SOFTWARE REVERSE ENGINEERING PROJECT PROPOSAL

By

Drew Kerby

Alk203

CSE 6363 Software Reverse Engineering
Class Section 1
Dr. Wesley McGrew
January 31st, 2017

Introduction

The purpose of this assignment is to make a proposal for further research into a version of the *Sakurel* malware sample used to attack the aerospace industry in 2014 [1]. The proposal will provide identifying information of the malware and discuss the process of selecting the sample. Additionally, as the execution of the project is expected to be carried out over three months in a course that focuses on the Intel x86 architecture, this proposal will provide evidence that the selected sample meets the outlined requirements. Lastly, the proposal will discuss the information covered by existing analyses, and outline a plan to add to this knowledge in the three-month scope of the project.

Malware Identification

MD5	c869c75ed1998294af3c676bdbd56851
SHA1	dd3a61eed9c454cf96d882f290abc86108ffeea5
Common Name	Sakurel, Sakula

VirusTotal.com Report

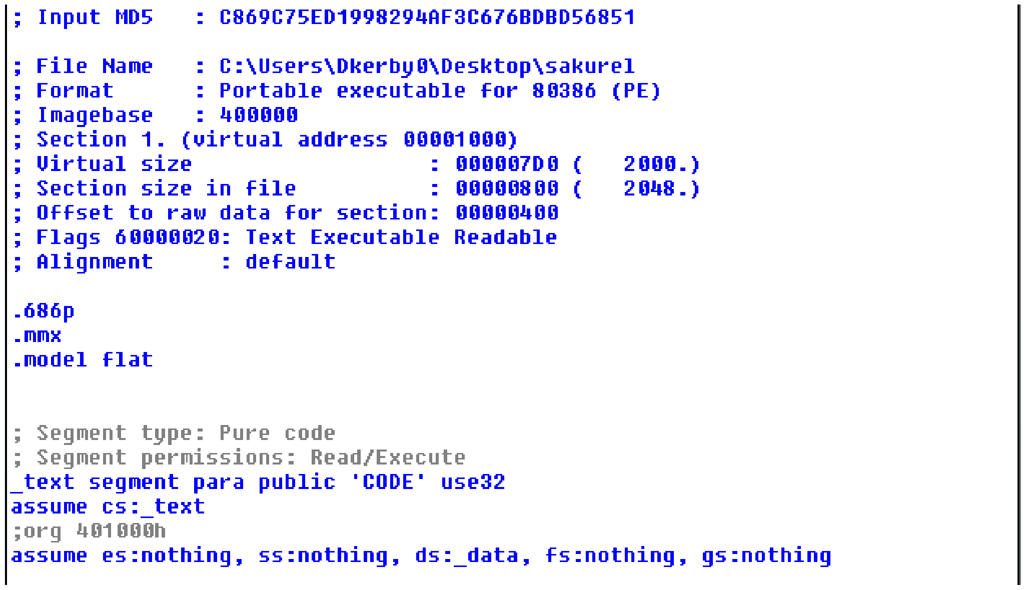
The malware sample report was found on VirusTotal.com using its MD5 hash value. The link to the full report can be found in the references section [1], and a summary will be provided here, due to the considerable length of the report. Out of fifty-seven total antivirus engines used to scan the sample, forty-five detected the malware. It was noted that the signing certificate of the malware was revoked by the issuer, and the malware appeared to be packed. The file metadata indicated that the sample was a Microsoft Windows executable targeting the Intel x86 architecture. Lastly, the report indicated that when executed, the malware sample manipulated files, created processes, and made HTTP requests to the endpoint *oa[.]ameteksen.com*, among other behaviors characteristic of the sample that are outlined in more detail in the full report [1].

Selection of Sample

After browsing the APTnotes repository [2] for a suitable malware family to focus on, I found a security response report from Symantec [3] detailing campaigns targeting the aerospace, energy, and healthcare industries. In this report, MD5 hashes were listed for several of the samples used by Black Vine in their campaigns of recent years. Through search by MD5 hash on VirusShare.com, I located a sample of a Trojan known as *Sakurel* that was used to attack a European aerospace industry in the first half of 2014, according to the Symantec report [3]. Samples of the newer *Mivast* malware used in Black Vine's attack on Anthem [3], a healthcare insurance company, were unable to be located on VirusShare.com or similar sites. A *Mivast* sample would have been preferred due to its more recent creation.

Suitability of Sample to Project Requirements

The malware sample selected, known as *Sakurel*, is primarily composed of 32-bit Intel code and targets Windows. Several steps were taken to ensure that *Sakurel* met these criteria, beyond looking at the malware metadata given in the VirusTotal report [1] and the PE header of the sample. The malware sample was disassembled to verify that the sample used 32-bit Intel code, as shown in Figures 1 and 2 below. Running the malware sample as an administrator spawned a new process called MediaCenter.exe (Figure 3), showing that the sample targets the Windows OS. The filename was consistent with those found in the VirusTotal report [1].



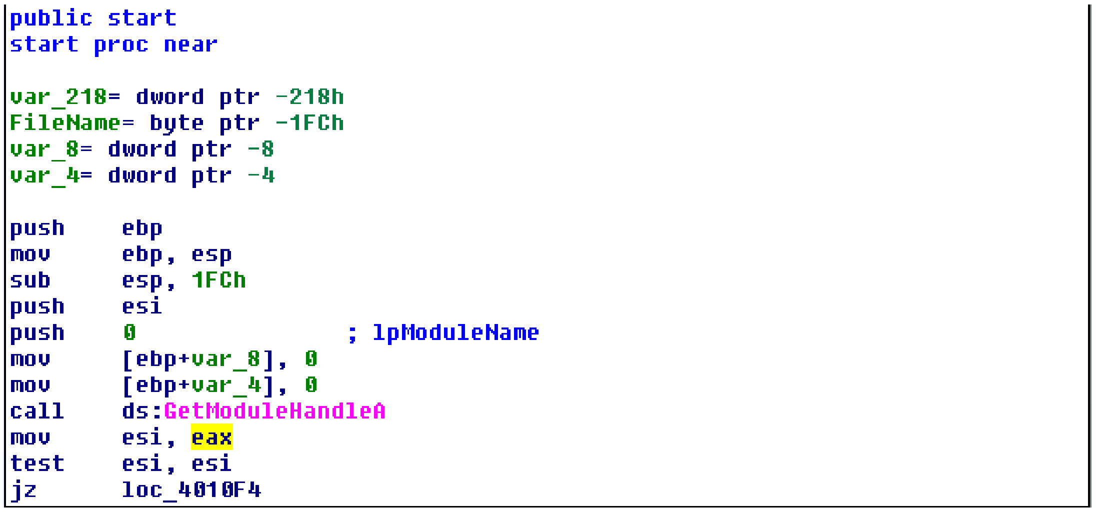
```
; Input MD5    : C869C75ED1998294AF3C676BDBD56851

; File Name    : C:\Users\Dkerby0\Desktop\sakurel
; Format       : Portable executable for 80386 (PE)
; Imagebase    : 400000
; Section 1. (virtual address 00001000)
; Virtual size                  : 000007D0 (    2000.)
; Section size in file          : 00000800 (    2048.)
; Offset to raw data for section: 00000400
; Flags 60000020: Text Executable Readable
; Alignment     : default

.686p
.mmx
.model flat


; Segment type: Pure code
; Segment permissions: Read/Execute
_text segment para public 'CODE' use32
assume cs:_text
;org 401000h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
```

Figure 1. IDA Pro 5.0 identification of sample.

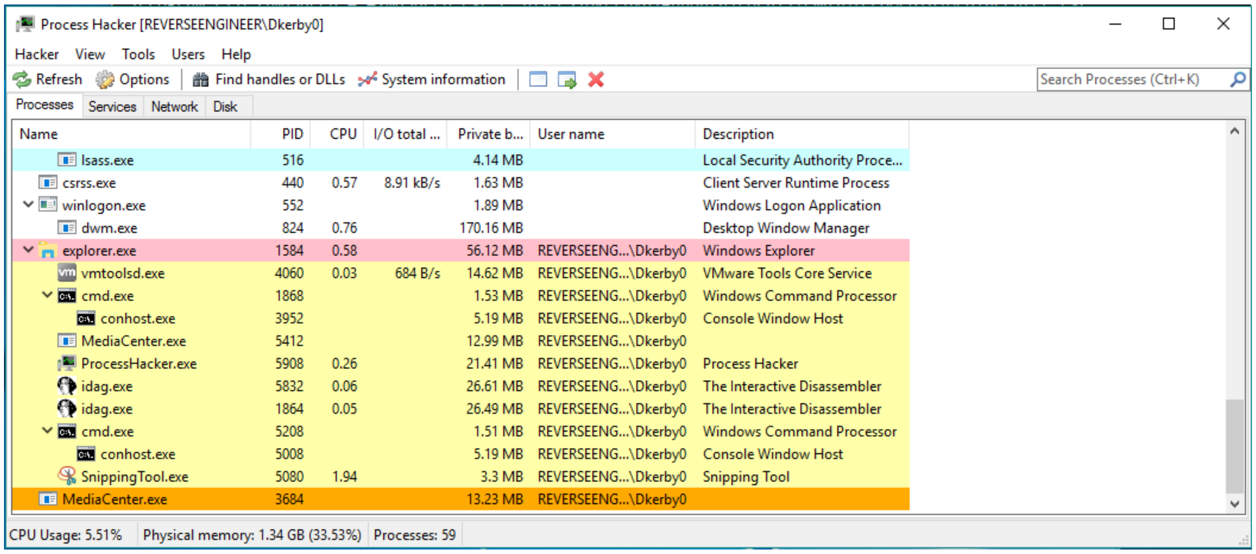


```
public start
start proc near

var_218= dword ptr -218h
FileName= byte ptr -1FCh
var_8= dword ptr -8
var_4= dword ptr -4

push    ebp
mov     ebp, esp
sub     esp, 1FCh
push    esi
push    0                ; lpModuleName
mov     [ebp+var_8], 0
mov     [ebp+var_4], 0
call    ds:GetModuleHandleA
mov     esi, eax
test    esi, esi
jz      loc_4010F4
```

Figure 2. Disassembly of Intel x86 instructions at the program start location using IDA Pro 5.0.



Process Hacker [REVERSEENGINEER\Dkerby0]

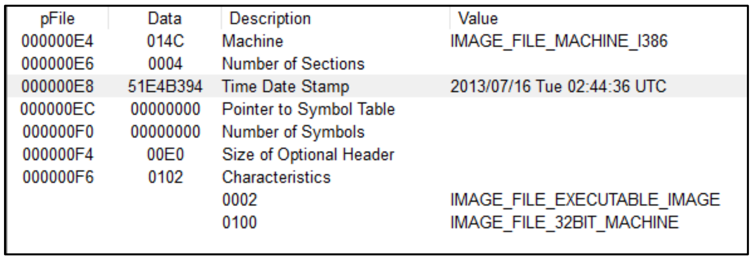
Name	PID	CPU	I/O total ...	Private b...	User name	Description
lsass.exe	516			4.14 MB		Local Security Authority Proce...
csrss.exe	440	0.57	8.91 kB/s	1.63 MB		Client Server Runtime Process
winlogon.exe	552			1.89 MB		Windows Logon Application
dwm.exe	824	0.76		170.16 MB		Desktop Window Manager
explorer.exe	1584	0.58		56.12 MB	REVERSEENG...\Dkerby0	Windows Explorer
vmtoolsd.exe	4060	0.03	684 B/s	14.62 MB	REVERSEENG...\Dkerby0	VMware Tools Core Service
cmd.exe	1868			1.53 MB	REVERSEENG...\Dkerby0	Windows Command Processor
conhost.exe	3952			5.19 MB	REVERSEENG...\Dkerby0	Console Window Host
MediaCenter.exe	5412			12.99 MB	REVERSEENG...\Dkerby0	
ProcessHacker.exe	5908	0.26		21.41 MB	REVERSEENG...\Dkerby0	Process Hacker
idag.exe	5832	0.06		26.61 MB	REVERSEENG...\Dkerby0	The Interactive Disassembler
idag.exe	1864	0.05		26.49 MB	REVERSEENG...\Dkerby0	The Interactive Disassembler
cmd.exe	5208			1.51 MB	REVERSEENG...\Dkerby0	Windows Command Processor
conhost.exe	5008			5.19 MB	REVERSEENG...\Dkerby0	Console Window Host
SnippingTool.exe	5080	1.94		3.3 MB	REVERSEENG...\Dkerby0	Snipping Tool
MediaCenter.exe	3684			13.23 MB	REVERSEENG...\Dkerby0	

CPU Usage: 5.51% Physical memory: 1.34 GB (33.53%) Processes: 59

Figure 3. New process created after running the sample.

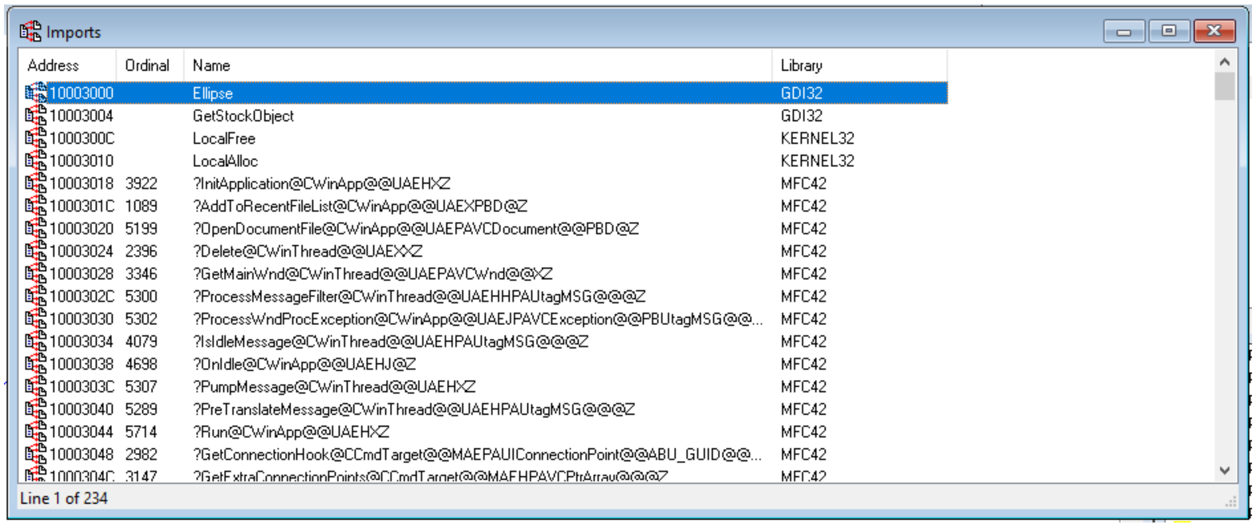
As shown in Figure 4, the compilation date listed in the PE header of the sample is July 16th, 2013, while the Symantec report used as a primary source of information regarding the *Sakurel* malware was published on July 28th, 2015 [3], both of which fall within the time-of-creation requirements for this project. In regards to publicly available source code or documentation for this malware, thorough searches for the MD5 and SHA1 hashes obtained from the malware sample yielded no results, as well as searches for *Sakurel* and *Mivast*. Furthermore, (though not heavily relied upon) the Symantec report provided no indication that any source code or documentation for *Sakurel* was made publicly available online [3].

In fact, the report by Symantec did not contain a very detailed analysis of the selected malware sample, but instead gave a high-level overview of the known features of the family of Trojans used by the Black Vine group [3]. Backdoor-like functionality, execution of files and commands, manipulation of the registry, and the transmission of data gathered on the infected device were listed as features in the report [3]. To verify that the malware sample selected did indeed have enough complexity for a 3-month project, I unpacked and dissembled a suspicious file created by the malware sample after it was executed, *MicroSoftSecurityLogin.ocx*. The disassembled file turned out to be a .dll with a great number of imports relating to GUI and command and control functionality (Figure 5), giving a view into the complexity of the malware.



pFile	Data	Description	Value
000000E4	014C	Machine	IMAGE_FILE_MACHINE_I386
000000E6	0004	Number of Sections	
000000E8	51E4B394	Time Date Stamp	2013/07/16 Tue 02:44:36 UTC
000000EC	00000000	Pointer to Symbol Table	
000000F0	00000000	Number of Symbols	
000000F4	00E0	Size of Optional Header	
000000F6	0102	Characteristics	
		0002	IMAGE_FILE_EXECUTABLE_IMAGE
		0100	IMAGE_FILE_32BIT_MACHINE

Figure 4. File header showing the compilation date of the executable.



Imports

Address	Ordinal	Name	Library
10003000		Ellipse	GDI32
10003004		GetStockObject	GDI32
1000300C		LocalFree	KERNEL32
10003010		LocalAlloc	KERNEL32
10003018	3922	?InitApplication@CWinApp@@UAEHXZ	MFC42
1000301C	1089	?AddToRecentFileList@CWinApp@@UAEXPBD@Z	MFC42
10003020	5199	?OpenDocumentFile@CWinApp@@UAEPAVCDocument@@PBD@Z	MFC42
10003024	2396	?Delete@CWinThread@@UAEXXZ	MFC42
10003028	3346	?GetMainWnd@CWinThread@@UAEPAVCWnd@@XZ	MFC42
1000302C	5300	?ProcessMessageFilter@CWinThread@@UAEHHPAUtagMSG@@@Z	MFC42
10003030	5302	?ProcessWndProcException@CWinApp@@UAEJPAVCException@@PBUtagMSG@@...	MFC42
10003034	4079	?IsIdleMessage@CWinThread@@UAEHPAUtagMSG@@@Z	MFC42
10003038	4698	?OnIdle@CWinApp@@UAEHJ@Z	MFC42
1000303C	5307	?PumpMessage@CWinThread@@UAEHXZ	MFC42
10003040	5289	?PreTranslateMessage@CWinThread@@UAEHPAUtagMSG@@@Z	MFC42
10003044	5714	?Run@CWinApp@@UAEHXZ	MFC42
10003048	2982	?GetConnectionHook@CCmdTarget@@MAEPAUIConnectionPoint@@ABU_GUID@@...	MFC42
1000304C	3147	?GetExtraConnectionPoints@CCmdTarget@@MAEHPAVCPtrArray@@@Z	MFC42

Line 1 of 234

Figure 5. Imports of the hidden .dll file.

Existing Analyses

As previously mentioned, the functionality covered in the report by Symantec was not very in-depth, covering commonalities of the features of three tools used by Black Vine, which are known as *Hurix*, *Sakurel*, and *Mivast* [3]. The aim of the report was to provide details on the Black Vine cyberespionage group, not provide in-depth details on the malware samples mentioned. Fortunately, this leaves several areas of *Sakurel's* functionality open to be to reverse engineered and described at a more detailed technical level by the author of this proposal, as the primary source did not have this goal in mind.

There are several areas that are planned to be included in the scope of this project. First, the encryption algorithm that Hurix and Sakurel utilize to for network communication is described being implemented with arithmetic operations using static variables [3]. Because this description is a fairly vague, I plan to write a Python tool that could be used to decrypt the strings, if the encryption algorithm does not exceed my level of experience. Secondly, there are several GUI manipulation functions found in the malware that I plan to reverse engineer in order to find out the ways the malware sample attempts to trick less tech-savvy users into thinking it is legitimate software. Thirdly, the method by which the malware was packed and unpacks itself is interesting to a novice reverse engineer and warrants further analysis. Preliminary analysis shows it was packed using UPX, but the exact way it is copied and loaded is unknown at this time. Lastly, there was no listing or technical description of the commands an attacker could give to *Sakurel* to carry out on the infected host. Therefore, at minimum I would like cover the key commands that could be used to conduct cyberespionage, namely those that were implemented for the purpose of exfiltrating blueprints, design documents, and other high-value data in the aerospace industry. With time allowing, auxiliary functions could also be reverse engineered as well, but a preference would be given to writing a command and control server for *Sakurel* that could control data exfiltration from a remote shell. As this sample was primarily used for cyberespionage, reverse engineering its core functions well enough to write a command and control server would provide a demonstration of a good analysis of the *Sakurel* sample.

REFERENCES

[1] "SHA256: 6b6e92be036b1a67c383d027bafc7eb63cf515006bb3b3c6ca362a2332542801", *VirusTotal.com*, https://www.virustotal.com/en/file/6b6e92be036b1a67c383d027bafc7eb63cf515006bb3b3c6ca362a2332542801/analysis/ (accessed Jan 30, 2017).

[2] K. Bandla and D. Wescott, "APTnotes.csv", *APTnotes*, Dec. 2015, https://github.com/aptnotes/data/blob/master/APTnotes.csv (accessed Jan 30, 2017).

[3] J. DiMaggio, "The Black Vine Cyberespionage Group", *Symantec Corporation*, Ver. 1.1, Jul. 2015, https://app.box.com/s/0ahidgtzecyx94hgvxoai9kmu5r6yw49 (accessed Jan 30, 2017).