A “Wotsuper” Infostealer

March 23, 2025

Kollin Labowski

klabowski@ufl.edu

Malware Reverse Engineering: Practical 2

Section 1: Executive Summary

The malware sample “sample2.exe” is a launcher created with Smart Install Maker that creates and runs two new executable files “wotsuper.exe” and “wotsuper1.exe”. The files are compressed into a CAB archive that is stored in sample2.exe, and these files are extracted at runtime. sample2.exe installs these files as a program under the name “wotsuper 2.1”, and modifies the appropriate registry keys to do so. sample2.exe also starts two Internet Explorer processes connecting to iplogger.org.

The first installed program, wotsuper.exe, is also a launcher malware; this one runs a program stored in its .data section into memory. This payload is decrypted in memory using a basic decryption algorithm, revealing a shellcode that is then run. This shellcode attempts to communicate with “telete.in”, which is a known malicious domain associated with the Raccoon malware family. Therefore, the shellcode loaded by this program appears to be a Raccoon program, which is an information stealer malware.

The second installed program, wotsuper1.exe, is another information stealer that appears to be part of the Arkei malware family, most likely Vidar, to be specific. This malware logs all sorts of information from the infected machine, such as currently running processes, installed software, cryptocurrency wallet passwords, internet cookies, and much more. These logs can be found in C:\ProgramData. The program tries to communicate with a known C2 server “manillamemories.com”, likely to send the data it has collected from the machine to the malware author.

sample2.exe and its installed files can be detected by looking for communications with the domains “telete.in”, “manillamemories”, or either of the iplogger.org addresses discussed in this report. Another indicator of compromise is the installation of a program called “wotsuper 2.1”, along with corresponding registry key changes and files added to the C:\Program Files directory.

We will also briefly analyze another file sample2b.exe, which appears to be another information stealer that is a variant of Raccoon. It communicates with the domains telegatt.top, telegka.top, and telegin.top.

Section 2: Static Analysis

Section 2A: Basic static analysis of sample2.exe

property	value
md5	DD1B734796B4AA40AF46B4D69E1E2DA2
sha1	D5273BE84DFA0C54FC9CEFFF7BCC24FED3E20E1C
sha256	361411E6321C45C845669AC89E32FEEC0BDD97916B5D73F508C43576B8A15A20
md5-without-overlay	wait...
sha1-without-overlay	wait...
sha256-without-overlay	wait...
first-bytes-hex	4D 5A 50 00 02 00 00 00 04 00 0F 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 1A 00 00 00 00 00
first-bytes-text	M Z P .. .. @ .. ..
file-size	892326 (bytes)
size-without-overlay	wait...
entropy	7.876
imphash	9623D7EC9B5866D2710C111879DB45FE
signature	BobSoft Mini Delphi -> BoB / BobSoft
entry-point	55 8B EC 83 C4 F0 B8 88 53 42 00 E8 24 F2 FD FF B8 C8 54 42 00 E8 2A 1C FE FF 8B 15 40 88 42 00 89
file-version	2.1
description	wotsuper 2.1 Installation
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x2A425E19 (Fri Jun 19 18:22:17 1992)
debugger-stamp	n/a
resources-stamp	
exports-stamp	n/a
version-stamp	empty
certificate-stamp	n/a

Figure 1: PEStudio output for sample2.exe

We can see in the above screenshot that PEStudio has identified this malware sample as an **executable** file. The program was compiled for **32-bit** architectures and appears to have been compiled on **June 19, 1992** (we will see evidence later to suggest this date is inaccurate). The program has **GUI** capabilities. The description says "**wotsuper 2.1 installation**", so for the remainder of this report, we will sometimes refer to this malware sample as the "wotsuper installer". Searching the internet for "wotsuper", we can find a website (https://wotsuper.software.informer.com/) that references a program with the same name. Perhaps this is the original source of sample2.exe.

property	value	value	value	value	value	value	value	value
name	CODE	DATA	BSS	.idata	.tls	.rdata	.reloc	.rsrc
md5	5E14E4EDE2E2215BC7D7283...	ABAFCBFBD7F8AC0226CA4...	n/a	A4E0AC39D5ED487CEEA059...	n/a	C4FDD0C5C9EFB616FCC85D...	867A1120317D51734587A74...	228790ED3FB3790E5D7F22D...
entropy	6.594	3.794	n/a	4.886	n/a	0.204	6.587	4.749
file-ratio (20.25%)	16.70 %	1.20 %	n/a	0.69 %	n/a	0.06 %	0.75 %	0.86 %
raw-address	0x00000400	0x00024A00	0x00027400	0x00027400	0x00028C00	0x00028C00	0x00028E00	0x0002A800
raw-size (180736 bytes)	0x00024600 (148992 bytes)	0x00002A00 (10752 bytes)	0x00000000 (0 bytes)	0x00001800 (6144 bytes)	0x00000000 (0 bytes)	0x00000200 (512 bytes)	0x00001A00 (6656 bytes)	0x00001E00 (7680 bytes)
virtual-address	0x00401000	0x00426000	0x00429000	0x0042B000	0x0042D000	0x0042E000	0x0042F000	0x00431000
virtual-size (183149 bytes)	0x000244CC (148684 bytes)	0x00002894 (10388 bytes)	0x000010F5 (4341 bytes)	0x00001798 (6040 bytes)	0x00000008 (8 bytes)	0x00000018 (24 bytes)	0x00001884 (6276 bytes)	0x00001CDC (7388 bytes)
entry-point	0x00025468	-						
characteristics	0x60000020	0xC0000040	0xC0000000	0xC0000040	0xC0000000	0x50000040	0x50000040	0x50000040
writable	-	x	x	x	x	-	-	-
executable	x	-	-	-	-	-	-	-
shareable	-	-	-	-	-	x	x	x
discardable	-	-	-	-	-	-	-	-

Figure 2: The section tab of PEStudio for sample2.exe

This PE file contains 8 different sections: CODE, DATA, BSS, .idata, .tls, .rdata, .reloc, and .rsrc. The CODE section likely contains the main text of the program (in place of a typical .text section). The DATA section may contain any global variables that are assigned a value prior to compilation (in place of a typical .data section). The BSS section

may contain any global variables without an initial value declared prior to compilation (in place of a typical .bss section). The .idata section contains imported libraries and other functions. The .tls section can be used to allocate variables that are unique to individual threads (it is also unusual for this section to appear in benign programs). The .rdata section contains read-only constants. The .reloc section contains information that can be used to relocate the program if it cannot be loaded at its preferred base address. The .rsrc section contains resources that may be used for various UI elements, like images and icons.

There are a few indicators that hint towards sample2.exe being packed, or at least obfuscated in some way. In *Figure 1*, we see the malware has the signature "**Bobsoft Mini Delphi -> Bob / BobSoft**". This seems to refer to a known Delphi packing tool, which can be found at https://unprotect.it/technique/bobsoft-mini-delphi-packer/. Another indicator suggesting the malware is packed is its overall entropy of 7.876 (also seen in *Figure 1*), which is very high.

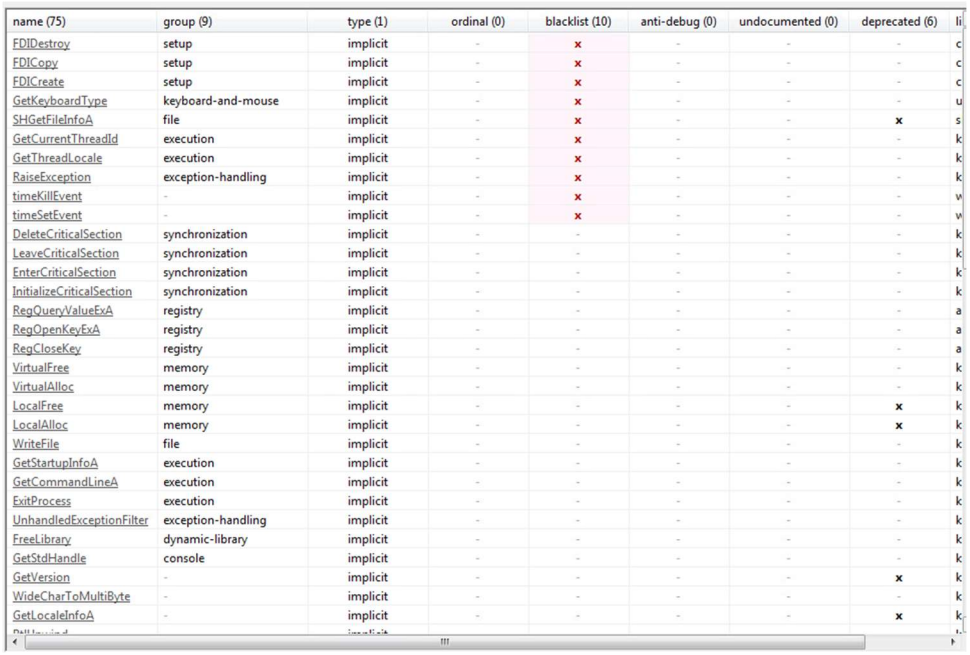
Despite the apparent use of packing to obfuscate sample2.exe, there are multiple indicators that suggest the packing is not particularly sophisticated. One such indicator is that the entropy of all sections in the PE file is below 7, which is standard for a normal program. We can also observe that the difference between the raw and virtual size of each section is small, which is also typical of standard programs. (Note that the file ratio of these combined sections is only about 20%, indicating the actual packed payload is stored in the remaining 80% of the file.) Furthermore, we will see there are a total of 75 imports, and over 10,000 strings in this program, which is an unusually high number for a packed program.

property	value
md5	86A248DC2BD0840445581D445640282F
sha1	54211CAFD97D19E282045D7CFFC58389FB24DEFB
sha256	0F4F4A5A03AD8144CB91257933AC8B246566CAF0C85664E7673E662F93FAB6...
entropy	7.999
file-offset	0x0002C600
size	710566 (bytes)
signature	Smart-installer
first-bytes-hex	53 6D 61 72 74 20 49 6E 73 74 61 6C 6C 20 4D 61 6B 65 72 20 76 2E 20 35 2E 30 ...
first-bytes-text	S m a r t I n s t a l l M a k e r v . 5 . 0 4 . . 0 . . 0 . .
file-ratio	79.63 %

Figure 3: The overlay tab of PEStudio for sample2.exe

The overlay section in PEStudio reveals the reason for the strange obfuscation behavior observed. The signature "**Smart-installer**" and text "**Smart Install Maker v. 5.04**" indicate that this program is an installer program that was created using Smart Install Maker (http://www.sminstall.com/). Note that this overlay has incredibly high entropy (7.999), and it makes up 79.63% of the file contents, explaining the high overall file entropy despite the smaller PE section entropy. Interestingly, according to the previously linked

website, Smart Install Maker v. 5.04 was released in 2011, nearly 20 years after the supposed compile date.



name (75)	group (9)	type (1)	ordinal (0)	blacklist (10)	anti-debug (0)	undocumented (0)	deprecated (6)	li
FDIDestroy	setup	implicit	-	x	-	-	-	c
FDICopy	setup	implicit	-	x	-	-	-	c
FDICreate	setup	implicit	-	x	-	-	-	c
GetKeyboardType	keyboard-and-mouse	implicit	-	x	-	-	-	u
SHGetFileInfoA	file	implicit	-	x	-	-	x	s
GetCurrentThreadId	execution	implicit	-	x	-	-	-	k
GetThreadLocale	execution	implicit	-	x	-	-	-	k
RaiseException	exception-handling	implicit	-	x	-	-	-	k
timeKillEvent	-	implicit	-	x	-	-	-	w
timeSetEvent	-	implicit	-	x	-	-	-	w
DeleteCriticalSection	synchronization	implicit	-	-	-	-	-	k
LeaveCriticalSection	synchronization	implicit	-	-	-	-	-	k
EnterCriticalSection	synchronization	implicit	-	-	-	-	-	k
InitializeCriticalSection	synchronization	implicit	-	-	-	-	-	k
RegQueryValueExA	registry	implicit	-	-	-	-	-	a
RegOpenKeyExA	registry	implicit	-	-	-	-	-	a
RegCloseKey	registry	implicit	-	-	-	-	-	a
VirtualFree	memory	implicit	-	-	-	-	-	k
VirtualAlloc	memory	implicit	-	-	-	-	-	k
LocalFree	memory	implicit	-	-	-	-	x	k
LocalAlloc	memory	implicit	-	-	-	-	x	k
WriteFile	file	implicit	-	-	-	-	-	k
GetStartupInfoA	execution	implicit	-	-	-	-	-	k
GetCommandLineA	execution	implicit	-	-	-	-	-	k
ExitProcess	execution	implicit	-	-	-	-	-	k
UnhandledExceptionFilter	exception-handling	implicit	-	-	-	-	-	k
FreeLibrary	dynamic-library	implicit	-	-	-	-	-	k
GetStdHandle	console	implicit	-	-	-	-	-	k
GetVersion	-	implicit	-	-	-	-	x	k
WideCharToMultiByte	-	implicit	-	-	-	-	-	k
GetLocaleInfoA	-	implicit	-	-	-	-	x	k

Figure 4: The imports tab of PEStudio for sample2.exe

As stated earlier, this program imports 75 functions, 10 of which were blacklisted by PEStudio. Here are some of the most interesting:

- **FDICreate/FDICopy/FDIDestroy:** These are used for decompressing CAB archive files. Most likely, these functions are being used to install the “wotsuper” program.
- **GetKeyboardType:** If the program has keylogging capabilities, knowing the type of the keyboard being used may be used to guide keylogging.
- **SHGetFileInfo:** Retrieves information about a file or folder. If the malware is looking for files that match a particular pattern, this function may be useful to it.
- **TimeSetEvent/TimeKillEvent:** Can be used to schedule periodic or single-use timers. The malware may use timers to activate certain capabilities after a set amount of time, probably to make dynamic analysis more difficult. Similarly, timers could potentially be used to cause the program to exhibit different behavior if it suspects it is being debugged.
- **SetPixel/SetTextColor/MessageBoxA/etc.:** These functions appear to correspond to different GUI elements in a program. These imports are not inherently malicious, but they are more support towards the program using a GUI.

type (2)	size (bytes)	file-offset	blacklist (72)	hint (68)	group (19)	value (10235)
ascii	19	0x00028246	x	-	windowing	SetForegroundWindow
ascii	10	0x000283FC	x	-	windowing	GetCapture
ascii	11	0x00028426	x	-	windowing	EnumWindows
ascii	12	0x000286B7	x	-	windowing	GetClassLong
ascii	19	0x000287D7	x	-	storage	SetCurrentDirectory
ascii	14	0x00028B72	x	-	shell	SHChangeNotify
ascii	10	0x00028A9E	x	-	setup	FDIDestroy
ascii	7	0x00028AAC	x	-	setup	FDICopy
ascii	9	0x00028AB6	x	-	setup	FDICreate
ascii	15	0x00027CB0	x	-	security	OpenThreadToken
ascii	16	0x00027CC2	x	-	security	OpenProcessToken
ascii	7	0x00027CEC	x	-	security	FreeSid
ascii	8	0x00027CF6	x	-	security	EqualSid
ascii	24	0x00027D02	x	-	security	AllocateAndInitializeSid
ascii	21	0x00027D1E	x	-	security	AdjustTokenPrivileges
ascii	20	0x00028A3D	x	-	security	LookupPrivilegeValue
ascii	9	0x0002849A	x	-	resource	CopyImage
ascii	25	0x000287A3	x	-	registry	WritePrivateProfileString
ascii	13	0x000289D1	x	-	registry	RegSetValueEx
ascii	15	0x00027BB4	x	-	keyboard-and-mouse	GetKeyboardType
ascii	11	0x000283B0	x	-	keyboard-and-mouse	GetKeyState
ascii	13	0x000285AD	x	-	file	SHGetFileInfo
ascii	17	0x000287C1	x	-	file	SetFileAttributes
ascii	15	0x000287EF	x	-	file	RemoveDirectory
ascii	12	0x00028943	x	-	file	FindNextFile
ascii	13	0x00028953	x	-	file	FindFirstFile
ascii	10	0x00028981	x	-	file	DeleteFile
ascii	26	0x00028B2E	x	-	file	SHGetSpecialFolderLocation
ascii	19	0x00028B4D	x	-	file	SHGetPathFromIDList
ascii	17	0x00028B85	x	-	file	SHBrowseForFolder
ascii	13	0x0000E1A8	x	-	execution	EnumProcesses
ascii	18	0x0000E1B8	x	-	execution	EnumProcessModules
ascii	17	0x0000E1CD	x	-	execution	GetModuleBaseName
ascii	17	0x0000E1F9	x	-	execution	GetModuleBaseName
ascii	20	0x0000E224	x	-	execution	GetModuleInformation
ascii	15	0x0000E23C	x	-	execution	EmptyWorkingSet
ascii	27	0x0000E25C	x	-	execution	InitializeProcessForWsWatch
ascii	23	0x0000E28D	x	-	execution	GetDeviceDriverBaseName
ascii	23	0x0000E2D9	x	-	execution	GetDeviceDriverBaseName
ascii	20	0x0000E324	x	-	execution	GetProcessMemoryInfo
ascii	24	0x0000E564	x	-	execution	CreateToolhelp32Snapshot

Figure 5: The strings tab of PEStudio for sample2.exe

We can see that this malware contains 10,235 strings in total, 72 of which have been blacklisted. Some of the most interesting strings are:

- **GetKeyState:** This seems to be a reference to a Windows API function that could be used for keylogging.
- **WritePrivateProfileString:** This references a Windows API function that is used for writing .ini files. This could be a way for the malware to achive persistence.
- **FindFirstFile/FindNextFile:** These reference Windows API functions that suggest the malware might be navigating the file system, possibly to look for specific files, encrypt all files in a folder, or something similar.
- **WinExec:** A reference to a Windows API function that can be used to execute an executable file.
- **ShellExecute/ShellExecuteEx:** These strings reference Windows API functions that can be used to execute shell commands.
- **Deflate 1.1.4 Copyright 1995-2002 Jean-loup Gailly:** This appears to come from a C program called deflate.c (https://dev.w3.org/Amaya/libpng/zlib/deflate.c). This program is probably used somewhere in sample2.exe, although it is possible it is part of the Smart Install Maker tool. Note that the years referenced here make no sense if the program was truly compiled in 1992.

- **Inflate 1.1.4 Copyright 1995-2002 Mark Adler:** Like the previous string, this seems to come from another C program called inftrees.c (https://gnu.googlesource.com/gcc/+/refs/tags/releases/gcc-3.3.6/zlib/inftrees.c).
- **regedit.exe:** This is a reference to an executable that can be used to edit registry keys.
- **http://www.wotsuper.com/:** This seems to be a domain name that is used in some way by the wotsuper program that sample2.exe will install. This could be used as a C2 server, or some other communication with the malware author.
- **https://iplogger.org/1Ldta7.html *and* https://iplogger.org/1smEq7.html:** These appear to be links that may allow for an IP author to track the IP address of the infected machine. In the dynamic analysis section, we will want to look for any communication to these domain names.
- **@$&%04\wotsuper.exe, @$&%04\wotsuper1.exe, *and more strings in this form:*** These strings appear to reference new executable files called "wotsuper.exe" and "wotsuper1.exe". These could refer to new executable files that are created by sample2.exe during the installation process.
- **cabinet.dll:** References a DLL that is used for managing CAB files.
- **disk@$&%19.pak:** This appears to reference a .pak file, which is a compressed file like a .zip file. Because of the similarity in structure to the earlier wotsuper.exe strings, these executable files are likely stored in this file before being installed using Smart Install Maker.
- **wotsuper@company.com:** This string appears to be a fake email address. Perhaps this can be used to trace some sort of account that was used to access the Smart Install Maker program.
- **\s @$&%02\wotsuper.reg:** This seems to be referencing a new registry key called "wotsuper". In the dynamic analysis, we should keep a look out for the creation of this key.
- **Smart Install Maker v 5.04:** This is another indication that this program was created using Smart Install Maker, specifically v 5.04.
- **:\Program Files:** This indicates that the malware may be accessing the Program Files folder, which makes sense if it is installing a program.
- **Russian:** This may be some indication that the malware was originally written by someone who speaks Russian.

As can be seen in *Figure 1*, the hashes for sample2.exe are dd1b734796b4aa40af46b4d69e1e2da2 (MD5), d5273be84dfa0c54fc9cefff7bcc24fed3e20e1c (SHA1), and 361411e6321c45c845669ac89e32feec0bdd97916b5d73f508c43576b8a15a20 (SHA256).

In VirusTotal (not pictured), we can see that 57 out of 73 vendors flagged sample2.exe as malicious, so this sample is known to the security community.

Section 2B: Decompiling sample2.exe

In this section, we will use Ghidra to decompile sample2.exe and get a better look at its functionality. There are many functions that can be found in this file, so for this report, we will focus on just some of the most interesting pieces of code.

```
2  void entry(void)
3
4  {
5    undefined4 uVar1;
6    undefined4 extraout_ECX;
7    undefined4 extraout_ECX_00;
8
9    setup_program();
10   uVar1 = setup_menu();
11   *(undefined4 *)PTR_DAT_00428840 = uVar1;
12   install_programs(extraout_ECX);
13   call_installation_functions(extraout_ECX_00);
14   exchange_messages();
15                     /* WARNING: Subroutine does not return */
16   close_program();
17 }
```

Figure 6: The entry function of sample2.exe

In the above screenshot, we can see the top level of the sample2.exe program (prior to unpacking). Functions called in this screenshot are labeled with informed assumptions about the functionality of each. Ghidra identified hundreds of different functions in this file, which is far too many to analyze manually within a reasonable amount of time. To limit the scope of our analysis, we will first examine the FDI functions identified by PEStudio, as they appear to be related to the packing technique used.

```
2  void FUN_0040e64c(void)
3
4  {
5    int in_EAX;
6    undefined4 uVar1;
7
8    if (*(int *)(in_EAX + 4) == 0) {
9      FUN_00402a94(0,0xc);
10     uVar1 = FDICreate(&LAB_0040e518,&LAB_0040e528,&LAB_0040e538,FUN_0040e54c,FUN_0040e560,
11                       FUN_0040e574,&LAB_0040e584,0xffffffff,in_EAX + 0x3c);
12     *(undefined4 *)(in_EAX + 4) = uVar1;
13   }
14   return;
15 }
```

Figure 7: A call to FDICreate in sample2.exe

The FDICreate function is called in the function shown above. This is used to initialize an FDI context, which is used for extracting CAB files.

```
FUN_0040ea00();
FUN_004036e8(extraout_ECX,local_c);
FUN_00405c64(extraout_ECX_00,&local_10);
FUN_004036e8(extraout_ECX_01,local_10);
FUN_0040c47c(extraout_ECX_02);
FUN_0040c47c(extraout_ECX_03);
FUN_0040372c(extraout_ECX_04,*(int *)(in_EAX + 8));
while (puVar1 = puStack_28, local_8 != (undefined4 *)0x0) {
  FUN_00403694();
  FUN_00403694();
  FUN_004036e8(extraout_ECX_05,local_8);
  FUN_00403694();
  *(undefined1 *)(in_EAX + 0x38) = 0;
  FUN_00405c8c();
  puVar2 = check_eax_nonzero();
  FUN_0040d95c();
  puVar3 = check_eax_nonzero();
  uVar6 = FUN_0040e634();
  FDICopy((int)uVar6,puVar3,puVar2);
```

Figure 8: A segment of a function (at 0x40ea3c) that calls FDICopy

A snippet of the code around where FDICopy was called is shown in the above screenshot. This function is related to the actual extraction of files from the CAB archive. The FDIDestroy function is also called in the function at address 0x40e6a4, and this function destroys the FDI context. (We will see in the dynamic analysis section that, by breaking on calls to these functions, we can find the actual files compressed in the CAB archives.)

property	value	value	value	value	value
name	CODE	DATA	BSS	.idata	.tls
md5	5E14E4EDE2E2215BC7D7283...	281C1AC1D4F602BCC76D47...	C01C905CB1D57AE03DF4FF...	8AEE97F029EB584241433B60...	n/a
entropy	6.594	3.796	0.972	5.282	n/a
file-ratio (99.47%)	77.39 %	5.59 %	2.39 %	3.19 %	n/a
raw-address	0x00000400	0x00024A00	0x00027400	0x00028600	0x00029E00
raw-size (191488 bytes)	0x00024600 (148992 bytes)	0x00002A00 (10752 bytes)	0x00001200 (4608 bytes)	0x00001800 (6144 bytes)	0x00000000 (0 bytes)
virtual-address	0x00401000	0x00426000	0x00429000	0x0042B000	0x0042D000
virtual-size (212992 bytes)	0x00025000 (151552 bytes)	0x00003000 (12288 bytes)	0x00002000 (8192 bytes)	0x00002000 (8192 bytes)	0x00001000 (4096 bytes)
entry-point	0x0000A7B8	-	-	-	-
characteristics	0x60000020	0xC0000040	0xC0000000	0xC0000040	0xC0000000
writable	-	x	x	x	x
executable	x	-	-	-	-

Figure 9: The sections tab of sample2.exe in PEStudio after unpacking

Now that we have identified function calls related to the unpacking process, we will analyze sample2.exe after it has been unpacked. (We will discuss the process of unpacking sample2.exe in the dynamic analysis section.) After unpacking sample2.exe, we observe little difference in the structure of the file compared to the packed file. One notable point is that the file ratio is nearly 100%, as shown in the above screenshot. As a result, the file now has normal entropy (6.463). This implies that the Smart Install Maker overlay is not part of the unpacked file.

```
signature: CAB, location: CODE, offset: 0x0001E3A0, size: -1
signature: Smart-installer, location: CODE, offset: 0x00021ED0, size: -1
signature: CAB, location: CODE, offset: 0x000245DC, size: -1
```

Figure 10: The identification of file type references in unpacked sample2.exe

We can see from the above figure that the file still contains references to two CAB files, and a file with the Smart-installer signature, but they are listed as having size -1, indicating they have become inaccessible in the unpacking process.

When analyzing the code of the unpacked file in Ghidra, we can see that it is not very different at all from the code Ghidra identified in the packed sample2.exe. We will highlight a few of the interesting functions that can be found in this file.

```
DVar3 = GetFileAttributesA(pCVar2);
if ((DVar3 == 0xffffffff) || ((DVar3 & 0x10) != 0)) {
  FUN_00405c8c();
  FUN_00405ce4();
}
else {
  pvVar6 = (HANDLE)0x0;
  dwFlagsAndAttributes = 0x80;
  dwCreationDisposition = 3;
  lpSecurityAttributes = (LPSECURITY_ATTRIBUTES)0x0;
  dwShareMode = 0;
  dwDesiredAccess = 0x80000000;
  pCVar2 = FUN_00403af4();
  pvVar6 = CreateFileA(pCVar2,dwDesiredAccess,dwShareMode,lpSecurityAttributes,
                       dwCreationDisposition,dwFlagsAndAttributes,pvVar6);
  GetFileTime(pvVar6,(LPFILETIME)0x0,(LPFILETIME)0x0,&local_18);
  CloseHandle(pvVar6);
```

Figure 11: Part of a function in sample2.exe that is used for creating new files

The function shown above is used to create new files. We will see in the dynamic analysis section that this program creates several files, and most of them are created here.

```
FUN_0040b86c(uVar20,*(undefined4 **)(*(int *)PTR_DAT_004287f8 + 0x94));
FUN_00403940(*(undefined4 **)(*(int *)PTR_DAT_00428864 + 0xdc),
             (undefined4 *)"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Uninstall\\");
puVar6 = *(undefined4 **)(*(int *)PTR_DAT_00428864 + 0xdc);
FUN_041f8d8((undefined4 *)"DisplayName",unaff_EBX,'\x01','\x01',0,puVar6);
puVar26 = (undefined4 *)0x0;
puVar25 = (undefined4 *)0x1;
FUN_041f8d8((undefined4 *)"DisplayVersion",unaff_EBX,'\x01','\x01',0,
             *(undefined4 **)(*(int *)PTR_DAT_00428864 + 0x8c));
puVar11 = *(undefined4 **)(*(int *)PTR_DAT_00428864 + 0x15c);
puVar24 = (undefined4 *)0x2;
iVar23 = 1;
FUN_041f8d8((undefined4 *)"VersionMajor",unaff_EBX,'\x01','\x01',2,puVar11);
iVar22 = 2;
iVar8 = 1;
uVar20 = 1;
FUN_041f8d8((undefined4 *)"VersionMinor",unaff_EBX,'\x01','\x01',2,
             *(undefined4 **)(*(int *)PTR_DAT_00428864 + 0x1dc));
puVar19 = *(undefined4 **)(*(int *)PTR_DAT_00428864 + 0xa8);
FUN_041f8d8((undefined4 *)"Publisher",unaff_EBX,'\x01','\x01',0,puVar19);
puVar18 = *(undefined4 **)(*(int *)PTR_DAT_00428864 + 0x16c);
iVar17 = 0;
uVar16 = 1;
FUN_041f8d8((undefined4 *)"DisplayIcon",unaff_EBX,'\x01','\x01',0,puVar18);
puVar15 = *(undefined4 **)(*(int *)PTR_DAT_00428864 + 0x16c);
puVar14 = (undefined4 *)0x0;
```

Figure 12: A function in sample2.exe that appears to set up registry keys

The function in this screenshot clearly references the registry key “SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall”. We will see in the dynamic

analysis section that the malware creates and modifies many registry keys, including values with the names referenced here (e.g. “DisplayName”, “DisplayVersion”, etc.).

```
    BVar5 = ShellExecuteExA(&local_54);
    if (BVar5 != 0) {
      while (DVar6 = WaitForSingleObject(local_54.hProcess,0x32), DVar6 == 0x102) {
        FUN_0040bd64();
      }
      CloseHandle(local_54.hProcess);
    }
  }
  else {
    nShowCmd = local_c;
    lpDirectory = check_eax_nonzero();
    lpParameters = check_eax_nonzero();
    lpFile = check_eax_nonzero();
    ShellExecuteA(*(HWND *)(*(int *)PTR_DAT_00428840 + 0x1c),"open",lpFile,lpParameters,
                  lpDirectory,nShowCmd);
  }
  FUN_0041d648(0x2800,0);
 }
 iVar2 = iVar2 + -1;
} while (iVar2 != 0);
```

Figure 13: Part of a function is sample2.exe that is used for executing shell commands

This code segment can be used for executing shell commands. As we will see in the dynamic analysis section, these commands are used to start five new processes.

Section 2C: Basic static analysis of wotsuper.exe

When we run the program in the dynamic analysis section of the report, we will see that sample2.exe creates three executable files wotsuper.exe, wotsuper1.exe, and Uninstall.exe in C:\Program Files\wotsuper\wotsuper. In this section, we will statically analyze the first of these executable files.

property	value
md5	7B20F5C61780FE383F45CA6E18ED5A6A
sha1	BC9BFD59F0CDE312CD9A0D20784887FED9B8C836
sha256	26CCBCB079B3F0CC183293351C40DA3146D2DDEC9B4D6CD314090CFAB94834DF
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z . . . . . @ . . . . .
file-size	460288 (bytes)
size-without-overlay	n/a
entropy	7.622
imphash	FD6E2D905392A4911591F53458632D22
signature	Microsoft Visual C++ 8
entry-point	E8 E5 3B 00 00 E9 78 FE FF FF CC CC CC CC CC CC CC 8B 4C 24 04 F7 C1 03 00 00 00 74 24 8A 01 83 C1
file-version	n/a
description	n/a
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x5D2AEA8D (Sun Jul 14 04:40:45 2019)
debugger-stamp	n/a
resources-stamp	
exports-stamp	n/a
version-stamp	empty
certificate-stamp	n/a

Figure 14: The output on PEStudio for wotsuper.exe

We can see that this program is a standard **executable** file, compiled on **July 14, 2019** for **32-bit** architectures. Its signature "**Microsoft Visual C++ 8**", indicating this program was written in C++ originally. We can see that the entropy of the file is 7.622, which is quite high, indicating this file is using some sort of packing or obfuscation.

property	value	value	value	value
name	.text	.rdata	.data	.rsrc
md5	5163DA13228C7AD14E2717B...	5D13526130BDA22887390DB...	8BFD933F01274E5AD81F730...	2CAA09347095C7FCA57057...
entropy	6.592	5.417	7.843	6.055
file-ratio (99.78%)	9.79 %	2.34 %	74.86 %	12.79 %
raw-address	0x00000400	0x0000B400	0x0000DE00	0x00062000
raw-size (459264 bytes)	0x0000B000 (45056 bytes)	0x00002A00 (10752 bytes)	0x00054200 (344576 bytes)	0x0000E600 (58880 bytes)
virtual-address	0x00401000	0x0040C000	0x0040F000	0x032CC000
virtual-size (49119964 byt...	0x0000AF60 (44896 bytes)	0x000028F0 (10480 bytes)	0x02EBC544 (49005892 bytes)	0x0000E548 (58696 bytes)
entry-point	0x00001CBF	-	-	-
characteristics	0x60000020	0x40000040	0xC0000040	0x40000040
writable	-	-	x	-
executable	x	-	-	-

Figure 15: The section tab on PEStudio for wotsuper.exe

Examining the sections in wotsuper.exe reveals why the entropy is so high. This file contains four sections: .text, .rdata, .data, and .rsrc. Three of the four sections have normal entropy and raw-to-virtual size ratios. However, the .data section has very high entropy (7.843), and its virtual size is over 140 times larger than its raw size. Moreover, this section comprises nearly 75% of the data in the file. There is clearly some sort of important information being stored in .data. (We will see later that this section contains an encrypted shellcode.)

name (91)	group (9)	type (1)	ordinal (0)	blacklist (11)	anti-debug (0)	undocumented (0)	deprecated (10)	library (2)
AdjustTokenPrivileges	security	implicit	-	x	-	-	-	advapi32.dll
ReadProcessMemory	memory	implicit	-	x	-	-	-	kernel32.dll
VirtualProtect	memory	implicit	-	x	-	-	-	kernel32.dll
SetFileShortNameW	file	implicit	-	x	-	-	-	kernel32.dll
DeleteFileA	file	implicit	-	x	-	-	-	kernel32.dll
GetEnvironmentStringsW	execution	implicit	-	x	-	-	-	kernel32.dll
TerminateProcess	execution	implicit	-	x	-	-	-	kernel32.dll
GetCurrentThreadId	execution	implicit	-	x	-	-	-	kernel32.dll
GetEnvironmentStrings	execution	implicit	-	x	-	-	-	kernel32.dll
GetCurrentProcessId	execution	implicit	-	x	-	-	-	kernel32.dll
GetModuleFileNameA	dynamic-library	implicit	-	x	-	-	-	kernel32.dll
IsDebuggerPresent	system-information	implicit	-	-	-	-	-	kernel32.dll
QueryPerformanceCount...	system-information	implicit	-	-	-	-	-	kernel32.dll
GetTickCount	system-information	implicit	-	-	-	-	-	kernel32.dll
InterlockedDecrement	synchronization	implicit	-	-	-	-	-	kernel32.dll
ResetEvent	synchronization	implicit	-	-	-	-	-	kernel32.dll
CreateMutexW	synchronization	implicit	-	-	-	-	-	kernel32.dll
InterlockedIncrement	synchronization	implicit	-	-	-	-	-	kernel32.dll
EnterCriticalSection	synchronization	implicit	-	-	-	-	-	kernel32.dll
LeaveCriticalSection	synchronization	implicit	-	-	-	-	-	kernel32.dll
DeleteCriticalSection	synchronization	implicit	-	-	-	-	-	kernel32.dll
InitializeCriticalSectionA...	synchronization	implicit	-	-	-	-	-	kernel32.dll
GlobalAlloc	memory	implicit	-	-	-	-	x	kernel32.dll
LocalAlloc	memory	implicit	-	-	-	-	x	kernel32.dll
HeapAlloc	memory	implicit	-	-	-	-	-	kernel32.dll

Figure 16: The imports tab for wotsuper.exe in PEStudio

As can be seen in the screenshot above, this program imports 91 functions, 11 of which were blacklisted by PEStudio. Here are some of the most notable imports:

- **VirtualProtect:** This import could be used to make a segment of memory executable for loading a shellcode or other process. (We will see in the dynamic analysis section that this is, indeed, happening.)
- **DeleteFileA:** This allows for a file to be deleted, which could be used by the malware to cover its tracks, or to delete data from the infected machine.
- **IsDebuggerPresent:** This import can be used to check whether the program is being debugged, so the malware may be designed to behave differently when being debugged compared to when it is run normally.

type (2)	size (bytes)	file-offset	blacklist (13)	hint (11)	value (4555)
ascii	357	0x0006988B	-	-	3HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
unicode	323	0x0006FBE8	-	-	Tuzuvuha viyixoxuguwebe lediNucij jedatajafize vinijizojicayi mugakamopume hufumexubo puv maguweg
ascii	290	0x00063096	-	-	99999999999999999999999999999999999999999999999999999999999999999999999999999999999999
unicode	225	0x0007044A	-	-	=Favigomad bifavuve yolic nesutixikusane welokusosay pewebizaw^Wohicozelacey yewinucap cugupinex
unicode	218	0x0006F9C9	-	-	Micik seki madunebe gixekucaw8Zitawanujewoj cowuberiwehuku hirewulijip fubu laco bipelQJerohihofuje
unicode	204	0x00070119	-	-	Kazuverozacog wulewexop wexaxe?Vikedafi jiral toyu nayuxuya gigojebodoyap yewitaruwuko yosabepBKar
unicode	203	0x0006FF76	-	-	Tedoxocid!Bakico yubaseh zicoyid lowanunehelKumabifuleyotaz zuyaranefatuc rigurivajemixol mukorowu
unicode	184	0x0006FDF1	-	-	0Jacesesulateco yoseyu zikefenefu zadatenohokejodaWicixulumato jahitogug jumi fukunuvor kisumuci ka:
ascii	161	0x00063ABF	-	-	AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ascii	161	0x0000B825	-	-	This indicates a bug in your application. It is most likely the result of calling an MSIL-compiled (/clr) functi
unicode	140	0x0006F73D	-	-	KLarora maxujuyuh povujuriniteye yabigaca tosepuwawe bayihif rejesup tezupow?Lusibig lamalodalopag v
unicode	104	0x0007024F	-	-	Gariyojamiley fiyiber tiw nadosHBiregikew ruputose soxewumavo bobogejucuj deday xipu jurogebud mak
unicode	104	0x0006F869	-	-	Vizigugutizut fadesu mutoreLTayaba zezoza jic pizeriyosi degan daborobiyo zolip wubazewufalino repapatu
ascii	95	0x0000CCD8	-	-	!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`ABCDEFGHIJKLMNOPQRSTU
ascii	95	0x0000BD57	-	-	!"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxy
ascii	95	0x0000CB58	-	-	!"#$%&'()*+,-./0123456789:;<=>?@abcdefghijklmnopqrstuvwxyz[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
ascii	88	0x0000B7CE	-	-	R6033\r\n- Attempt to use MSIL code from this assembly during native code initialization
ascii	86	0x0000B72E	-	-	R6034\r\nAn application has made an attempt to load the C runtime library incorrectly.
unicode	86	0x0007030F	-	-	Sajuva putoce vexirelol>Tuvarahito doyibe puye facositomef jadaheb cezuzo gojabifowija
ascii	77	0x0000BB2A	-	-	This application has requested the Runtime to terminate it in an unusual way.
ascii	71	0x0000B783	-	-	Please contact the application's support team for more information.\r\n
ascii	71	0x0000BB76	-	-	Please contact the application's support team for more information.\r\n
ascii	65	0x000628B8	-	-	99999999999999999999999999999999999999999999999999999999999999!
ascii	56	0x0000BA40	-	-	R6024\r\n- not enough space for _onexit/atexit table\r\n
ascii	56	0x0000B9E0	-	-	R6026\r\n- not enough space for stdio initialization\r\n
ascii	56	0x0000B9A8	-	-	R6027\r\n- not enough space for lowio initialization\r\n

Figure 17: The strings tab in PEStudio for wotsuper.exe

The strings tab shows 4555 strings in wotsuper.exe, with 13 of those blacklisted by PEStudio. Interestingly, despite the large number of strings in the program, none of them seem particularly notable or interesting. This is likely because this file has very high entropy, and so most of the file will have a very random structure. There are, however, several large strings in the file (shown in the above screenshot) that could correspond to encrypted messages, possibly mapping to strings written in a language other than English. Until the file is analyzed more thoroughly, there is no way to know for certain.

property	value
md5	E6ED30A657E08C600054C4A45C5D22A5
sha1	9227E10D7A2E85AE0CFB613A2C42D682D3012FE1
sha256	7C8611B70DE70D264C28022ADED4F00FAB134E175BFB6E236F6D972405721895
file-type	**static-link library**
date	empty
language	n/a
code-page	n/a
InternalSurname	dhrj.uxe
Copyright	Copyrighd (C) 2020, odfrjv
ProductionVersion	1.0.4.8

Figure 18: Version information for wotsuper.exe from PESTudio

In the version tab of PEStudio, we can see the file type has been labeled as a **static-link library**. The internal surname is “dhrj.uxe”, which doesn’t appear particularly meaningful on its own (but looking this string up does yield several results discussing malware samples, such as in https://any.run/report/6f2432e4ac98575aa653094123b478bf6c3aa9b00aaad84dfd09045a96d6b970/099b0717-5e3a-4e40-8bcc-4639a38fae3b). The copyright string “Copyrighd © 2020, odfrjv” also appears to be unusual, since it is spelled incorrectly, and is trailed with seemingly random characters. Perhaps some of the meaningful data that was there has been overwritten by some sort of obfuscation or encryption method. The production version is labeled given as 1.0.4.8.

non-standard	size (57508 bytes)	file-ratio (12.49%)	md5	entropy	language (2)	first-bytes-hex
-	428	0.09 %	E6ED30A657E08C600054C4A45C5D22A5	3.345	neutral	AC 01 34 00 00 00 56 00 53 00 5F 00 56 ...
-	1668	0.36 %	4B2170DFB3ADBFA62F2ABF51872D0CB2	3.311	neutral	4B 00 4C 00 61 00 72 00 6F 00 72 00 61 ...
-	1580	0.34 %	F86F9C0740854BF4B6072D80E675CD70	3.316	neutral	30 00 4A 00 61 00 63 00 65 00 73 00 65 ...
-	476	0.10 %	9D989FB0A883EC760E704C11E3D8A06C	3.114	neutral	3D 00 46 00 61 00 76 00 69 00 67 00 6F ...
-	118	0.03 %	160A4674D1A4048D80B3617538C5C764	2.858	Kinyarwanda	00 00 01 00 08 00 30 30 00 00 01 00 08 ...
-	118	0.03 %	DEEB35E325F9E67400CD54A41F81565B	2.917	Kinyarwanda	00 00 01 00 08 00 30 30 00 00 01 00 08 ...
-	3752	0.82 %	4ACE2332807B8C20A366553989184869	4.677	Kinyarwanda	28 00 00 00 30 00 00 00 60 00 00 00 01 ...
-	2216	0.48 %	4206A77D0B471450102C71C3928A3DAA	4.768	Kinyarwanda	28 00 00 00 20 00 00 00 40 00 00 00 01 ...
-	1736	0.38 %	0156432DD61FF5EAC6FD2A5DCD28477F	4.729	Kinyarwanda	28 00 00 00 18 00 00 00 30 00 00 00 01 ...
-	1384	0.30 %	3D0BB24398CB49CCB0DCA03F5CE21D6A	4.155	Kinyarwanda	28 00 00 00 10 00 00 00 20 00 00 00 01 ...
-	9640	2.09 %	A98546A5C7E6247F24EF1998DC6637BB	5.259	Kinyarwanda	28 00 00 00 30 00 00 00 60 00 00 00 01 ...
-	4264	0.93 %	4E60935A03B321F81EBAD0D6A4F91F0D	5.837	Kinyarwanda	28 00 00 00 20 00 00 00 40 00 00 00 01 ...
-	2440	0.53 %	D2C432669FCBF3DFFE38683A1DD64D48	5.927	Kinyarwanda	28 00 00 00 18 00 00 00 30 00 00 00 01 ...
-	1128	0.25 %	F700E24FC4A6A0807B165DA4C81A25F0	5.999	Kinyarwanda	28 00 00 00 10 00 00 00 20 00 00 00 01 ...
-	3752	0.82 %	EE1E9AF9BC34891498685552612 29C2E	4.184	Kinyarwanda	28 00 00 00 30 00 00 00 60 00 00 00 01 ...
-	2216	0.48 %	5F4CE847EE96744DCC978230E016647F	4.213	Kinyarwanda	28 00 00 00 20 00 00 00 40 00 00 00 01 ...
-	1736	0.38 %	F1302F3F98DEDA9FA972E09495174855	4.274	Kinyarwanda	28 00 00 00 18 00 00 00 30 00 00 00 01 ...
-	1384	0.30 %	5ADF8BEEF72EDF55F82B279A42923CAF	4.053	Kinyarwanda	28 00 00 00 10 00 00 00 20 00 00 00 01 ...
-	9640	2.09 %	019544B92A3F4985AA2A7625302BDBAD	6.100	Kinyarwanda	28 00 00 00 30 00 00 00 60 00 00 00 01 ...
-	4264	0.93 %	C9F4B7EB91BDCFE05FFE86D0A5769771	6.528	Kinyarwanda	28 00 00 00 20 00 00 00 40 00 00 00 01 ...
-	2440	0.53 %	0E2DA3CDCAC9A38CFB1331449A5C6C...	6.620	Kinyarwanda	28 00 00 00 18 00 00 00 30 00 00 00 01 ...
-	1128	0.25 %	6FD5FFF536A54BF594BC0E2B4EFC74E2	6.432	Kinyarwanda	28 00 00 00 10 00 00 00 20 00 00 00 01 ...

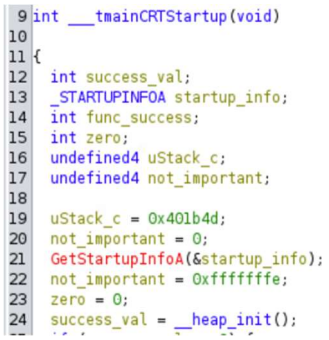
Figure 19: The resources tab in PEStudio for wotsuper.exe

In wotsuper.exe, we can also see that many of the resources use text that is written in Kinyarwanda. This could hint to the malware authors being from Rwanda, or perhaps it could imply the people in Rwanda are the intended targets for this malware. It may also be the case that this language was chosen to make analysis more difficult since it is not a commonly spoken language.

As we can see in the first screenshot of this section, the hashes for this file are 7b20f5c61780fe383f45ca6e18ed5a6a (MD5), bc9bfd59f0cde312cd9a0d20784887fed9b8c836 (SHA1), and 26ccbcb079b3f0cc183293351c40da3146d2ddec9b4d6cd314090cfab94834df (SHA256).

In VirusTotal (not pictured), we can see that 55/71 vendors recognized wotsuper.exe as malicious. This is a comparable amount to those that flagged sample2.exe as malicious, which is not surprising.

Section 2D: Decompiling wotsuper.exe



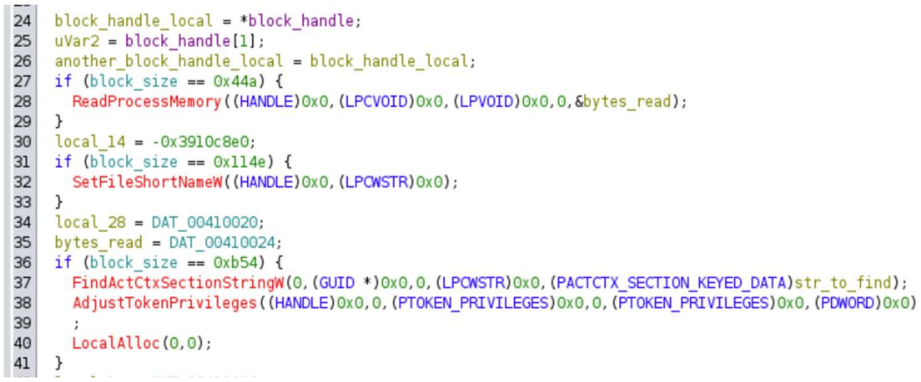
```
int __tmainCRTStartup(void)

{
  int success_val;
  _STARTUPINFOA startup_info;
  int func_success;
  int zero;
  undefined4 uStack_c;
  undefined4 not_important;

  uStack_c = 0x401b4d;
  not_important = 0;
  GetStartupInfoA(&startup_info);
  not_important = 0xfffffffe;
  zero = 0;
  success_val = __heap_init();
```

Figure 20: A snippet of code from the main function of wotsuper.exe

Since wotsuper.exe appears to be packed, the decompilation of this file does not reveal much interesting about this program's behavior.



```
block_handle_local = *block_handle;
uVar2 = block_handle[1];
another_block_handle_local = block_handle_local;
if (block_size == 0x44a) {
  ReadProcessMemory((HANDLE)0x0,(LPCVOID)0x0,(LPVOID)0x0,0,&bytes_read);
}
local_14 = -0x3910c8e0;
if (block_size == 0x114e) {
  SetFileShortNameW((HANDLE)0x0,(LPCWSTR)0x0);
}
local_28 = DAT_00410020;
bytes_read = DAT_00410024;
if (block_size == 0xb54) {
  FindActCtxSectionStringW(0,(GUID *)0x0,0,(LPCWSTR)0x0,(PACTCTX_SECTION_KEYED_DATA)str_to_find);
  AdjustTokenPrivileges((HANDLE)0x0,0,(PTOKEN_PRIVILEGES)0x0,0,(PTOKEN_PRIVILEGES)0x0,(PDWORD)0x0)
  ;
  LocalAlloc(0,0);
}
```

Figure 21: Code used to read the memory of the wotsuper.exe process

However, there is one function at address 0x401010 that is highly suspicious. In the screenshot above, we can see code that is used to read the memory of the current process.

```
do {
  if (block_size_local == 0x4c3) {
    ResetEvent((HANDLE)0x0);
    block_size_local = block_size;
  }
  uVar1 = block_handle_local * 0x10 + local_1c;
  if (block_size_local == 0xfa9) {
    _DAT_032ae9e4 = 0xedeb2e40;
  }
  local_24 = local_14 + block_handle_local;
  block_handle_local = (block_handle_local >> 5) + local_20;
  _DAT_032ae9e0 = 0x9150ce2e;
  if (block_size_local == 0x912) {
    DeleteFileA((LPCSTR)0x0);
    _DAT_004724ec = 0;
  }
  uVar2 = uVar2 - (block_handle_local ^ local_24 ^ uVar1);
  local_10 = 4;
  another_block_handle_local =
       another_block_handle_local -
       (uVar2 * 0x10 + local_28 ^ (uVar2 >> 5) + bytes_read ^ local_14 + uVar2);
  _DAT_004724e8 = 0;
  if (block_size == 0x1d5) {
    GetConsoleCP();
  }
  add_to_eax();
  loop_count = loop_count + -1;
  block_size_local = extraout_ECX;
  block_handle_local = another_block_handle_local;
} while (loop_count != 0);
```

Figure 22: A decryption routine found in wotsuper.exe

Further down in the same function, we can find a loop that involves performing multiple XOR and bit shift operations. This appears to be decrypting the information that was read from memory in the previous screenshot (and we will see later that this information came from the encrypted .data section of this file). The algorithm used appears to be similar to the Tiny Encryption Algorithm (https://en.wikipedia.org/wiki/Tiny_Encryption_Algorithm). We can see the numbers 0xedb2e40 and 0x9150ce2e set to variables in the above screenshots. These are probably magic numbers used by the decryption routine, but searching for them online doesn't seem to yield any meaningful results. To analyze the decrypted information, we will need to run the program; therefore, we will defer that analysis until the dynamic analysis section.

Section 2E: Basic static analysis of wotsuper1.exe

In this section, we will perform basic static analysis on the wotsuper1.exe file, which appears to contain the main functionality of the program.

property	value
md5	B8181CB72764C24E73C7B6204B16BED6
sha1	C430CC4776FF5E21D08BCA9A0D73CFAF29108FA4
sha256	FDB5A0D4E97EE36D2B23605B0D8A2785D08D046058F07A8714E4908E8A2485A2
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z . . . . . @ . . . . .
file-size	557568 (bytes)
size-without-overlay	n/a
entropy	6.686
imphash	50AC3D5CF691B8BCE399538F4883F0AD
signature	Microsoft Visual C++ 8
entry-point	E8 0A AF 00 00 E9 89 FE FF FF 8B FF 55 8B EC 8B 45 08 85 C0 74 12 83 E8 08 81 38 DD DD 00 00 75 07
file-version	n/a
description	n/a
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x5F05B478 (Wed Jul 08 07:56:40 2020)
debugger-stamp	n/a
resources-stamp	
exports-stamp	n/a
version-stamp	n/a
certificate-stamp	n/a

Figure 23: The output of PEStudio for wotsuper1.exe

Similar to wotsuper.exe, this program is an **executable** that appears to have been compiled from **C++** for **32-bit** systems on July 8, 2020. It also has GUI capabilities. The entropy of this file is 6.686, which is standard for an executable program.

property	value	value	value
name	.text	.rdata	.data
md5	E4E87EA1B52537282C0970FE...	5550E35F6033C3CEFA0BB9A...	67CD574740C7CEB8EB0C58...
entropy	6.685	5.743	3.660
file-ratio (99.82%)	83.38 %	14.60 %	1.84 %
raw-address	0x00000400	0x00071C00	0x00085A00
raw-size (556544 bytes)	0x00071800 (464896 bytes)	0x00013E00 (81408 bytes)	0x00002800 (10240 bytes)
virtual-address	0x00401000	0x00473000	0x00487000
virtual-size (566597 bytes)	0x000716FD (464637 bytes)	0x00013D44 (81220 bytes)	0x00005104 (20740 bytes)
entry-point	0x0005BC0D	-	-
characteristics	0x60000020	0x40000040	0xC0000040
writable	-	-	x
executable	x	-	-
shareable	-	-	-
discardable	-	-	-

Figure 24: The sections tab of PEStudio for wotsuper1.exe

This program contains only three sections: .text, .rdata, and .data. The entropy and raw to virtual file size ratios for each section appears normal. We will also see that this program has a large amount of imports and meaningful strings, indicating this program is not using any sort of packing or sophisticated obfuscation.

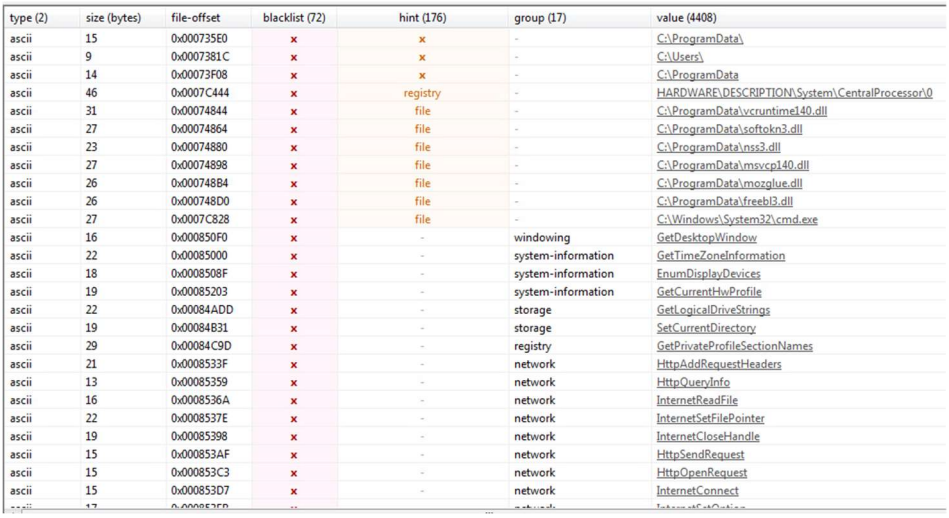
name (201)	group (14)	type (1)	ordinal (0)	blacklist (56)	anti-debug (0)	undocumented (0)	deprecated (16)	library (12)
GetDesktopWindow	windowing	implicit	-	x	-	-	-	user32.dll
GetTimeZoneInformation	system-information	implicit	-	x	-	-	-	kernel32.dll
EnumDisplayDevicesW	system-information	implicit	-	x	-	-	-	user32.dll
GetCurrentHwProfileA	system-information	implicit	-	x	-	-	-	advapi32.dll
SetCurrentDirectoryA	storage	implicit	-	x	-	-	-	kernel32.dll
GetLogicalDriveStringsA	storage	implicit	-	x	-	-	-	kernel32.dll
GetPrivateProfileSection...	registry	implicit	-	x	-	-	x	kernel32.dll
InternetConnectA	network	implicit	-	x	-	-	-	wininet.dll
HttpOpenRequestA	network	implicit	-	x	-	-	-	wininet.dll
HttpSendRequestA	network	implicit	-	x	-	-	-	wininet.dll
InternetCloseHandle	network	implicit	-	x	-	-	-	wininet.dll
InternetSetFilePointer	network	implicit	-	x	-	-	-	wininet.dll
InternetReadFile	network	implicit	-	x	-	-	-	wininet.dll
HttpQueryInfoA	network	implicit	-	x	-	-	-	wininet.dll
HttpAddRequestHeadersA	network	implicit	-	x	-	-	-	wininet.dll
InternetSetOptionA	network	implicit	-	x	-	-	-	wininet.dll
InternetOpenA	network	implicit	-	x	-	-	-	wininet.dll
InternetOpenUrlA	network	implicit	-	x	-	-	-	wininet.dll
GlobalMemoryStatus	memory	implicit	-	x	-	-	x	kernel32.dll
HeapSetInformation	memory	implicit	-	x	-	-	-	kernel32.dll
UnlockFile	file	implicit	-	x	-	-	-	kernel32.dll
LockFile	file	implicit	-	x	-	-	-	kernel32.dll
LockFileEx	file	implicit	-	x	-	-	-	kernel32.dll
UnlockFileEx	file	implicit	-	x	-	-	-	kernel32.dll
UnmapViewOfFile	file	implicit	-	x	-	-	-	kernel32.dll
MapViewOfFile	file	implicit	-	x	-	-	-	kernel32.dll
CreateFileMappingA	file	implicit	-	x	-	-	-	kernel32.dll

Figure 25: The imports tab of PEStudio for wotsuper1.exe

PEStudio identified 201 imports in the program, 56 (over 25%) of which were blacklisted by PEStudio. There are many interesting imports to discuss among these, but here are just a few of the interesting ones:

- **InternetConnectA, HttpOpenRequestA, etc.:** Many imports similar to these can be found in the file, indicating that wotsuper1.exe will be attempting to communicate over the internet.
- **MapViewOfFile:** This function is rarely used in benign code, and may be used as part of the process for deploying some sort of shellcode.

- **FindFirstFileW and FindNextFileW:** These imports can be used by the malware to navigate the file system, which is a behavior commonly exhibited by ransomware, or possibly credential stealers.
- **ShellExecuteA:** Can be used by the malware to execute shell commands, possibly for privilege escalation, accessing restricted files, or other uses.
- **BCryptDecrypt snd others:** Functions of this form indicate that some sort of encryption or decryption is taking place.
- **IsDebuggerPresent:** The malware may exhibit different behavior when being debugged versus when running normally.



type (2)	size (bytes)	file-offset	blacklist (72)	hint (176)	group (17)	value (4408)
ascii	15	0x000735E0	x	x	-	C:\ProgramData\
ascii	9	0x0007381C	x	x	-	C:\Users\
ascii	14	0x00073F08	x	x	-	C:\ProgramData
ascii	46	0x0007C444	x	registry	-	HARDWARE\DESCRIPTION\System\CentralProcessor\0
ascii	31	0x00074844	x	file	-	C:\ProgramData\vcruntime140.dll
ascii	27	0x00074864	x	file	-	C:\ProgramData\softokn3.dll
ascii	23	0x00074880	x	file	-	C:\ProgramData\nss3.dll
ascii	27	0x00074898	x	file	-	C:\ProgramData\msvcp140.dll
ascii	26	0x000748B4	x	file	-	C:\ProgramData\mozglue.dll
ascii	26	0x000748D0	x	file	-	C:\ProgramData\freebl3.dll
ascii	27	0x0007C828	x	file	-	C:\Windows\System32\cmd.exe
ascii	16	0x000850F0	x	-	windowing	GetDesktopWindow
ascii	22	0x00085000	x	-	system-information	GetTimeZoneInformation
ascii	18	0x0008508F	x	-	system-information	EnumDisplayDevices
ascii	19	0x00085203	x	-	system-information	GetCurrentHwProfile
ascii	22	0x00084ADD	x	-	storage	GetLogicalDriveStrings
ascii	19	0x00084B31	x	-	storage	SetCurrentDirectory
ascii	29	0x00084C9D	x	-	registry	GetPrivateProfileSectionNames
ascii	21	0x0008533F	x	-	network	HttpAddRequestHeaders
ascii	13	0x00085359	x	-	network	HttpQueryInfo
ascii	16	0x0008536A	x	-	network	InternetReadFile
ascii	22	0x0008537E	x	-	network	InternetSetFilePointer
ascii	19	0x00085398	x	-	network	InternetCloseHandle
ascii	15	0x000853AF	x	-	network	HttpSendRequest
ascii	15	0x000853C3	x	-	network	HttpOpenRequest
ascii	15	0x000853D7	x	-	network	InternetConnect

Figure 26: The strings tab of PEStudio for wotsuper1.exe

PEStudio identified 4408 strings in wotsuper1.exe, 72 of which were listed as blacklisted. Here are some of the most interesting strings:

- **C:\ProgramData\:** It seems that this folder is being accessed by the malware, possibly for logs related to execution activity for the malware.
- **C:\Users\:** This directory is also being accessed, perhaps to collect or modify information for individual users on the infected machine.
- **HARDWARE\DESCRIPTION\System\CentralProcessor\0:** This appears to be referencing a registry key that the malware could be accessing to learn information about the environment it is running in (for example, to see if it is in a sandbox).
- **C:\Windows\System32\cmd.exe:** This string is probably being used by the malware to open a command prompt under certain conditions.
- **Many SQL-style commands:** There are many strings in this file that resemble SQL queries. This implies that some sort of information is being stored or read from

some database. Perhaps there is a database being used to store sensitive information about the infected machine.

- **HTTP header information:** There are several strings in this file that appear to be part of headers for HTTP requests. Using dynamic analysis, we might be able to more closely observe information being sent and received via HTTP. One HTTP header designates the preferred language as Russian, giving another indication that the creator of this malware speaks Russian.
- **\Packages\Microsoft.MicrosoftEdge 8wekyb3d8bbwe\AC\#!001\MicrosoftEdge\Cookies:** This shows that the program may be trying to access cookies used by Microsoft Edge, perhaps to perform some sort of session hijacking attack, or for simple data collection.
- **2010-12-07 20:14:09 a586a4deeb25330037a49df295b36aaf624d0f45:** This appears to be some sort of checksum or similar (probably SHA1) hash for a particular file, along with a timestamp indicating that the file was created in 2010.
- **SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall:** This is a registry key that contains information that will allow programs to be uninstalled if needed. This may be used for uninstalling the wotsuper program, perhaps as a means of making the program appear legitimate.
- **Software\Martin Prikryl\WinSCP 2\Configuration:** This is a registry key that is associated with WinSCP. Could be used by the malware to communicate with other servers via FTP or SFTP.
- **\Authy Desktop\Local Storage*.localstorage:** This references local storage files created by the application “Authy Desktop”. This is a known application that is used for two-factor authentication codes.
- **“os crypt”:{“encrypted key”:** This string seems to be referencing some kind of encryption key that is being used and stored in a file (probably JSON).
- **Many strings referencing cryptocurrency wallets:** This indicates that the malware will likely try to steal information relating to cryptocurrency such as Ethereum and Bitcoin from the infected machine.
- **UseMasterPassword:** Perhaps this malware has some sort of password that is being used to decrypt a lot of different information, and the password is referred to with this string.
- **Passphrase.json:** The name of this file indicates that it could be used to store a passphrase that will allow access to some system.
- **Password %s:** This seems to be the input to a printf, scanf, or some similar function call. Perhaps this is worth investigating to see if there are any potentially meaningful passwords referenced with it.

- **\logins\json:** This is often where saved login credentials are stored, so the malware may be attempting to steal saved information.

wotsuper1.exe has these hashes: b8181cb72764c24e73c7b6204b16bed6 (MD5), c430cc4776ff5e21d08bca9a0d73cfaf29108fa4 (SHA1), fdb5a0d4e97ee36d2b23605b0d8a2785d08d046058f07a8714e4908e8a2485a2 (SHA256).

In VirusTotal, 58 out of 72 vendors marked this executable as malicious.

Section 2F: Decompiling wotsuper1.exe

wotsuper1.exe contains much more code that can be analyzed statically.

```
24   CreateMutexA((LPSECURITY_ATTRIBUTES)0x0,0,mutex_name);
25   buffer_manipulation(not_important,1,0);
26   buffer_manipulation(local_5c,1,0);
27   buffer_manipulation(hw_guid,1,0);
28   last_error = GetLastError();
29   if (last_error == 0xb7) {
30 LAB_004078f1:
31                     /* WARNING: Subroutine does not return */
32     ExitProcess(0);
33   }
```

Figure 27: A segment of code in wotsuper1.exe that creates a mutex

The above code segment is used to create a mutex. If the program recognizes that that mutex with the same name has already been created, the program will shut down so it doesn't interfere with other instances of the same malware.

```
local_1a8 = (uchar *)InternetConnectA(internet_handle,server_name,0x50,param_1[0xf],
                                      param_1[0x10],3,0,1);
if (local_1a8 != (uchar *)0x0) {
  InternetSetOptionA(local_1a8,0x41,1,0);
  ppppuVar3 = (undefined4 ****)local_178[0];
  if (local_164 < 0x10) {
    ppppuVar3 = local_178;
  }
  url_start = HttpOpenRequestA(local_1a8,&DAT_004752a8,ppppuVar3,0,0,0,0x400000,1);
```

Figure 28: Code from wotsuper1.exe used to establish an internet connection

The above code is used by wotsuper1.exe for internet communication. Several other related internet communication functions can be found later in the same function. We will see later that this code is being used to communicate with a C2 server.

Figure 29: A segment of code in wotsuper1.exe used for logging information

The above segment of code allows information about the infected machine to be logged in a file called files\information.txt.

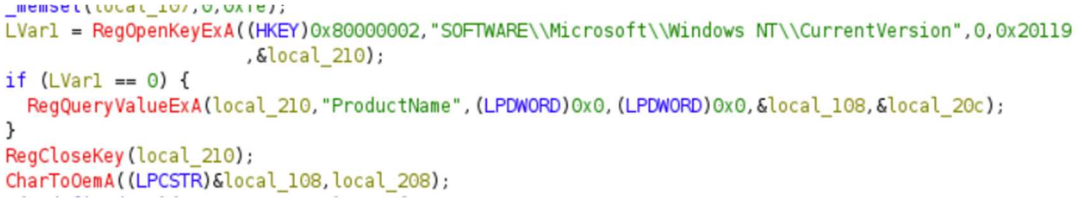


Figure 30: Code for a registry access by wotsuper1.exe

We can see above that the malware is accessing a registry key, and there are many other functions like this one that access different registry keys.

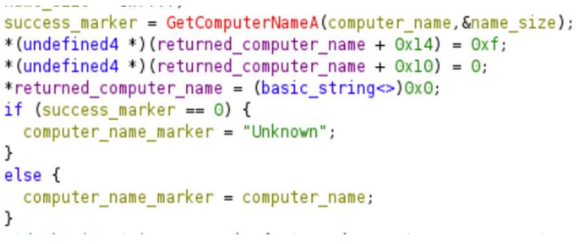


Figure 31: Code for accessing the computer name of the infected machine

The above code allows wotsuper1.exe to get the NetBIOS name of the infected machine.

```
hdc = CreateDCA("DISPLAY",(LPCSTR)0x0,(LPCSTR)0x0,(DEVMODEA *)0x0);
iVar1 = GetDeviceCaps(hdc,8);
iVar2 = GetDeviceCaps(hdc,10);
ReleaseDC((HWND)0x0,hdc);
```

Figure 32: Code to allow wotsuper1.exe to find device-specific information

The code in the screenshot above can be used to learn more information about the infected machine.

```
nBuff = GetKeyboardLayoutList(0,(HKL *)0x0);
lpList = (HKL *)LocalAlloc(0x40,nBuff << 2);
uVar1 = GetKeyboardLayoutList(nBuff,lpList);
```

Figure 33: Code that allows wotsuper1.exe to get keyboard information

The above code allows the program to learn about the keyboard layout of the infected machine.

```
iVar3 = find_first_char(local_2158);
if (iVar3 != -1) {
  local_221c = local_226c;
  local_226c[0] = (basic_string<>)0x0;
  pcStack_2274 = (char *)0x405a40;
  std::basic_string<>::assign(local_226c,"%APPDATA%");
  local_2204._0_1_ = 10;
  pcStack_2274 = (char *)0x405a4e;
  pcStack_228c = _getenv("APPDATA");
  local_2224 = auStack_2288;
  uStack_2290 = 0x405a5c;
  set_more_values();
  local_2218 = acStack_22a4;
  local_2204._0_1_ = 0xb;
  FUN_00402223(local_2158);
  local_2204._0_1_ = 2;
  FUN_00404a01(local_2174);
  local_2204._0_1_ = 0xc;
  memory_management();
  local_2204._0_1_ = 2;
  buffer_manipulation(local_2174);
}
```

Figure 34: A code segment where wotsuper1.exe accesses environment variables

This program may be accessing environment variables so they can be used later for storing log files or for collecting more data.

```
while( true ) {
  FUN_0044f879();
  local_8._0_1_ = 0x32;
  FUN_004088ec();
  local_8._0_1_ = 7;
  buffer_manipulation(local_50);
  FUN_0044f5af();
  local_8._0_1_ = 0x33;
  FUN_004088ec();
  local_8._0_1_ = 7;
  buffer_manipulation(local_50);
  FUN_0044f67e();
  local_8._0_1_ = 0x34;
  FUN_004088ec();
  local_8._0_1_ = 7;
  buffer_manipulation(local_50);
  set_some_values();
  local_8._0_1_ = 0x35;
```

Figure 35: A loop that runs roughly every minute and a half in wotsuper1.exe

A large portion of the code in wotsuper1.exe is run periodically every 86 seconds. The functions in the loop include those that attempt to steal information relating to cryptocurrency. There are also new files created at each iteration of the loop for logging purposes.

```
std::basic_string<>::assign(backslash_ethereum_str,"\\Ethereum\\");
FUN_00407dc2();
keystore_str[0] = (basic_string<>)0x0;
local_54 = 0x4083c0;
std::basic_string<>::assign(keystore_str,"default_wallet");
local_54 = 0xf;
local_58 = 0;
ethereum_str[0] = (basic_string<>)0x0;
local_70 = 0x4083df;
std::basic_string<>::assign(ethereum_str,"Electrum");
local_70 = 0xf;
local_74 = 0;
backslash_ethereum_str[0] = (basic_string<>)0x0;
std::basic_string<>::assign(backslash_ethereum_str,"\\Electrum\\wallets\\");
FUN_00407dc2();
keystore_str[0] = (basic_string<>)0x0;
local_54 = 0x408424;
std::basic_string<>::assign(keystore_str,"default_wallet");
local_54 = 0xf;
local_58 = 0;
ethereum_str[0] = (basic_string<>)0x0;
local_70 = 0x408443;
std::basic_string<>::assign(ethereum_str,"ElectrumLTC");
local_70 = 0xf;
local_74 = 0;
backslash_ethereum_str[0] = (basic_string<>)0x0;
std::basic_string<>::assign(backslash_ethereum_str,"\\Electrum-LTC\\wallets\\");
FUN_00407dc2();
```

Figure 36: Code in wotsuper1.exe that can be used to steal cryptocurrency information

This malware can find wallets for cryptocurrency such as Electrum, Ethereum, DogeCoin, and more.

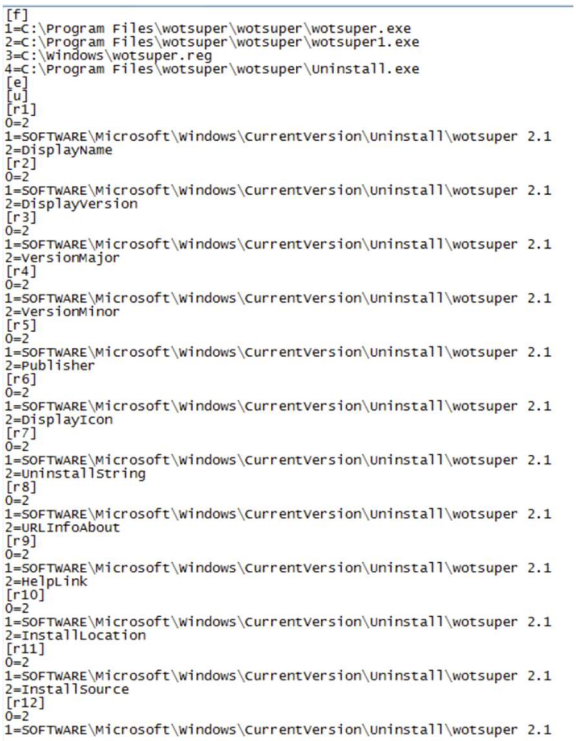
```
std::basic_string<>::assign(abStack_29c,"files\\Cookies\\IE_Cookies.txt");
local_22c = abStack_2b8;
uStack_2a4 = 0xf;
uStack_2a8 = 0;
local_8._0_1_ = 0x17;
abStack_2b8[0] = (basic_string<>)0x0;
uStack_2c0 = 0x4136c8;
```

Figure 37: Internet Explorer cookie information stored by wotsuper1.exe

The malware also accesses cookie information relating to Internet Explorer and Microsoft Edge and saves it to log files.

There are many more functionalities that can be identified in this program, but for the purposes of maintaining a reasonably concise report, any further functions will be omitted from direct discussion. All functionalities found in this program relate to its core purpose of stealing and logging information.

Section 2G: Analyzing Uninstall.exe and Uninstall.ini



```
[f]
1=C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2=C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
3=C:\Windows\wotsuper.reg
4=C:\Program Files\wotsuper\wotsuper\Uninstall.exe
[e]
[u]
[r1]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=DisplayName
[r2]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=DisplayVersion
[r3]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=VersionMajor
[r4]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=VersionMinor
[r5]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=Publisher
[r6]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=DisplayIcon
[r7]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=UninstallString
[r8]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=URLInfoAbout
[r9]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=HelpLink
[r10]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=InstallLocation
[r11]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2=InstallSource
[r12]
0=2
1=SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
```

Figure 38: The contents of Uninstall.ini

In “Uninstall.ini” in the wotsuper folder, we can see a list of files and registry keys created by sample2.exe, likely so they can be deleted properly.

Uninstall.exe appears to be an automatically created file that was added by Smart Install Maker. No vendors on VirusTotal marked this file as malicious. If we run the file after ending the wotsuper.exe and wotsuper1.exe processes, we see that the wotsuper program appears to be uninstalled. It does not appear that Uninstall.exe is doing anything malicious, and so we will omit a full analysis of this file.

Section 3: Dynamic Analysis

Section 3A: Basic dynamic analysis of sample2.exe

In this section, we will run the malware sample2.exe on a Windows 7 machine. The malware sample can be run by double-clicking its icon. Immediately after running the program, two Internet Explorer tabs pop up attempting to access a website via the domain name iplogger.org.

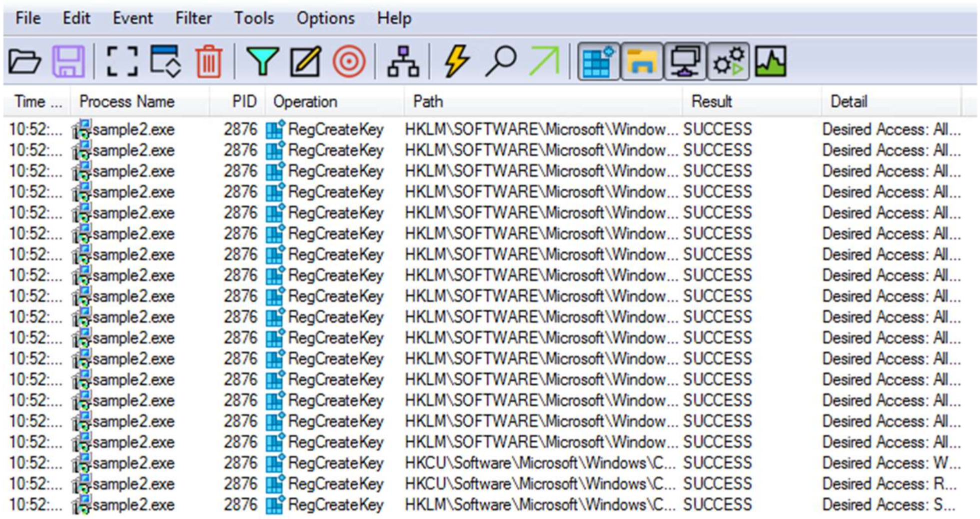


Figure 39: Registry keys created by running sample2.exe

The following registry keys are created by sample2.exe:

1. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1
2. HKCU\Software\Microsoft\Windows\CurrentVersion\ShellExtensions\Cached
3. HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\
4. HKLM\Software\Microsoft\Windows\CurrentVersion\Explorer

The most interesting of these keys is the first one. This registry key was likely added to allow for the wotsuper program to be uninstalled. This could have been done to make the malware appear more legitimate, but it is most likely just a feature automatically implemented by Smart Install Maker.

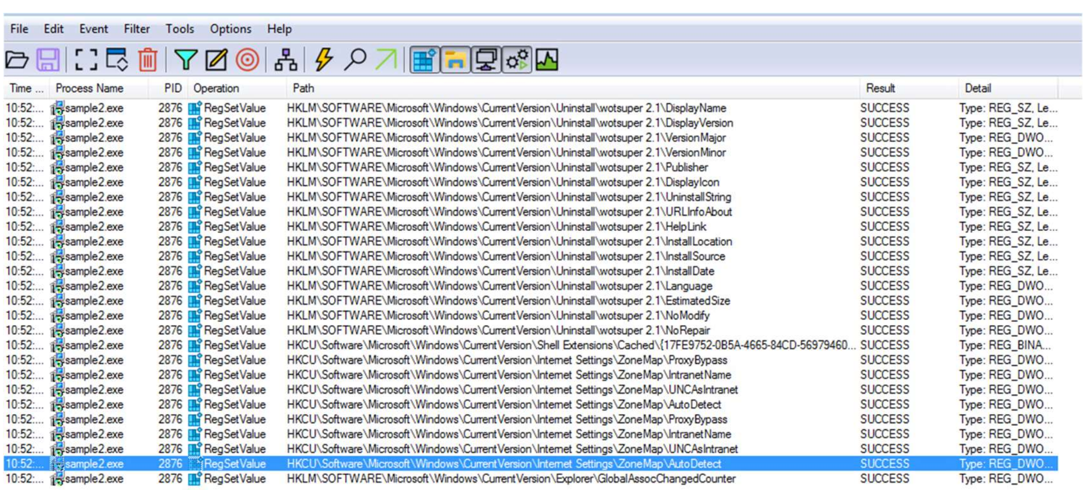
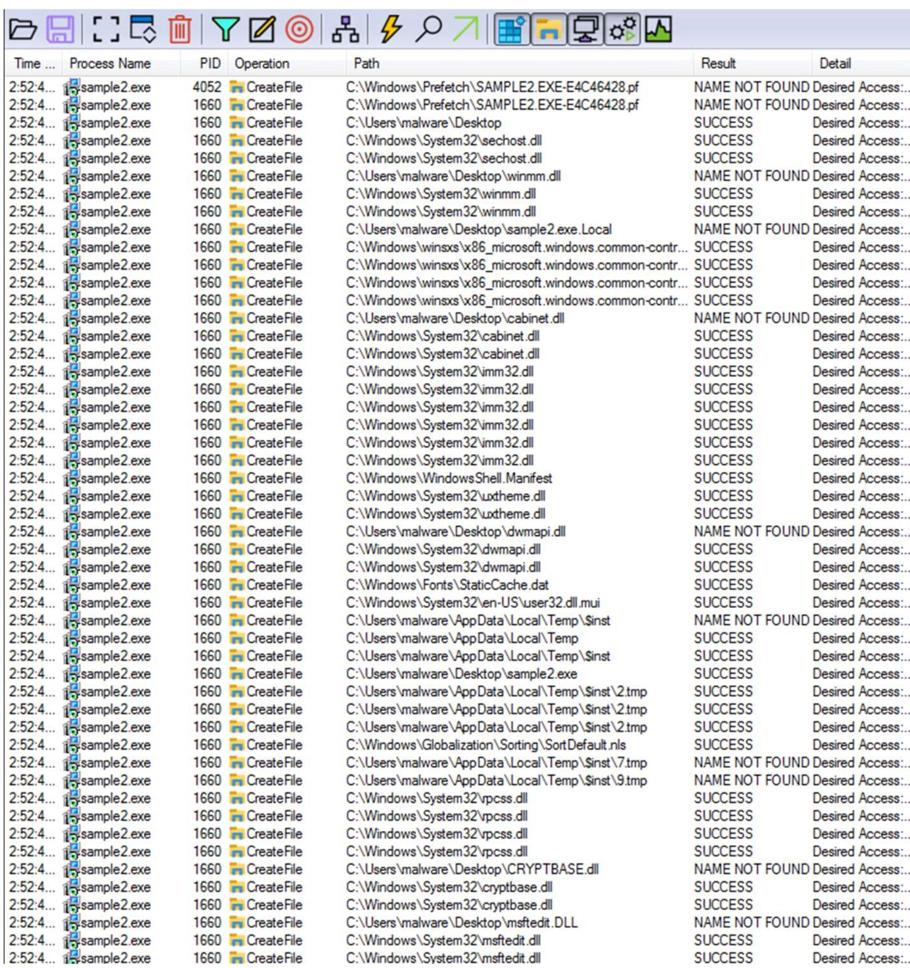


Figure 40: Registry key values modified by sample2.exe

The following registry key values were modified by sample2.exe:

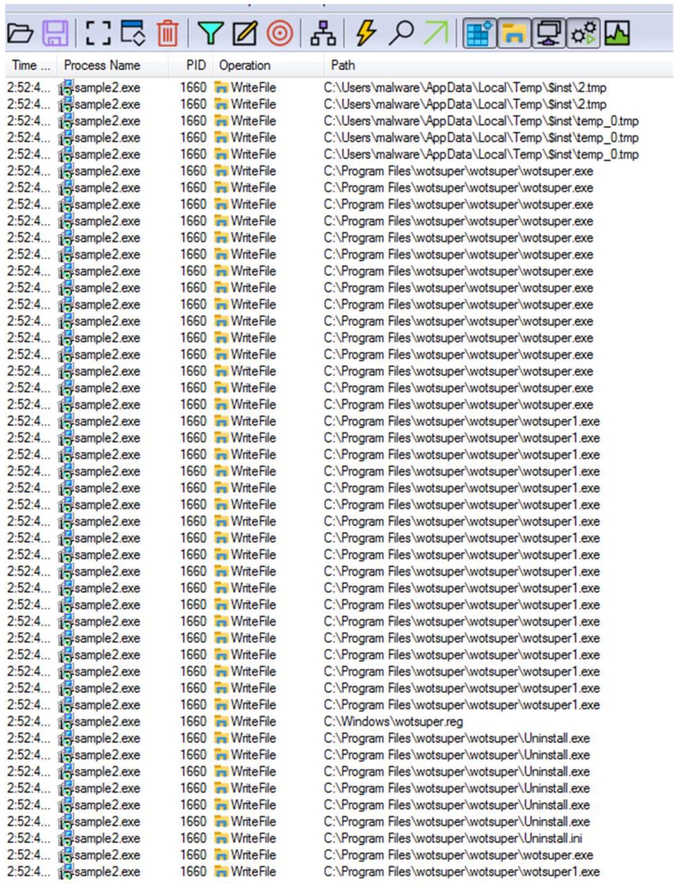
1. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\wotsuper 2.1 (a total of 16 keys is this format were modified, as shown in the screenshot)
2. HKCU\Software\Microsoft\Windows\CurrentVersion\Shell Extensions\Cached\{17FE9752-0B5A-4665-... //Continues here
3. HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass (also IntranetName, UNCAsIntranet, and AutoDetect)
4. HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\GlobalAssocChangedCounter

The first of these modified values seems to be related to the installation of the wotsuper program. Most of the other keys appear to be related to internet communication, suggesting that some sort of internet connection is being made by sample2.exe.



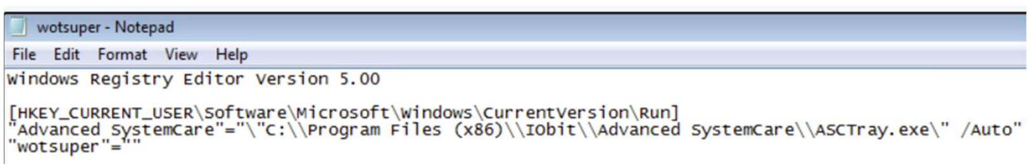
Time ...	Process Name	PID	Operation	Path	Result	Detail
2:52:4...	sample2.exe	4052	CreateFile	C:\Windows\Prefetch\SAMPLE2.EXE-E4C46428.pf	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\Prefetch\SAMPLE2.EXE-E4C46428.pf	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\Desktop	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\sechost.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\sechost.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\Desktop\winmm.dll	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\winmm.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\winmm.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\Desktop\sample2.exe.Local	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\winsxs\x86_microsoft.windows.common-contr...	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\winsxs\x86_microsoft.windows.common-contr...	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\winsxs\x86_microsoft.windows.common-contr...	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\winsxs\x86_microsoft.windows.common-contr...	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\Desktop\cabinet.dll	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\cabinet.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\cabinet.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\imm32.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\imm32.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\imm32.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\imm32.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\imm32.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\imm32.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\WindowsShell.Manifest	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\uxtheme.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\uxtheme.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\Desktop\dwmapi.dll	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\dwmapi.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\dwmapi.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\Fonts\StaticCache.dat	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\en-US\user32.dll.mui	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\AppData\Local\Temp\$inst	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\AppData\Local\Temp	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\AppData\Local\Temp\$inst	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\Desktop\sample2.exe	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\AppData\Local\Temp\$inst\2.tmp	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\AppData\Local\Temp\$inst\2.tmp	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\AppData\Local\Temp\$inst\2.tmp	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\Globalization\Sorting\SortDefault.nls	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\AppData\Local\Temp\$inst\7.tmp	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\AppData\Local\Temp\$inst\9.tmp	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\rpcss.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\rpcss.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\rpcss.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\rpcss.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\Desktop\CRYPTBASE.dll	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\cryptbase.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\cryptbase.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Users\malware\Desktop\msftedit.DLL	NAME NOT FOUND	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\msftedit.dll	SUCCESS	Desired Access:...
2:52:4...	sample2.exe	1660	CreateFile	C:\Windows\System32\msftedit.dll	SUCCESS	Desired Access:...

Figure 41: Files created by sample2.exe



Time ...	Process Name	PID	Operation	Path
2:52:4...	sample2.exe	1660	WriteFile	C:\Users\malware\AppData\Local\Temp\$inst\2.tmp
2:52:4...	sample2.exe	1660	WriteFile	C:\Users\malware\AppData\Local\Temp\$inst\2.tmp
2:52:4...	sample2.exe	1660	WriteFile	C:\Users\malware\AppData\Local\Temp\$inst\temp_0.tmp
2:52:4...	sample2.exe	1660	WriteFile	C:\Users\malware\AppData\Local\Temp\$inst\temp_0.tmp
2:52:4...	sample2.exe	1660	WriteFile	C:\Users\malware\AppData\Local\Temp\$inst\temp_0.tmp
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Windows\wotsuper.reg
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\Uninstall.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\Uninstall.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\Uninstall.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\Uninstall.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\Uninstall.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\Uninstall.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\Uninstall.ini
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	WriteFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe

Figure 42: Files written to by sample2.exe




```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run]
"Advanced SystemCare"="\"C:\\Program Files (x86)\\IObit\\Advanced SystemCare\\ASCTray.exe\" /Auto"
"wotsuper"=""
```

Figure 43: The contents of the newly created wotsuper.reg file after running sample2.exe

Here are some of the most interesting files accessed or modified by the malware:

- C:\Users\malware\AppData\Local: A few different files in this folder are accessed, possibly to store logs of the running executable file and any network communications it might be performing.
- C:\Windows\win.ini: Used in older versions of Windows for various system settings, and still supported for backwards compatibility.
- C:\Windows\Registration\ R00000000001e.clb: Used for storing COM+ component information for applications (this is probably related to standard program installation via Smart Install Maker).

- **C:\Program Files\wotsuper\wotsuper:** A few files inside this folder are created, including wotsuper.exe, wotsuper1.exe, Uninstall.exe, Uninstall.ini, and ui\SwDRM.dll. These files appear to be the main executables for the wotsuper program, and we have analyzed them in detail in the static analysis section.
- **C:\Windows\wotsuper.reg:** This is a script that can be used to modify various keys in the registry. Its contents immediately after running sample2.exe are shown in the above screenshot.
- C:\Program Files\Internet Explorer\iexplore.exe: This is used for accessing Internet Explorer, which is not surprising since we can observe Internet Explorer windows opening when we run the malware.
- C:\Windows\regedit.exe: We already know the malware is modifying the registry, so this also makes sense.

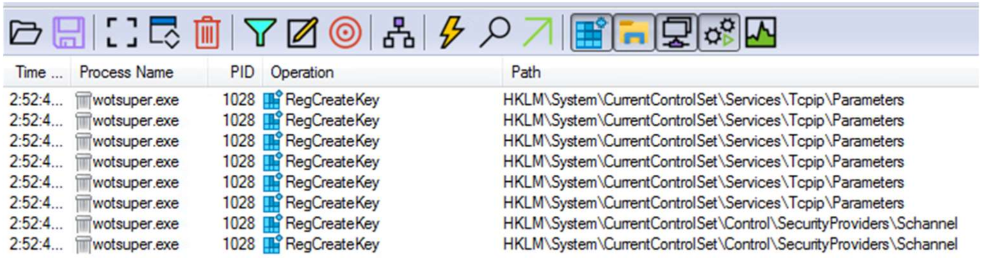
The malware also attempts to access a file C:\Users\malware\Desktop\https:\iplogger.org, but fails. This seems to be some sort of bug in the behavior of the program, causing this domain name to be incorrectly interpreted as a file.

Time ...	Process Name	PID	Operation	Path
2:52:4...	sample2.exe	1660	Process Create	C:\Program Files\Internet Explorer\iexplore.exe
2:52:4...	sample2.exe	1660	Process Create	C:\Program Files\wotsuper\wotsuper\wotsuper.exe
2:52:4...	sample2.exe	1660	Process Create	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe
2:52:4...	sample2.exe	1660	Process Create	C:\Windows\regedit.exe
2:52:4...	sample2.exe	1660	Process Create	C:\Program Files\Internet Explorer\iexplore.exe

Figure 44: Processes created by sample2.exe

sample2.exe creates five new processes. Two of these processes are for opening Internet Explorer; these processes are likely related to the tabs that open to the iplogger.org addresses. Two more processes are created based on the executables wotsuper.exe and wotsuper1.exe that were created. A final process is created for regedit.exe, which is used for modifying registry keys.

There do not appear to be any new services started by sample2.exe. We will now take a closer look at the registry keys, files, processes, and services created or modified by the new processes based on wotsuper.exe and wotsuper1.exe. We will start with the former. (We will omit discussion of modifications by the regedit.exe and iexplore.exe processes because there does not appear to be anything interesting that they change.)



Time ...	Process Name	PID	Operation	Path
2:52:4...	wotsuper.exe	1028	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper.exe	1028	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper.exe	1028	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper.exe	1028	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper.exe	1028	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper.exe	1028	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper.exe	1028	RegCreateKey	HKLM\System\CurrentControlSet\Control\SecurityProviders\Schannel
2:52:4...	wotsuper.exe	1028	RegCreateKey	HKLM\System\CurrentControlSet\Control\SecurityProviders\Schannel

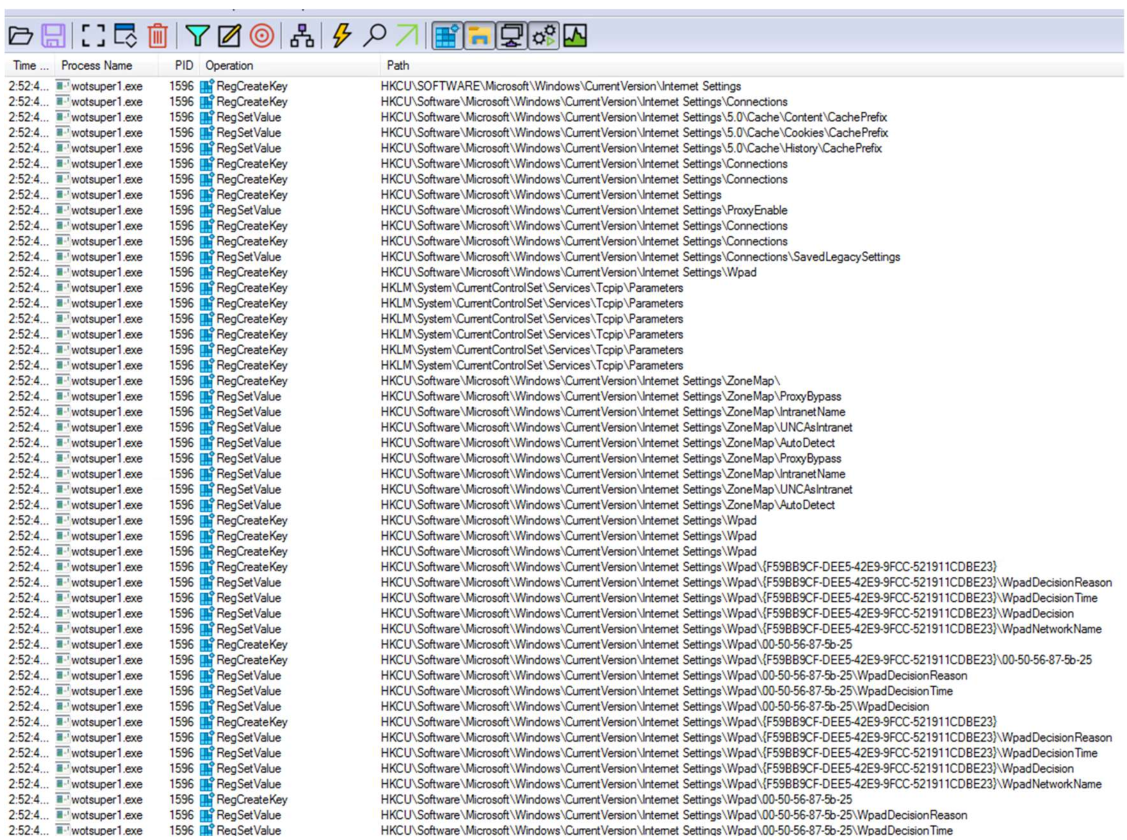
Figure 45: Registry created/modified by wotsuper.exe

The following registry keys are created or modified by the wotsuper.exe process:

1. HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2. HKLM\System\CurrentControlSet\Control\SecurityParameters\Schannel

These registry key accesses do not appear particularly abnormal.

Similarly, there do not appear to be any abnormal file creations/modifications by wotsuper.exe. wotsuper.exe also does not appear to create any new processes or services. We will now look at wotsuper1.exe, which seems to be more interesting.



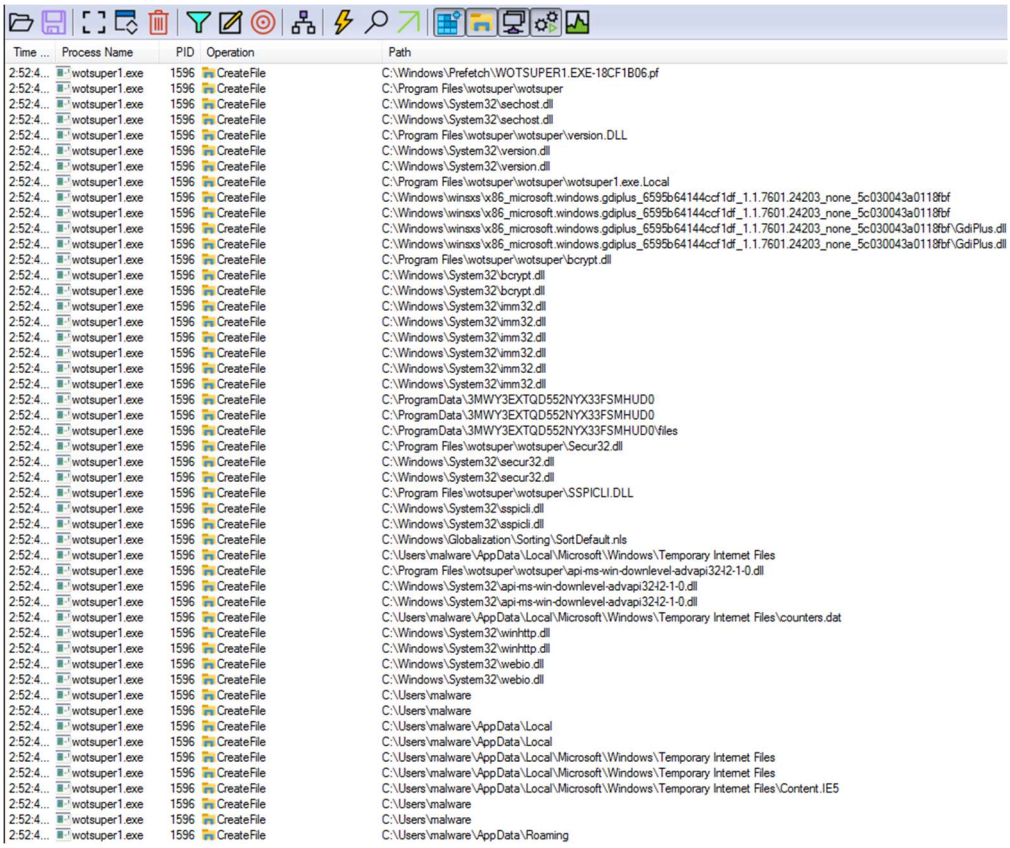
Time ...	Process Name	PID	Operation	Path
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Content\CachePrefix
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Cookies\CachePrefix
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\History\CachePrefix
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ProxyEnable
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections\SavedLegacySettings
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKLM\System\CurrentControlSet\Services\Tcpip\Parameters
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\ProxyBypass
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\IntranetName
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\AutoDetect
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\WpadDecisionReason
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\WpadDecisionTime
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\WpadDecision
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\WpadNetworkName
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\00-50-56-87-5b-25
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\00-50-56-87-5b-25
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\00-50-56-87-5b-25\WpadDecisionReason
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\00-50-56-87-5b-25\WpadDecisionTime
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\00-50-56-87-5b-25\WpadDecision
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\WpadDecisionReason
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\WpadDecisionTime
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\WpadDecision
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\{F59BB9CF-DEE5-42E9-9FCC-521911CDBE23}\WpadNetworkName
2:52:4...	wotsuper1.exe	1596	RegCreateKey	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\00-50-56-87-5b-25
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\00-50-56-87-5b-25\WpadDecisionReason
2:52:4...	wotsuper1.exe	1596	RegSetValue	HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Wpad\00-50-56-87-5b-25\WpadDecisionTime

Figure 46: Registry keys created or modified by wotsuper1.exe

The following registry keys are created/modified by wotsuper1.exe:

1. HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings
2. HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Connections
3. HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Content\CachePrefix
4. *Many more registry keys that are in*: HKCU\Software\Microsoft\Windows\CurrentVersion\Internet Settings
5. HKLM\System\CurrentControlSet\Services\Tcpip\Parameters

The keys this file is accessing indicate that some internet communication is taking place.



Time ...	Process Name	PID	Operation	Path
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\Prefetch\WOTSUPER1.EXE-18CF1B06.pf
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Program Files\wotsuper\wotsuper
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\sechost.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\sechost.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Program Files\wotsuper\wotsuper\version.DLL
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\version.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\version.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Program Files\wotsuper\wotsuper\wotsuper1.exe.Local
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\winsxs\x86_microsoft.windows.gdiplus_6595b64144ccf1df_1.1.7601.24203_none_5c030043a0118fbf
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\winsxs\x86_microsoft.windows.gdiplus_6595b64144ccf1df_1.1.7601.24203_none_5c030043a0118fbf
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\winsxs\x86_microsoft.windows.gdiplus_6595b64144ccf1df_1.1.7601.24203_none_5c030043a0118fbf\GdiPlus.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\winsxs\x86_microsoft.windows.gdiplus_6595b64144ccf1df_1.1.7601.24203_none_5c030043a0118fbf\GdiPlus.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Program Files\wotsuper\wotsuper\bcrypt.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\bcrypt.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\bcrypt.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\imm32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\imm32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\imm32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\imm32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\imm32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\imm32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\ProgramData\3MWY3EXTQD552NYX33FSMHUD0
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\ProgramData\3MWY3EXTQD552NYX33FSMHUD0
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\ProgramData\3MWY3EXTQD552NYX33FSMHUD0\files
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Program Files\wotsuper\wotsuper\Secur32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\secur32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\secur32.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Program Files\wotsuper\wotsuper\SSPICLI.DLL
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\sspicli.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\sspicli.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\Globalization\Sorting\SortDefault.nls
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware\AppData\Local\Microsoft\Windows\Temporary Internet Files
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Program Files\wotsuper\wotsuper\api-ms-win-downlevel-advapi32-2-1-0.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\api-ms-win-downlevel-advapi32-2-1-0.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\api-ms-win-downlevel-advapi32-2-1-0.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware\AppData\Local\Microsoft\Windows\Temporary Internet Files\counters.dat
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\winhttp.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\winhttp.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\webio.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Windows\System32\webio.dll
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware\AppData\Local
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware\AppData\Local
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware\AppData\Local\Microsoft\Windows\Temporary Internet Files
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware\AppData\Local\Microsoft\Windows\Temporary Internet Files
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware
2:52:4...	wotsuper1.exe	1596	CreateFile	C:\Users\malware\AppData\Roaming

Figure 47: Files created or modified by wotsuper1.exe

We can see that wotsuper1.exe is creating a folder **C:\ProgramData\3MWY3EXTQD552NYDX33FSMHUD0\files** (and as the process continues running, more folders with similar names are created). Inside this directory, a folder **Soft\Authy** is also created. This seems to be related to the Authy 2FA software (https://www.authy.com/), so perhaps the program is trying to collect information to bypass 2FA. There are also many temporary folders and files created in

C:\Users\malware\AppData that are automatically removed from the system after a certain point in the program's execution. Three of these folders were Local\Microsoft\Windows\Temporary\Internet Files, Roaming\Microsoft\Windows\Cookies, and Local\Microsoft\Windows\History.

Like wotsuper.exe, this program does not appear to create any new processes or services. Next, we will look for any IP addresses or domain names that sample2.exe, wotsuper.exe, or wotsuper1.exe are trying to communicate with.

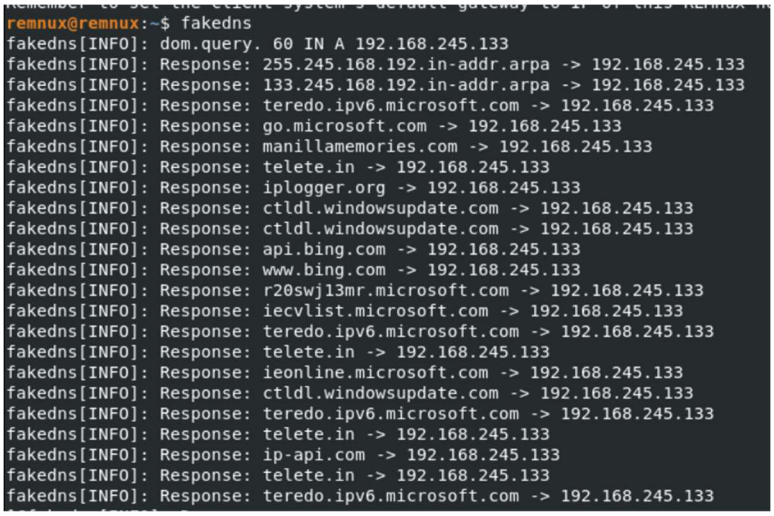


Figure 48: DNS queries found by fakedns after running sample2.exe

There are several interesting domain names the malware is trying to access, including the following:

1. **manillamemories.com**: This URL appears to lead to a C2 server that is known to be associated with the Azorult and Arkei families of info stealer malware (https://github.com/Esox-Lucius/PiHoleblocklists/blob/main/AZORult%20-%20Malware%20Domain%20Feed%20V2 and https://github.com/Esox-Lucius/PiHoleblocklists/blob/main/Arkei%20-%20Malware%20Domain%20Feed%20V2). This domain name is accessed by wotsuper1.exe specifically.
2. **telete.in:** This is another known malicious domain associated with the Raccoon family of info stealer malware (https://www.csk.gov.in/alerts/Raccoon_stealer.html). On closer inspection, this domain name is accessed by wotsuper.exe.
3. **Iplogger.org:** A website that tracks the IP address of visitors to the site.
4. **Ip-api.com:** Used for geolocation of a machine based on its IP address.

Section 3B: Debugging sample2.exe

When debugging sample2.exe, we will first unpack the file. One way to do this is to set a breakpoint on calls to CreateFileA, and wait until the wotsuper.exe file is created. Then, we can scroll up to the start of the function this is called in to find the OEP. Scylla will identify the unpacked program with a breakpoint set there. (We showed the output of the unpacked program in PEStudio earlier in the report.)

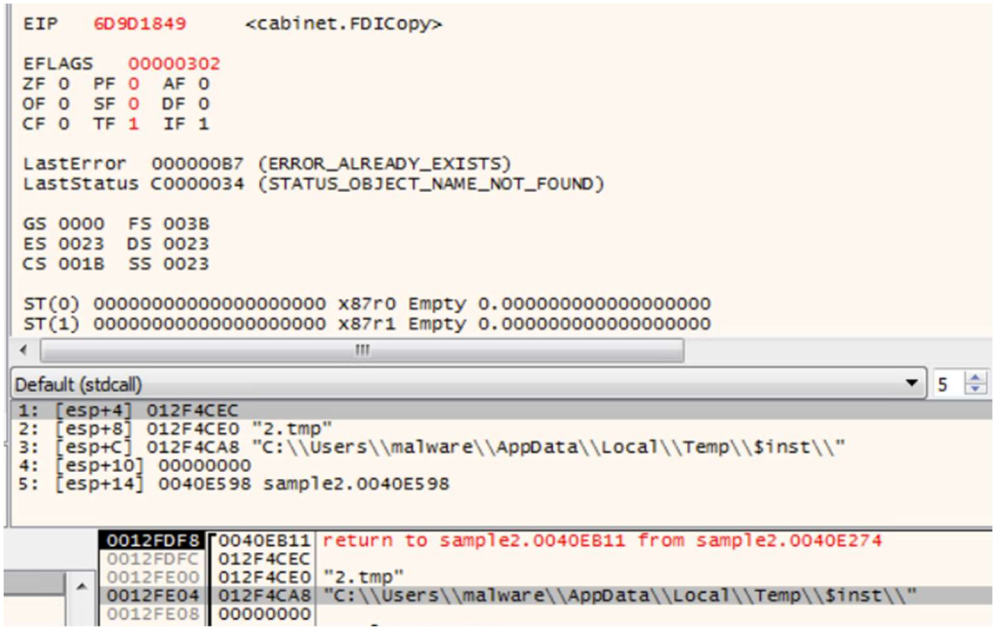


Figure 49: The inputs to FDICopy in sample2.exe

In the static analysis section, we identified that the packing process used CAB files. By observing the inputs to the CAB decompressing function FDICopy, we can see a reference to a file called “2”. Later, we see this function called again to create a file “temp_0”. Both of these are created in C:\Users\malware\AppData\Local\Temp\$inst\.

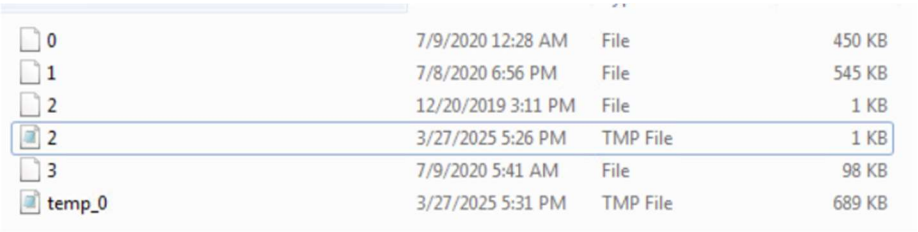


Figure 50: Temporary files created in the unpacking process

The “2” file is an empty CAB file, while the “temp_0” file is a CAB file that contains data. We can unzip temp_0 with 7zip to find four files labeled from “0” to “3” that were compressed inside it.

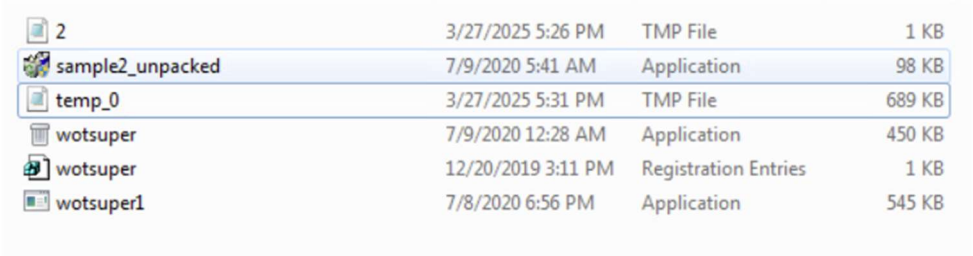


Figure 51: The unpacking files from sample2.exe renamed to their original names

Analyzing the hashes of these files, we see that they are the files created by sample2.exe. "0" contained wotsuper.exe, "1" contained wotsuper1.exe, "2" contained wotsuper.reg, and "3" contained the original unpacked code for sample2.exe. Note that the hashes were different for this copy of sample2.exe and the copy found using Scylla, because the Scylla unpacked copy still contained some artifacts of the Smart Install Maker and CAB files that are not present in this sample2.exe copy.

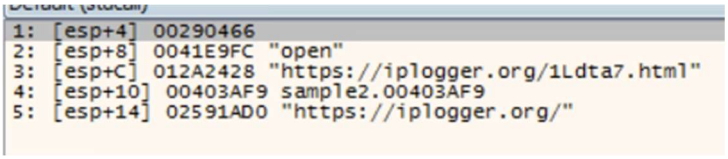


Figure 52: An input to ShellExecuteA in sample2.exe to start an Internet Explorer process

In the static analysis section, we identified a ShellExecuteA command that was being run, and using dynamic analysis, we can verify that it is being used to create the five processes started by sample2.exe. Above, we show the creation of an Internet Explorer process with iplogger.org, and below, we show the creation of the wotsuper1.exe process.

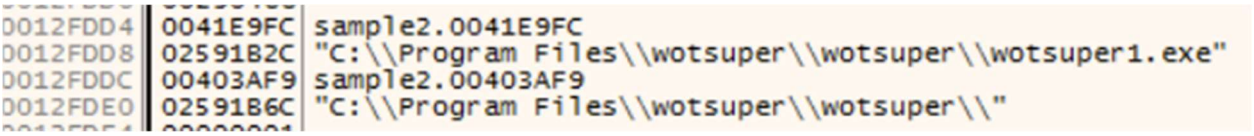


Figure 53: Inputs to ShellExecuteA to start the wotsuper1.exe process

Now that we've analyzed sample2.exe thoroughly, we can summarize its behavior as follows. The program is packed, and contains four files inside a compressed CAB archive that was stored in this file using Smart Install Maker. This program then installs a program called "wotsuper" to the infected machine, during which many registry keys are added or modified, three new files are created, and five processes are started, including two Internet Explorer processes connecting to iplogger.org.

Section 3C: Debugging wotsuper.exe

As noted in the static analysis section, this program appears to be decrypting the payload stored in its .data section. We will see that the decrypted information contains a shellcode.

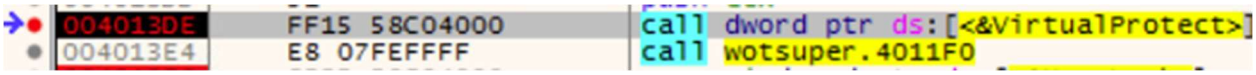


Figure 54: A call to VirtualProtect in wotsuper.exe

To find the shellcode, we want to see where memory is being manipulated. Recall that in the static analysis section, we identified a call to VirtualProtect. We can set a breakpoint to the call to this function, which happens at address 0x4013DE.

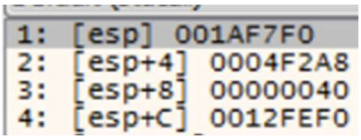


Figure 55: The arguments passed to the VirtualProtect call in wotsuper.exe

The above screenshot shows the parameters to the VirtualProtect call. Most notably, we can see the address of the section being changed (the first argument), and the protection values for that section (the third argument, which is 0x40, or PAGE_EXECUTE_READWRITE; typical of shellcode).

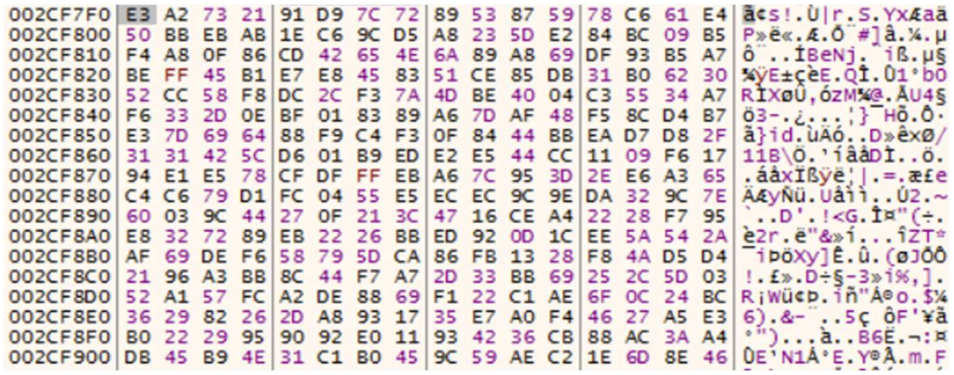


Figure 56: The encrypted contents of the .data section stored in memory

By viewing the memory contents before this point, we can see the encrypted .data payload has been loaded into this location (note that the pictured address is different because the shellcode address is different each time the program is run). Immediately following the VirtualProtect call, another function is called, and this is the function we discussed in the static analysis section that performs the actual decryption behavior.

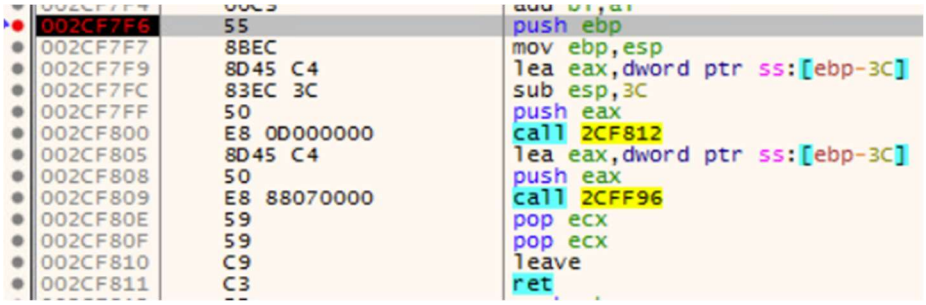


Figure 57: The start of the shellcode decrypted from the .data section

The above screenshot shows the top-level function in the loaded shellcode. Since we know from the previous section that wotsuper.exe is trying to communicate with telete.in, the loaded shellcode is most likely a member of the Raccoon malware family.

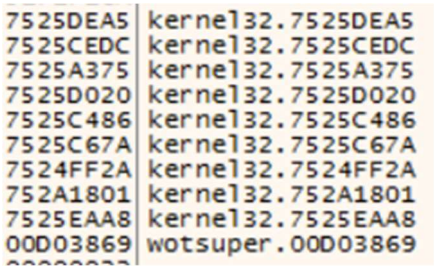


Figure 58: Kernel32 functions stored in memory for the shellcode to use

The shellcode finds and saves to memory the addresses of different kernel32 functions for later use. Shown in the above screenshot are LoadLibraryA, GetProcAddress, GlobalAlloc, GetLastError, Sleep, VirtualAlloc, CreateToolHelp32Snapshot, Module32First, and CloseHandle. Other functions, such as VirtualProtect, are saved for use elsewhere in the shellcode.

To summarize, wotsuper.exe is a launcher that contains an encrypted payload in its .data section. This payload is loaded into memory, and decrypted to get a shellcode. This shellcode, which appears to implement a Raccoon infostealer, is then run by the program.

Section 3D: Debugging wotsuper1.exe

From the static analysis, we have identified several interesting behaviors exhibited by this program, all of which focused on stealing and logging information from the infected machine. Because there is so much interesting information to find in this program, we will focus only on some of the most notable findings from the dynamic analysis here.

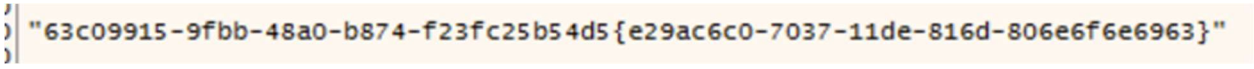


Figure 59: The string used to create a mutex at the beginning of the program

Recall that this program will create a mutex near the beginning of its execution. If any other processes on the system are in possession of the same mutex, the program will immediately exit.

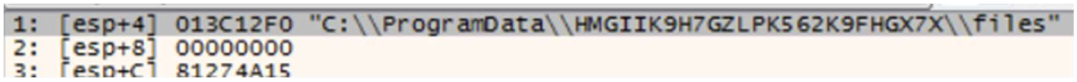


Figure 60: The parameters input to CreateDirectoryA before entering the execution loop

We can observe that this program creates a folder in C:\ProgramData each time it is executed. Inside this directory, a folder called “files” is created that can contain the files information.txt (more on this next), and passwords.txt (used for storing credentials to different services, such as those related to cryptocurrency). We also see that a new compressed file is created every 86 seconds, a behavior showcased in the below screenshot.

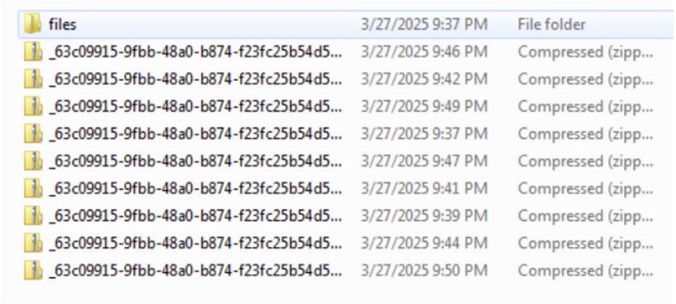


Figure 61: The format of log files used by wotsuper1.exe

The information.txt files that are created are deleted periodically by the malware, so to examine their contents, we need to stop execution before the files are removed. Below, we show the contents of information.txt as it was created at the start of execution.

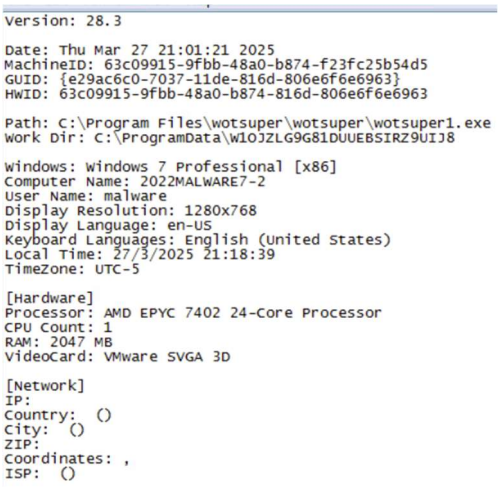


Figure 62: The first segment of the information.txt file

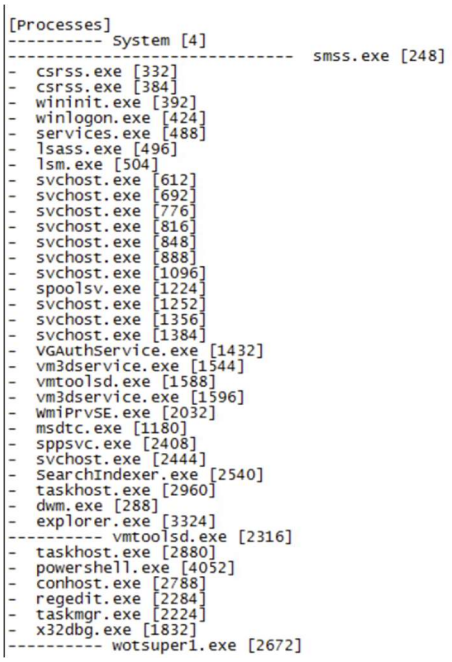


Figure 63: The second segment of the information.txt file

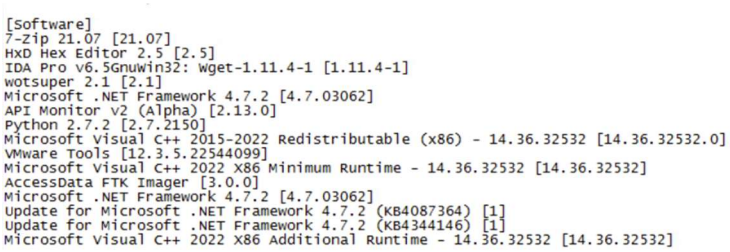


Figure 64: The third segment of the information.txt file

From the above screenshots, we can see that the malware is logging information about the host machine, including installed programs, running processes, network information, processor details, and more.

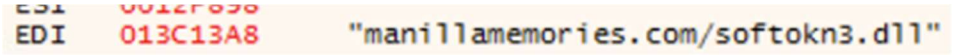


Figure 65: An argument passed to InternetConnectA by wotsuper1.exe

In the above screenshot, we can see more evidence that wotsuper1.exe is communicating with the known C2 server manillamemories.com. This communication suggests this malware is a variant of the Arkei infostealer malware. This website suggests the specific variant being used here is Vidar: https://any.run/report/837eb41e90a4ce53eeb107e3ad39c7baa22adfae133e0ff2672ecd3a323f7b04/a1f69fc3-ee52-4c92-a155-606f9f4e10a3. (According to https://www.hhs.gov/sites/default/files/vidar-malware-analyst-note-tlpclear.pdf, Vidar is a

direct evolution of the Arkei malware.) Also, the malware seems to be trying to download a file called “softokn3.dll”. This DLL appears to be related to Mozilla Firefox (https://bugzilla.mozilla.org/show_bug.cgi?id=489961), but it is possible this file is disguising itself as legitimate, but is actually be used to issue commands or for some other purpose.

Section 4: Indicators of Compromise and YARA rule

Host-based indicators:

- “wotsuper” folder in C:\Program Files containing executable files
- Running processes “wotsuper.exe” and “wotsuper1.exe”
- Folders in C:\ProgramData with 25 uppercase characters in their names that contain compressed files and a “files” folder.
- “wotsuper 2.1” key added to registry in HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
- Created file C:\Windows\wotsuper.reg

Network-based indicators:

- Outgoing communication to any of the following domains:
  - telete.in
  - manillamemories.com
  - iplogger.com/1Ldta7.html
  - iplogger.com/1smEq7.html

YARA rule:

```
rule sample2
{
        strings:
        $wotsuper = "@$&%04\\wotsuper.exe"
        $wotsuper1 = "@$&%04\\wotsuper1.exe"
        $wotsuper_reg = "@$&%02\\wotsuper.reg"
        $wotsuper_url = "http://www.wotsuper.com/"
        $wotsuper_email = "wotsuper@company.com"
        $iplogger1 = "https://iplogger.org/1Ldta7.html"
        $iplogger2 = "https://iplogger.org/1smEq7.html"

        condition:
        all of ($wotsuper*) and all of ($iplogger*)
}
```

Figure 66: A YARA rule that catches sample2.exe

The YARA rule given in the above screenshot can be used to catch sample2.exe. The rule involves searching for specific strings that appear in the malware sample. The first three rules refer to strings that match the names of files created by sample2.exe. Two other

included strings reference a wotsuper email and URL. The last two strings reference the specific iplogger.org pages that are accessed by the malware via Internet Explorer. This rule matches sample2.exe, but not any of the samples in the collection of Practical Malware Analysis labs.

Section 5: Analyzing sample2b.exe

For this practical, we were given another malware sample, “sample2b.exe”. In this section, we will give a very brief analysis of this file, focusing on how its functionality relates to that of sample2.exe.

property	value
md5	212FD7935D3F4ACD5DD27530B44B1F88
sha1	D35A4FFE496EE726BDC309329E21DA119C44A331
sha256	5558CAFDF5ABF043700E6E76FFD0FF449C0F817AD9F1E26878ADE02DA99B9DE9
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes-text	M Z ..... @ .....
file-size	596480 (bytes)
size-without-overlay	n/a
entropy	7.397
imphash	578DC7DAB5389346CBCC286CAE0D11E3
signature	Borland Delphi
entry-point	8B FF 55 8B EC E8 06 90 00 00 E8 11 00 00 00 5D C3 CC CC CC CC CC CC CC CC CC CC CC CC CC CC C...
file-version	n/a
description	n/a
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x5EF2DD69 (Wed Jun 24 00:58:17 2020)
debugger-stamp	0x61454108 (Fri Sep 17 21:29:44 2021)
resources-stamp	
exports-stamp	n/a
version-stamp	empty
certificate-stamp	n/a

Figure 67: The output in PEStudio for sample2.exe

We can see that, like sample2.exe, sample2b.exe seems to have been compiled from Delphi code for 32-bit architectures, and it also has GUI capabilities. Notably, the compiler stamp for sample2.exe is only a few weeks prior to that of wotsuper1.exe, indicating that there may be some similarity in the code of both programs.

property	value	value	value	value
name	.text	.data	.rsrc	.reloc
md5	CB8715439FA195C0541A9D...	36AA616AC8A08A3ADA54A...	40BE7418C6399B5E0F2FA17...	9E492AD6E41B17532C29E76...
entropy	7.552	3.290	5.946	3.460
file-ratio (99.83%)	91.76 %	1.37 %	2.66 %	4.03 %
raw-address	0x00000400	0x00085E00	0x00087E00	0x0008BC00
raw-size (595456 bytes)	0x00085A00 (547328 bytes)	0x00002000 (8192 bytes)	0x00003E00 (15872 bytes)	0x00005E00 (24064 bytes)
virtual-address	0x00401000	0x00487000	0x007B9000	0x007BD000
virtual-size (3933136 bytes)	0x0008589A (546970 bytes)	0x00331064 (3346532 bytes)	0x00003DE8 (15848 bytes)	0x00005CEA (23786 bytes)
entry-point	0x00064E00	-	-	-
characteristics	0x60000020	0xC0000040	0x40000040	0x42000040
writable	-	x	-	-
executable	x	-	-	-
shareable	-	-	-	-
discardable	-	-	-	x
initialized-data	-	x	x	x
uninitialized-data	-	-	-	-
unreadable	-	-	-	-
self-modifying	-	-	-	-
virtualized	-	-	-	-
file	n/a	n/a	n/a	n/a

Figure 68: The sections tab of sample2b.exe in PEStudio

Examining PEStudio, we can observe that the .text section has a very high entropy value of 7.552. This indicates that the main code section is packed or otherwise obfuscated. To unpack the malware, we will need to run it, but first, we will run it usually on a Windows 7 system.

```
fakedns[INFO]: Response: telegatt.top -> 192.168.245.133
fakedns[INFO]: Response: teredo.ipv6.microsoft.com -> 192.168.245.133
fakedns[INFO]: Response: telegka.top -> 192.168.245.133
fakedns[INFO]: Response: telegin.top -> 192.168.245.133
fakedns[INFO]: Response: t.me -> 192.168.245.133
```

Figure 69: Domains contacted by sample2.exe

When running sample2b.exe, not much appears to happen at first. No interesting files seem to be created, and no registry key modifications stand out as unusual (at least after the first several seconds the malware has run). However, we can see that the malware attempts to communicate with multiple domains. telegatt.top, telegka.top, and telegin.top are all known to be used by variants of the Raccoon information stealer malware (https://www.acronis.com/en-us/cyber-protection-center/posts/raccoon-stealer-a-popular-and-dangerous-threat/). These are likely C2 servers, like the telete.in domain contacted by wotsuper.exe. We can also see the malware attempt to communicate with t.me, which is a short domain name used for Telegram communication (https://telegram.org/faq).

```
00432408   FF15 34B34600   call dword ptr ds:[<&WinHttpOpen>]
           837E 14 10      cmp dword ptr ds:[esi+14],10
           8945 E0         mov dword ptr ss:[ebp-20],eax
00432418   72 02           jb sample2b.43241C
0043241A   8B36            mov esi,dword ptr ds:[esi]
0043241C   8D85 20FCFFFF   lea eax,dword ptr ss:[ebp-3E0]
```

Breakpoint Not Set

esi:&"http://telegatt.top/jdiamond13", [esi]:"http://telegatt.top/jdiamond13"

Figure 70: A call to WinHTTPOpen by sample2b.exe

Next, we will unpack sample2b.exe. Since we know the malware is trying to communicate with remote servers, we can set breakpoints on calls to Windows API functions related to internet communication. In this case, we find that WinHttpOpen is being used. The first call of this function is used to communicate with telegatt.top, specifically to access a file called “jdiamond13”, as pictured above. A file with the same name is also requested by the other two known Raccoon domains. At this point, the program has been unpacked, so we can use Scylla to dump the unpacked file.

property	value	value	value	value	value
name	.text	.data	.rsrc	.reloc	.SCY
md5	F35F9C47B58CA4FCFAA066...	62558A1FF37AA0155973483...	40BE7418C6399B5E0F2FA17...	BE2079A4382D9010CFC64B...	29A2CD7CD774470A6260E8...
entropy	6.545	0.118	5.946	6.602	5.477
file-ratio (99.97%)	13.95 %	85.22 %	0.40 %	0.25 %	0.14 %
raw-address	0x00000400	0x00085E00	0x003B6000	0x003B9E00	0x003BC400
raw-size (3921408 bytes)	0x00085A00 (547328 bytes)	0x00330200 (3342848 bytes)	0x00003E00 (15872 bytes)	0x00002600 (9728 bytes)	0x00001600 (5632 bytes)
virtual-address	0x00401000	0x00487000	0x007B9000	0x007BD000	0x007C3000
virtual-size (3948544 bytes)	0x00086000 (548864 bytes)	0x00332000 (3350528 bytes)	0x00004000 (16384 bytes)	0x00006000 (24576 bytes)	0x00002000 (8192 bytes)
entry-point	0x00032411	-	-	-	-
characteristics	0xE0000020	0xC0000040	0x40000040	0x42000040	0xE0000060

Figure 71: The sections tab of PEStudio after sample2b.exe has been unpacked

We can see above that the entropy of the .text section is normal, and PEStudio can now identify many more strings (nearly 5,000). In the strings we can find an IP address 25.3.82.91, which identifies a machine located in the United Kingdom (tested via https://www.iplocation.net/ip-lookup). We can also see strings like “Login Data”, “\wallets” (probably referencing cryptocurrency wallets), “steamstr” (might reference the Steam gaming platform), “\data.json”, and many more that seem to indicate information stealing behavior similar to the programs installed by sample2.exe.

Based on this brief analysis, we have enough information to reasonably conclude that sample2b.exe is a variant of the Raccoon malware, like wotsuper.exe. Therefore, it appears that sample2b.exe, wotsuper.exe, and wotsuper1.exe all belong to a closely-related family of information stealing malware, and this is how they are related.