°.Ùé°.Ù
02230090  2A BF 55 D9 EA B0 0A D9 8A 92 20 D9 E8 B0 0A D9  *¿UÙê°.Ù.. Ùè°.Ù
022300A0  CE 76 67 D9 E8 B0 0A D9 F7 E2 8E D9 E8 B0 0A D9  ÎvgÙè°.Ù÷â.Ùè°.Ù
022300B0  13 93 4A D9 EE B0 0A D9 13 93 13 D9 EB B0 0A D9  ..JÙî°.Ù...Ùë°.Ù
022300C0  2A BF 57 D9 F6 B0 0A D9 E9 B0 0B D9 17 B0 0A D9  *¿WÙö°.Ùé°.Ù.°.Ù
022300D0  F2 2D A4 D9 C7 B0 0A D9 F2 2D 90 D9 E8 B0 0A D9  ò-¤ÙÇ°.Ùò-.Ùè°.Ù
022300E0  E9 B0 9D D9 E8 B0 0A D9 F2 2D 97 D9 E8 B0 0A D9  é°.Ùè°.Ùò-.Ùè°.Ù
022300F0  52 69 63 68 E9 B0 0A D9 00 00 00 00 00 00 00 00  Riché°.Ù........
```

We now need to go to the memory location where the payload is loaded. So we use Process Hacker and navigate to the right tab "demo2_simbdabin".



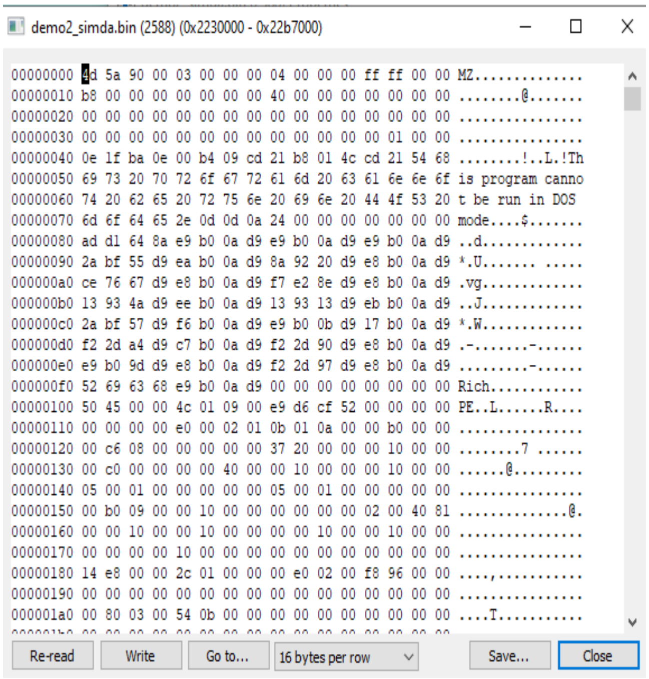
explorer.exe	4324	0,10		70,48 MB	DESKTOP-UV...\FlareVM	Windows Explorer
VBoxTray.exe	5132		28 B/s	2,54 MB	DESKTOP-UV...\FlareVM	VirtualBox Guest Additions Tra...
x32dbg.exe	3456	0,10		43,74 MB	DESKTOP-UV...\FlareVM	x64dbg
demo2_simda.bin	2588	49,29		3,47 MB	DESKTOP-UV...\FlareVM	Command line RAR
ProcessHacker.exe	2504	1,25		15,88 MB	DESKTOP-UV...\FlareVM	Process Hacker

And we locate the memory location 02230000 or 0x223000 in memory.



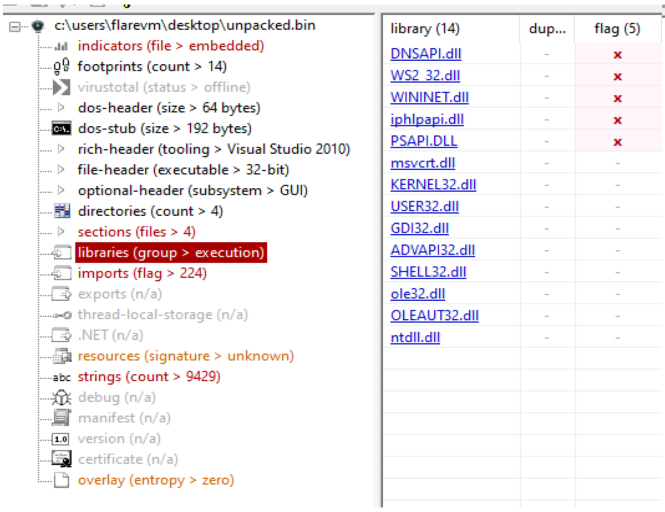
0x2220000	Private	64 kB	RW	Heap 32-bit (ID 2)	20 kB	20 kB
0x2230000	Private	540 kB	RWX		540 kB	540 kB
0x2230000	Private: Commit	540 kB	RWX		540 kB	540 kB
0x22d0000	Private	1.024 kB	RW	Heap segment (ID 1)	160 kB	160 kB

We save the payload to our desktop under a recognizable name "unpacked.bin".



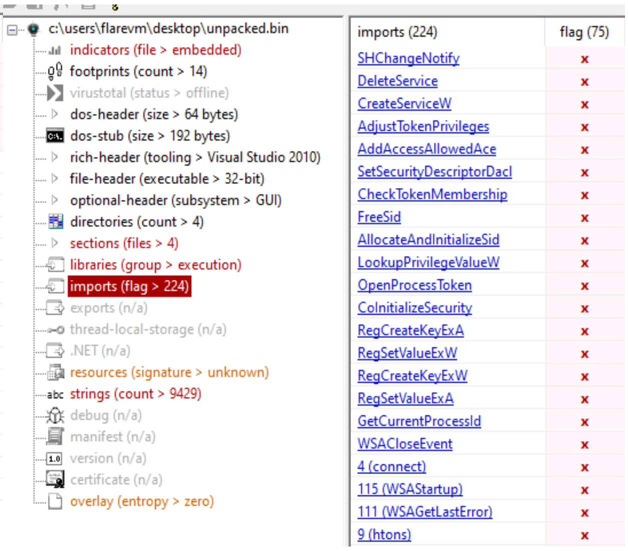
```
00000000 4d 5a 90 00 03 00 00 00 04 00 00 00 ff ff 00 00 MZ..............
00000010 b8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ........@.......
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 00 00 00 00 00 00 00 00 00 01 00 00 ................
00000040 0e 1f ba 0e 00 b4 09 cd 21 b8 01 4c cd 21 54 68 ........!..L.!Th
00000050 69 73 20 70 72 6f 67 72 61 6d 20 63 61 6e 6e 6f is program canno
00000060 74 20 62 65 20 72 75 6e 20 69 6e 20 44 4f 53 20 t be run in DOS
00000070 6d 6f 64 65 2e 0d 0d 0a 24 00 00 00 00 00 00 00 mode....$.......
00000080 ad d1 64 8a e9 b0 0a d9 e9 b0 0a d9 e9 b0 0a d9 ..d.............
00000090 2a bf 55 d9 ea b0 0a d9 8a 92 20 d9 e8 b0 0a d9 *.U....... .....
000000a0 ce 76 67 d9 e8 b0 0a d9 f7 e2 8e d9 e8 b0 0a d9 .vg.............
000000b0 13 93 4a d9 ee b0 0a d9 13 93 13 d9 eb b0 0a d9 ..J.............
000000c0 2a bf 57 d9 f6 b0 0a d9 e9 b0 0b d9 17 b0 0a d9 *.W.............
000000d0 f2 2d a4 d9 c7 b0 0a d9 f2 2d 90 d9 e8 b0 0a d9 .-.......-......
000000e0 e9 b0 9d d9 e8 b0 0a d9 f2 2d 97 d9 e8 b0 0a d9 .........-......
000000f0 52 69 63 68 e9 b0 0a d9 00 00 00 00 00 00 00 00 Rich............
00000100 50 45 00 00 4c 01 09 00 e9 d6 cf 52 00 00 00 00 PE..L......R....
00000110 00 00 00 00 e0 00 02 01 0b 01 0a 00 00 b0 00 00 ................
00000120 00 c6 08 00 00 00 00 00 37 20 00 00 00 10 00 00 ........7 ......
00000130 00 c0 00 00 00 00 40 00 00 10 00 00 00 10 00 00 ......@.........
00000140 05 00 01 00 00 00 00 00 05 00 01 00 00 00 00 00 ................
00000150 00 b0 09 00 00 10 00 00 00 00 00 00 02 00 40 81 ..............@.
00000160 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 ................
00000170 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 ................
00000180 14 e8 00 00 2c 01 00 00 00 e0 02 00 f8 96 00 00 ....,...........
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
000001a0 00 80 03 00 54 0b 00 00 00 00 00 00 00 00 00 00 ....T...........
```

We can now use PEStudio again but this time we notice a lot more information.



c:\users\flarevm\desktop\unpacked.bin
- indicators (file > embedded)
- footprints (count > 14)
- virustotal (status > offline)
- dos-header (size > 64 bytes)
- dos-stub (size > 192 bytes)
- rich-header (tooling > Visual Studio 2010)
- file-header (executable > 32-bit)
- optional-header (subsystem > GUI)
- directories (count > 4)
- sections (files > 4)
- libraries (group > execution)
- imports (flag > 224)
- exports (n/a)
- thread-local-storage (n/a)
- .NET (n/a)
- resources (signature > unknown)
- strings (count > 9429)
- debug (n/a)
- manifest (n/a)
- version (n/a)
- certificate (n/a)
- overlay (entropy > zero)

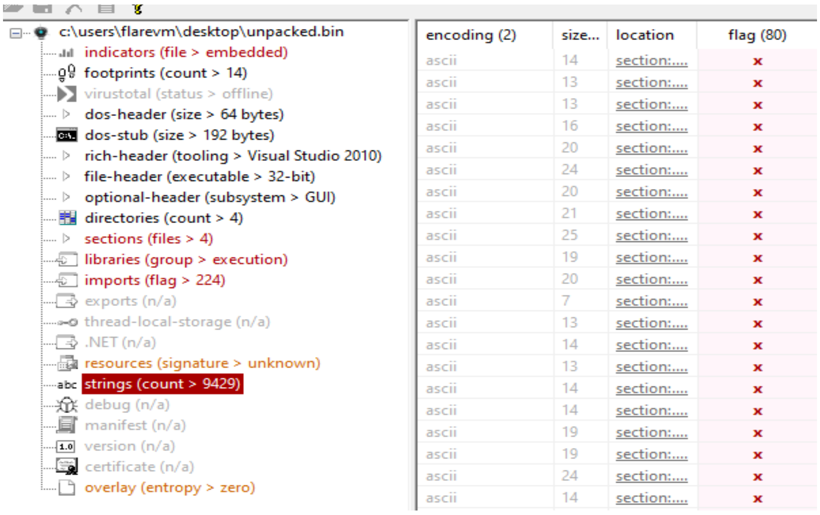
library (14)	dup...	flag (5)
DNSAPI.dll	-	x
WS2_32.dll	-	x
WININET.dll	-	x
iphlpapi.dll	-	x
PSAPI.DLL	-	x
msvcrt.dll	-	-
KERNEL32.dll	-	-
USER32.dll	-	-
GDI32.dll	-	-
ADVAPI32.dll	-	-
SHELL32.dll	-	-
ole32.dll	-	-
OLEAUT32.dll	-	-
ntdll.dll	-	-



c:\users\flarevm\desktop\unpacked.bin
- indicators (file > embedded)
- footprints (count > 14)
- virustotal (status > offline)
- dos-header (size > 64 bytes)
- dos-stub (size > 192 bytes)
- rich-header (tooling > Visual Studio 2010)
- file-header (executable > 32-bit)
- optional-header (subsystem > GUI)
- directories (count > 4)
- sections (files > 4)
- libraries (group > execution)
- imports (flag > 224)
- exports (n/a)
- thread-local-storage (n/a)
- .NET (n/a)
- resources (signature > unknown)
- strings (count > 9429)
- debug (n/a)
- manifest (n/a)
- version (n/a)
- certificate (n/a)
- overlay (entropy > zero)

imports (224)	flag (75)
SHChangeNotify	x
DeleteService	x
CreateServiceW	x
AdjustTokenPrivileges	x
AddAccessAllowedAce	x
SetSecurityDescriptorDacl	x
CheckTokenMembership	x
FreeSid	x
AllocateAndInitializeSid	x
LookupPrivilegeValueW	x
OpenProcessToken	x
CoInitializeSecurity	x
RegCreateKeyExA	x
RegSetValueExW	x
RegCreateKeyExW	x
RegSetValueExA	x
GetCurrentProcessId	x
WSACloseEvent	x
4 (connect)	x
115 (WSAStartup)	x
111 (WSAGetLastError)	x
9 (htons)	x

This is the real ammount of malicious libraries, imports and strings. Most libraries and imports have descriptions regarding networking which we will discover later during our dynamic analysis.
We see a lot of uncommon strings in the list such as Create/Delete service, RegSetValue/Create/Delete Key, WSAStartup, Write/Delte file and ShellExecute.

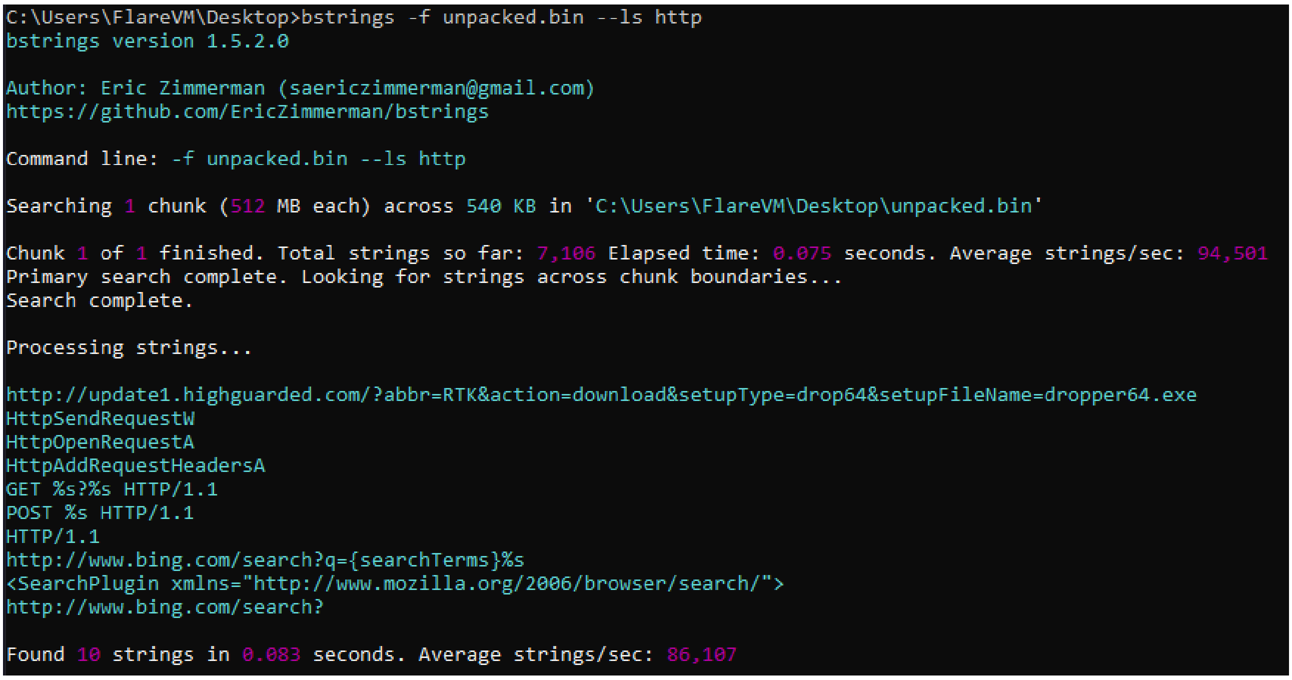


c:\users\flarevm\desktop\unpacked.bin
- indicators (file > embedded)
- footprints (count > 14)
- virustotal (status > offline)
- dos-header (size > 64 bytes)
- dos-stub (size > 192 bytes)
- rich-header (tooling > Visual Studio 2010)
- file-header (executable > 32-bit)
- optional-header (subsystem > GUI)
- directories (count > 4)
- sections (files > 4)
- libraries (group > execution)
- imports (flag > 224)
- exports (n/a)
- thread-local-storage (n/a)
- .NET (n/a)
- resources (signature > unknown)
- strings (count > 9429)
- debug (n/a)
- manifest (n/a)
- version (n/a)
- certificate (n/a)
- overlay (entropy > zero)

encoding (2)	size...	location	flag (80)
ascii	14	section:....	x
ascii	13	section:....	x
ascii	13	section:....	x
ascii	16	section:....	x
ascii	20	section:....	x
ascii	24	section:....	x
ascii	20	section:....	x
ascii	21	section:....	x
ascii	25	section:....	x
ascii	19	section:....	x
ascii	20	section:....	x
ascii	7	section:....	x
ascii	13	section:....	x
ascii	14	section:....	x
ascii	13	section:....	x
ascii	14	section:....	x
ascii	14	section:....	x
ascii	19	section:....	x
ascii	19	section:....	x
ascii	24	section:....	x
ascii	14	section:....	x

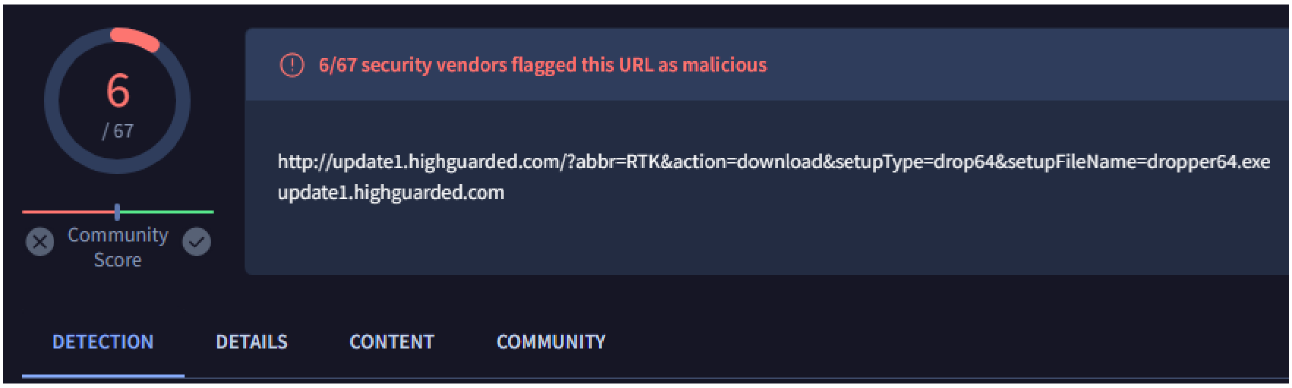
We use bstrings to do further string analysis and we look for http and .com.

With http we notice that it tries to go to a website update1.highguarded.com and download a file called dropper64.exe.



```
C:\Users\FlareVM\Desktop>bstrings -f unpacked.bin --ls http
bstrings version 1.5.2.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/bstrings

Command line: -f unpacked.bin --ls http

Searching 1 chunk (512 MB each) across 540 KB in 'C:\Users\FlareVM\Desktop\unpacked.bin'

Chunk 1 of 1 finished. Total strings so far: 7,106 Elapsed time: 0.075 seconds. Average strings/sec: 94,501
Primary search complete. Looking for strings across chunk boundaries...
Search complete.

Processing strings...

http://update1.highguarded.com/?abbr=RTK&action=download&setupType=drop64&setupFileName=dropper64.exe
HttpSendRequestW
HttpOpenRequestA
HttpAddRequestHeadersA
GET %s?%s HTTP/1.1
POST %s HTTP/1.1
HTTP/1.1
http://www.bing.com/search?q={searchTerms}%s
<SearchPlugin xmlns="http://www.mozilla.org/2006/browser/search/">
http://www.bing.com/search?

Found 10 strings in 0.083 seconds. Average strings/sec: 86,107
```

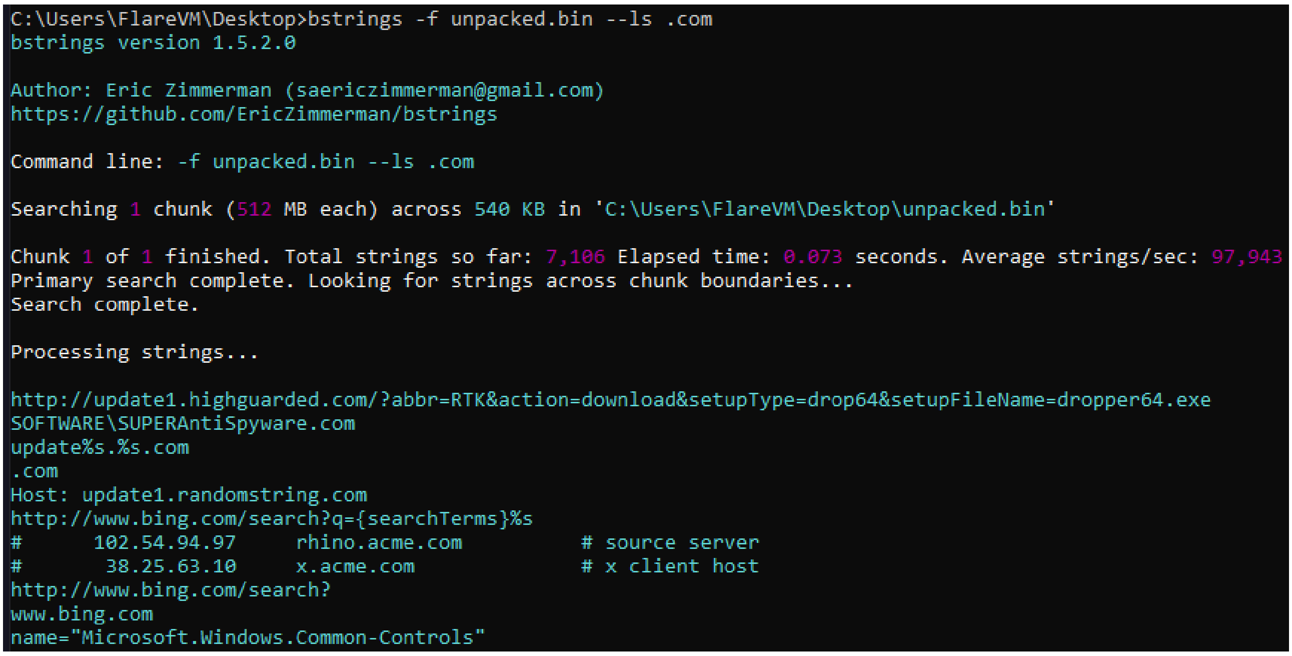
This dropper64.exe is malicious according to virustotal.



6/67 security vendors flagged this URL as malicious

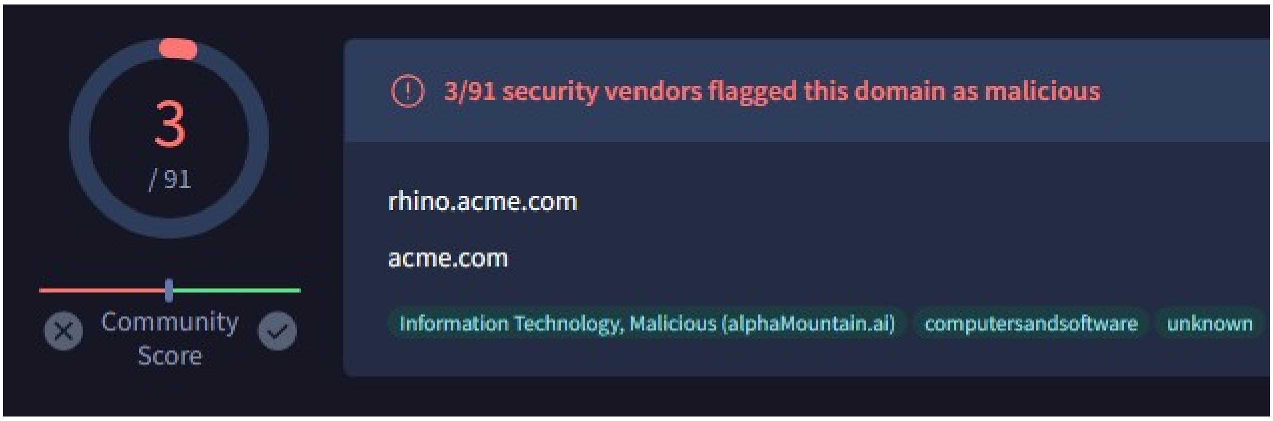
http://update1.highguarded.com/?abbr=RTK&action=download&setupType=drop64&setupFileName=dropper64.exe
update1.highguarded.com

Then for the .com check, we notice another domain rhino.acme.com and x.acme.com



```
C:\Users\FlareVM\Desktop>bstrings -f unpacked.bin --ls .com
bstrings version 1.5.2.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/bstrings

Command line: -f unpacked.bin --ls .com

Searching 1 chunk (512 MB each) across 540 KB in 'C:\Users\FlareVM\Desktop\unpacked.bin'

Chunk 1 of 1 finished. Total strings so far: 7,106 Elapsed time: 0.073 seconds. Average strings/sec: 97,943
Primary search complete. Looking for strings across chunk boundaries...
Search complete.

Processing strings...

http://update1.highguarded.com/?abbr=RTK&action=download&setupType=drop64&setupFileName=dropper64.exe
SOFTWARE\SUPERAntiSpyware.com
update%s.%s.com
.com
Host: update1.randomstring.com
http://www.bing.com/search?q={searchTerms}%s
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host
http://www.bing.com/search?
www.bing.com
name="Microsoft.Windows.Common-Controls"
```

Rhino and X both come up as malicious for the domain acme.com.



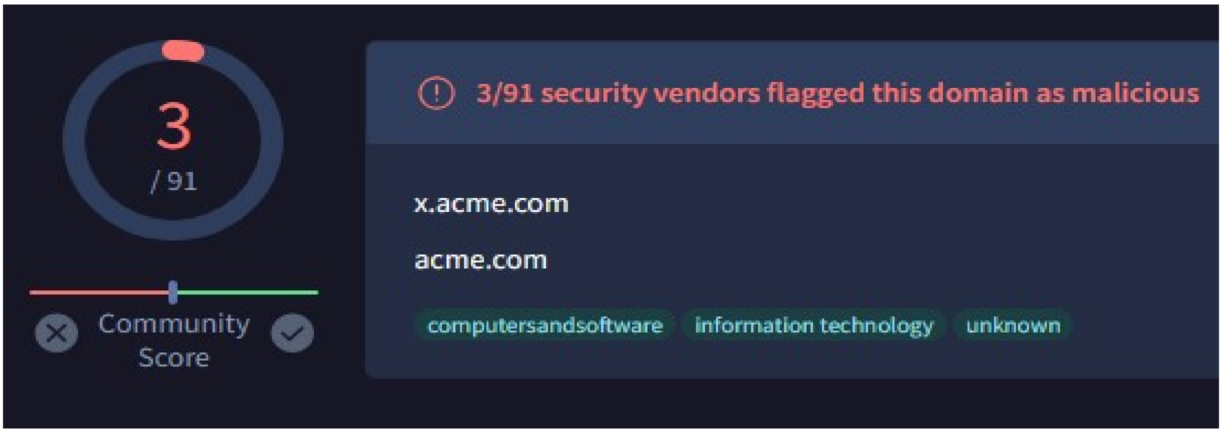
3/91 security vendors flagged this domain as malicious

rhino.acme.com

acme.com

Information Technology, Malicious (alphaMountain.ai) computersandsoftware unknown

And so does x.acme.com



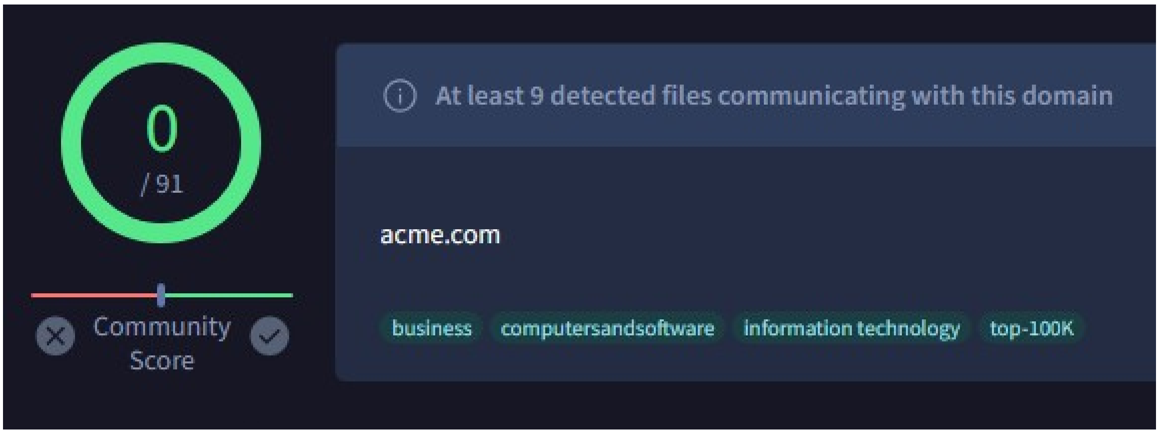
3/91 security vendors flagged this domain as malicious

x.acme.com

acme.com

computersandsoftware information technology unknown

Even tho the domain acme.com seems to be clean.



ⓘ At least 9 detected files communicating with this domain

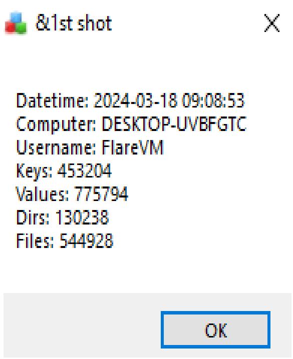
acme.com

So far we have gathered enough IoC's from our static analysis, let's move over to dynamic analysis and see what we can find.

Dynamic Analysis

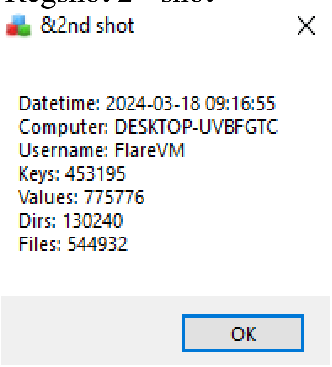
We start up Fakenet, Procmon and Regshot and take our first shot of our registry. We run the malware for a couple minutes and then we take our second shot to compare the two registry shots. We notice some differences in the two.

Regshot 1st shot



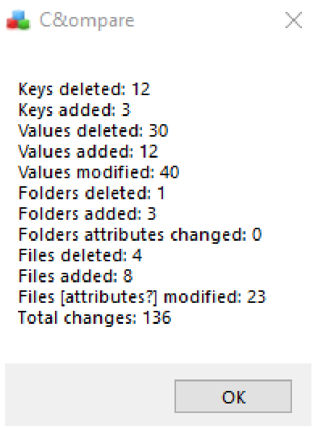
```
Datetime: 2024-03-18 09:08:53
Computer: DESKTOP-UVBFGTC
Username: FlareVM
Keys: 453204
Values: 775794
Dirs: 130238
Files: 544928
```

Regshot 2nd shot



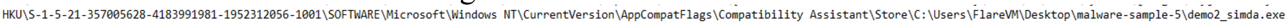
```
Datetime: 2024-03-18 09:16:55
Computer: DESKTOP-UVBFGTC
Username: FlareVM
Keys: 453195
Values: 775776
Dirs: 130240
Files: 544932
```

Comparison



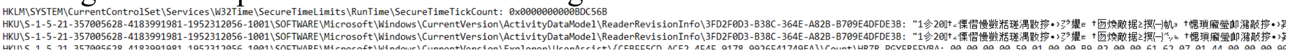
```
Keys deleted: 12
Keys added: 3
Values deleted: 30
Values added: 12
Values modified: 40
Folders deleted: 1
Folders added: 3
Folders attributes changed: 0
Files deleted: 4
Files added: 8
Files [attributes?] modified: 23
Total changes: 136
```

In the results we notice a value being added for simbda.exe



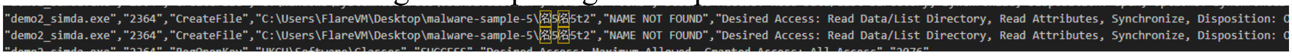
```
HKU\S-1-5-21-357005628-4183991981-1952312056-1001\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Compatibility Assistant\Store\C:\Users\FlareVM\Desktop\malware-sample-5\demo2_simda.exe
```

Among with multiple lines of chinese writing.



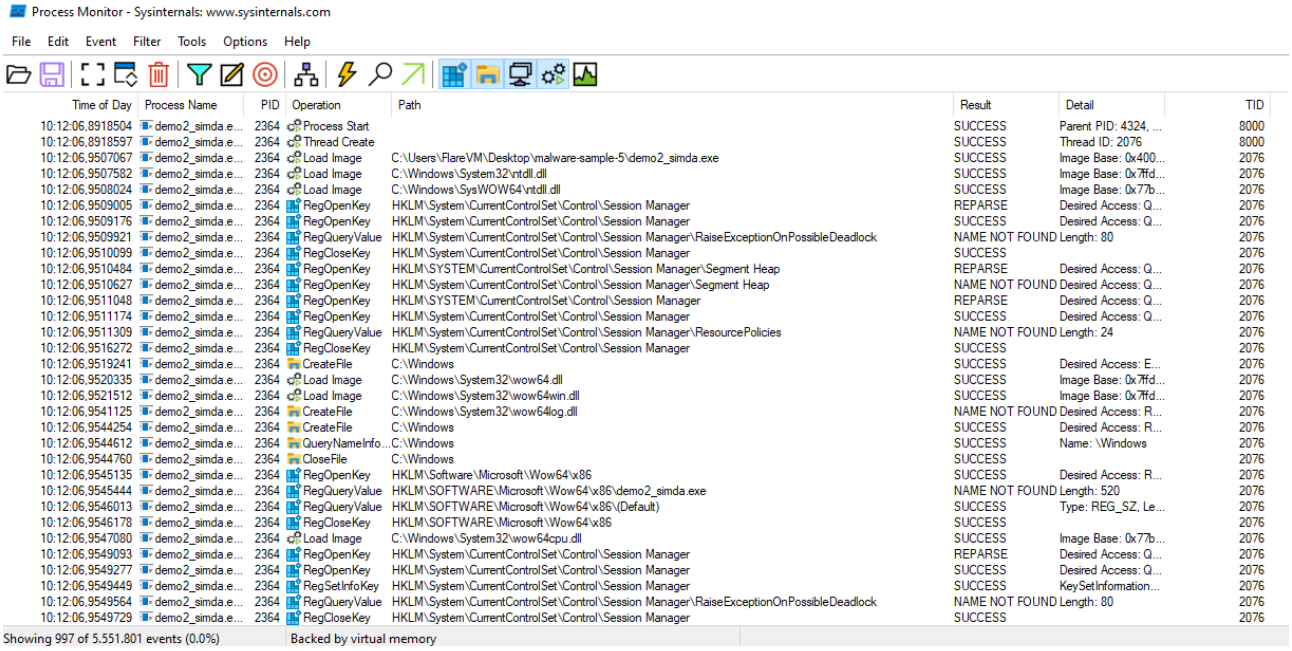
```
HKLM\SYSTEM\CurrentControlSet\Services\W32Time\SecureTimeLimits\RunTime\SecureTimeTickCount: 0x0000000000BDC56B
HKU\S-1-5-21-357005628-4183991981-1952312056-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\ActivityDataModel\ReaderRevisionInfo\3FD2F0D3-B38C-364E-A82B-B709E4DFDE3B: "1参20▯†⊿慄慴慢敳湉瑳湡散摻•›ジラ耀ᵉ †㘧煥敩据≥摂(−)帆ɜ †慄瑯瘍瑡卹湰散摻•›
HKU\S-1-5-21-357005628-4183991981-1952312056-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\ActivityDataModel\ReaderRevisionInfo\3FD2F0D3-B38C-364E-A82B-B709E4DFDE3B: "1参20▯†⊿慄慴慢敳湉瑳湡散摻•›ジラ耀ᵉ †㘧煥敩据≥摂(−)ベゥ% †慄瑯瘍瑡卹湰散摻•›
HKU\S-1-5-21-357005628-4183991981-1952312056-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\HRZR_PGYFRFFVBA: 00 00 00 00 50 01 00 00 B9 02 00 00 61 62 07 01 44 00 00 00 9
```

We notice more chinese writing when opening the output file in visualstudio code.



```
"demo2_simda.exe","2364","CreateFile","C:\Users\FlareVM\Desktop\malware-sample-5\名5名5t2","NAME NOT FOUND","Desired Access: Read Data/List Directory, Read Attributes, Synchronize, Disposition: O
"demo2_simda.exe","2364","CreateFile","C:\Users\FlareVM\Desktop\malware-sample-5\名5名5t2","NAME NOT FOUND","Desired Access: Read Data/List Directory, Read Attributes, Synchronize, Disposition: O
```

In Procmon we notice 997 events correlated with simda.exe. Most of these events regard operations on registry/windows .dll files.

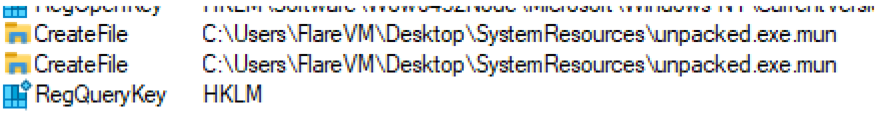


Process Monitor - Sysinternals: www.sysinternals.com

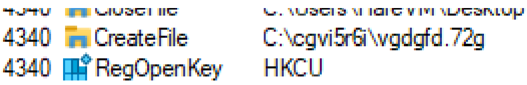
Time of Day	Process Name	PID	Operation	Path	Result	Detail	TID
10:12:06,8918504	demo2_simda.e...	2364	Process Start		SUCCESS	Parent PID: 4324, ...	8000
10:12:06,8918597	demo2_simda.e...	2364	Thread Create		SUCCESS	Thread ID: 2076	8000
10:12:06,9507067	demo2_simda.e...	2364	Load Image	C:\Users\FlareVM\Desktop\malware-sample-5\demo2_simda.exe	SUCCESS	Image Base: 0x400...	2076
10:12:06,9507582	demo2_simda.e...	2364	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7ffd...	2076
10:12:06,9508024	demo2_simda.e...	2364	Load Image	C:\Windows\SysWOW64\ntdll.dll	SUCCESS	Image Base: 0x77b...	2076
10:12:06,9509005	demo2_simda.e...	2364	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Q...	2076
10:12:06,9509176	demo2_simda.e...	2364	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Q...	2076
10:12:06,9509921	demo2_simda.e...	2364	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\RaiseExceptionOnPossibleDeadlock	NAME NOT FOUND	Length: 80	2076
10:12:06,9510099	demo2_simda.e...	2364	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS		2076
10:12:06,9510484	demo2_simda.e...	2364	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Segment Heap	REPARSE	Desired Access: Q...	2076
10:12:06,9510627	demo2_simda.e...	2364	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager\Segment Heap	NAME NOT FOUND	Desired Access: Q...	2076
10:12:06,9511048	demo2_simda.e...	2364	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Q...	2076
10:12:06,9511174	demo2_simda.e...	2364	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Q...	2076
10:12:06,9511309	demo2_simda.e...	2364	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\ResourcePolicies	NAME NOT FOUND	Length: 24	2076
10:12:06,9516272	demo2_simda.e...	2364	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS		2076
10:12:06,9519241	demo2_simda.e...	2364	CreateFile	C:\Windows	SUCCESS	Desired Access: E...	2076
10:12:06,9520335	demo2_simda.e...	2364	Load Image	C:\Windows\System32\wow64.dll	SUCCESS	Image Base: 0x7ffd...	2076
10:12:06,9521512	demo2_simda.e...	2364	Load Image	C:\Windows\System32\wow64win.dll	SUCCESS	Image Base: 0x7ffd...	2076
10:12:06,9541125	demo2_simda.e...	2364	CreateFile	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	Desired Access: R...	2076
10:12:06,9544254	demo2_simda.e...	2364	CreateFile	C:\Windows	SUCCESS	Desired Access: R...	2076
10:12:06,9544612	demo2_simda.e...	2364	QueryNameInfo...	C:\Windows	SUCCESS	Name: \Windows	2076
10:12:06,9544760	demo2_simda.e...	2364	CloseFile	C:\Windows	SUCCESS		2076
10:12:06,9545135	demo2_simda.e...	2364	RegOpenKey	HKLM\Software\Microsoft\Wow64\x86	SUCCESS	Desired Access: R...	2076
10:12:06,9545444	demo2_simda.e...	2364	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\x86\demo2_simda.exe	NAME NOT FOUND	Length: 520	2076
10:12:06,9546013	demo2_simda.e...	2364	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\x86\(Default)	SUCCESS	Type: REG_SZ, Le...	2076
10:12:06,9546178	demo2_simda.e...	2364	RegCloseKey	HKLM\SOFTWARE\Microsoft\Wow64\x86	SUCCESS		2076
10:12:06,9547080	demo2_simda.e...	2364	Load Image	C:\Windows\System32\wow64cpu.dll	SUCCESS	Image Base: 0x77b...	2076
10:12:06,9549093	demo2_simda.e...	2364	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Q...	2076
10:12:06,9549277	demo2_simda.e...	2364	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Q...	2076
10:12:06,9549449	demo2_simda.e...	2364	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	KeySetInformation...	2076
10:12:06,9549564	demo2_simda.e...	2364	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\RaiseExceptionOnPossibleDeadlock	NAME NOT FOUND	Length: 80	2076
10:12:06,9549729	demo2_simda.e...	2364	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS		2076

Showing 997 of 5.551.801 events (0.0%)   Backed by virtual memory

Also the creation of files under C:\

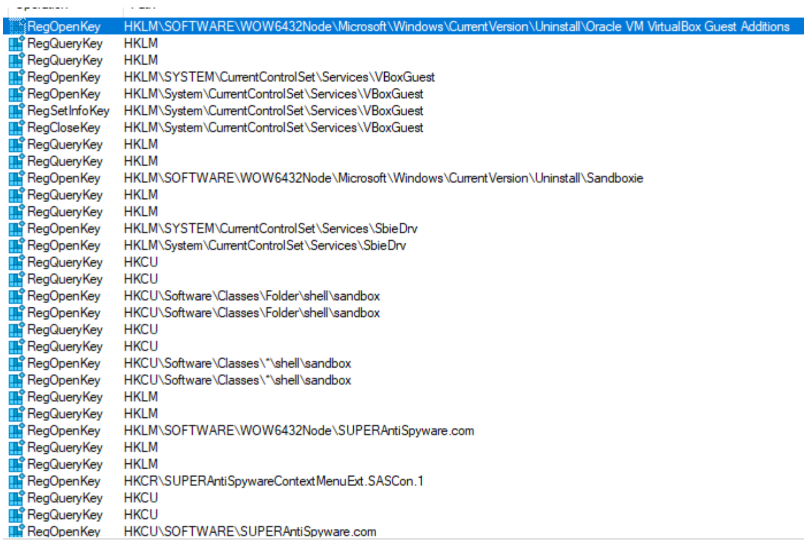


CreateFile	C:\Users\FlareVM\Desktop\SystemResources\unpacked.exe.mun
CreateFile	C:\Users\FlareVM\Desktop\SystemResources\unpacked.exe.mun
RegQueryKey	HKLM



4340	CreateFile	C:\cgvi5r6i\vgdgfd.72g
4340	RegOpenKey	HKCU

And trying to find evidence of being ran in a virtual environment.



```
RegOpenKey     HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Oracle VM VirtualBox Guest Additions
RegQueryKey    HKLM
RegQueryKey    HKLM
RegOpenKey     HKLM\SYSTEM\CurrentControlSet\Services\VBoxGuest
RegOpenKey     HKLM\System\CurrentControlSet\Services\VBoxGuest
RegSetInfoKey  HKLM\System\CurrentControlSet\Services\VBoxGuest
RegCloseKey    HKLM\System\CurrentControlSet\Services\VBoxGuest
RegQueryKey    HKLM
RegQueryKey    HKLM
RegOpenKey     HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Sandboxie
RegQueryKey    HKLM
RegQueryKey    HKLM
RegOpenKey     HKLM\SYSTEM\CurrentControlSet\Services\SbieDrv
RegOpenKey     HKLM\System\CurrentControlSet\Services\SbieDrv
RegQueryKey    HKCU
RegQueryKey    HKCU
RegOpenKey     HKCU\Software\Classes\Folder\shell\sandbox
RegOpenKey     HKCU\Software\Classes\Folder\shell\sandbox
RegQueryKey    HKCU
RegQueryKey    HKCU
RegOpenKey     HKCU\Software\Classes\*\shell\sandbox
RegOpenKey     HKCU\Software\Classes\*\shell\sandbox
RegQueryKey    HKLM
RegQueryKey    HKLM
RegOpenKey     HKLM\SOFTWARE\WOW6432Node\SUPERAntiSpyware.com
RegQueryKey    HKLM
RegQueryKey    HKLM
RegOpenKey     HKCR\SUPERAntiSpywareContextMenuExt.SASCon.1
RegQueryKey    HKCU
RegQueryKey    HKCU
RegOpenKey     HKCU\SOFTWARE\SUPERAntiSpyware.com
```

During the analysis of this sample file it it clear that the file is indeed malicious.
A gathering of the IoC's from this analysis:

Host IoCs
- Hash analysis gave a 62/71 on virustotal
- Hidden payload that contains malicious strings
- Adjustments to the registry
- Chinese file names
- Creation of multiple files
- Uninstalling guest tools for virtual environment
- Uninstalling detection/analysis tools

Network IoC's
- Navigating to malicious domains (3/91 on virustotal)
- Downloading malicious .exe file (6/67 on virustotal)