

Introduction/Summary

Many times, malware isn't necessarily what it seems. Throughout this report, I learned that the hard way. During phase one of my approach to analyzing this malware, I concluded that it was used to send sensitive information to a server and possibly leak it to the world or use it as leverage. When I started on Phase 2, I immediately noticed that this assumption had to be wrong. I found pieces of evidence to suggest the malware was actually trying to establish a remote connection to a server, then execute commands as an administrator to install other programs.

Malware Identification

MD5: 44DA2002082622C693B7C28F217E3F98

SHA-1: C1999E1E8B0406F781FD51E18B0E10C1D44378AB

Name: jrusse31.exe

Compilation Time: 2013/01/20 17:33:27

3 Section Headers: .text, .rdata, and .data.

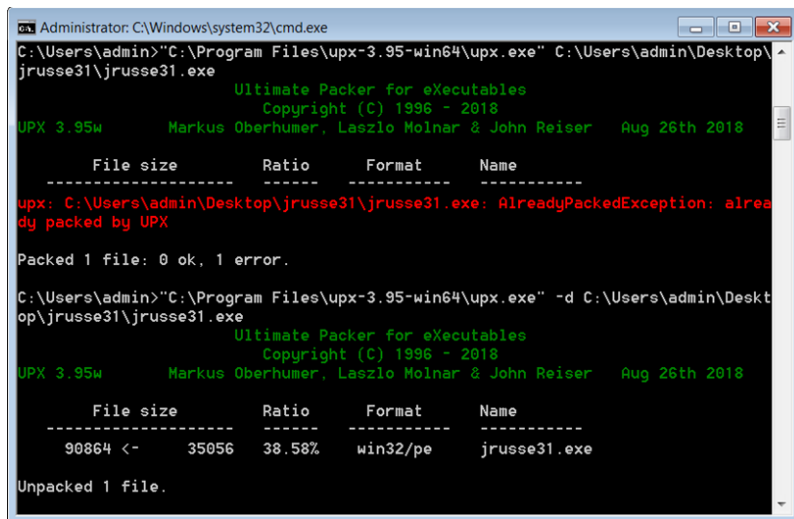
Phase 1 – Basic Analysis

For Phase 1 of the malware analysis, I will be looking at pieces of the malware statically and dynamically. I will achieve this without looking at the code, as that will be handled in Phase 2. In this phase, I want to get a basic understanding of the structure of the malware. This understanding will help me infer what the application's main goal is, but I will not be able to confirm these findings until Phase 2.

Packing

The first thing I did to analyze this malware, was determine if it was packed. When I unzipped the file into my virtual machine, I noticed that the file was only 34 kb. This seemed like a suspiciously

small size for a malware application, so I decided to unpack it using UPX. (Figure 1-1) After unpacking the malware, it changed from 34 kb to 89 kb. That's over twice the size!



```
Administrator: C:\Windows\system32\cmd.exe
C:\Users\admin>"C:\Program Files\upx-3.95-win64\upx.exe" C:\Users\admin\Desktop\jrusse31\jrusse31.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2018
UPX 3.95w Markus Oberhumer, Laszlo Molnar & John Reiser Aug 26th 2018

-----
File size      Ratio      Format      Name
-----
upx: C:\Users\admin\Desktop\jrusse31\jrusse31.exe: AlreadyPackedException: already packed by UPX

Packed 1 file: 0 ok, 1 error.

C:\Users\admin>"C:\Program Files\upx-3.95-win64\upx.exe" -d C:\Users\admin\Desktop\jrusse31\jrusse31.exe
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2018
UPX 3.95w Markus Oberhumer, Laszlo Molnar & John Reiser Aug 26th 2018

-----
File size      Ratio      Format      Name
-----
90864 <-      35056      38.58%      win32/pe      jrusse31.exe

Unpacked 1 file.
```

Figure 1-1

Strings

The next thing I did when analyzing this malware, was use strings.exe to export the strings that were used in the program. After looking through these strings, I found a lot of useless characters that meant nothing to the human eye, until I scrolled a little further. The first thing I found was a list of words. These words appear to be a password dump used to gain access to a system through a dictionary attack. (Figure 1-2)

```
HZS
123456
password
phpbb
qwerty
12345
jesus
12345678
1234
abc123
letmein
test
love
123
password1
- - -
```

Figure 1-2

That wasn't all I found in the strings, however. I also found a long list of items that may be used as variables to collect information from the system. These include Items like Favorites.dat, Password, Username, port, history.dat, and many more. After looking at these strings, I started to think about what the purpose of this malware was. These strings tell me that it may be trying to gather information, but further investigation is needed.

DLLs and API Calls

After analyzing the strings and trying to wrap my head around what this application was trying to do, I decided to look at the linked libraries for any new pieces of evidence using Dependency walker, and CFF explorer. I found that it called 8 libraries: KERNEL32.dll, advapi32.dll, ole32.dll, shlwapi.dll, user32.dll, userenv.dll, wininet.dll, and wsock32.dll.

First, let's discuss the imports from KERNEL32.dll. This library is used in almost every Windows program, so the fact that it's being called doesn't really tell me anything. This library is commonly used to access memory, files, and hardware. What's interesting about this link are the functions it calls. If we look at the imports from this library, we see functions such as CreateFileA, ReadFile, CloseHandle, and WriteFile. This tells me that the program is creating a file to store information in. What information is this storing? We'll look into that in phase 2. I also noticed that the malware calls the CreateFileMapping and MapViewOfFile functions, which may indicate that it's trying to open another process.

The next library I found interesting is the advapi32.dll library. This library is used to access and change the registry. Inside this library, the malware opens registry keys, changes registry keys, and gathers information – such as the username – from certain registry keys.

Ole32.dll doesn't show any odd calls that a program wouldn't normally call, but shlwapi.dll does. Shlwapi32.dll This is used to store information about the shell. In this library, StrStrIA, StrRChrIA,

StrToIntA, and more string functions are called. These functions try to find and alter strings. It's possible that the application is trying to find information on the user using this search method.

The last DLL's I will talk about go hand in hand: Wininet.dll, and wsock32.dll. These are both networking libraries, with slight differences between the two. Wsock32 is used for basic networking tasks, such as connecting to the internet or network. Under this library, a few functions imply that the application is trying to send information to a server. These functions include client side functions, such as socket, connect, send, select, and recv. Wininet.dll, on the other hand, is able to use high-level networking protocols, such as HTTP and FTP. Only two functions are called from this library: InternetCrackURLA and InternetCreateURLA. These both try to create a url for accessing files on the system. Perhaps it's trying to access the file it created earlier?

Dynamic Analysis

In order to perform a dynamic analysis, I used process monitor and Regshot. Process monitor will tell me if the application started/stopped any processes, and Regshot will tell me what registry keys the application changed, as we saw in the advapi31.dll. According to Regshot, 4 keys were changed in the registry.

- HKLM\SOFTWARE\Wow6432Node\Microsoft\WindowsNT\CurrentVersion\Perflib\CurrentLanguage\Counter
- HKU\S-1-5-21-1590843950-3972615591-659575068-1000\Software\Microsoft\Windows\CurrentVersion\Action Center\Checks\{11CD958A-C507-4EF3-B3F2-5FD9DFBD2C78}.check.101\CheckSetting
- HKU\S-1-5-21-1590843950-3972615591-659575068-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\HRZR_PGYFRFFVBA

- HKU\S-1-5-21-1590843950-3972615591-659575068-1000\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBFF5CD-ACE2-4F4F-9178-9926F41749EA}\Count\{6Q809377-6NS0-444O-8957-N3773S02200R}\Ertfubg-1.9.0\Ertfubg-k64-NAFV.rkr

I'm not quite sure why it's changing these registry keys, but it seems to be trying to alter user settings. When running process monitor, I found that the program is trying to open a lot of keys that may have information about the user. For instance, I noticed the program trying to open keys from the Mozilla Firefox registry key, and this may be to look at the user's browser history.

Phase 1 Conclusion:

After looking at the evidence mentioned above, I have a suspicion that the program's goal is to gather information about the user and send it to a server. I suspect this due to the fact that there were a lot of instances where the program wanted to find information about the system, such as in the registry keys and the DLL calls. The most suspicious aspect is that the program tried to reach out to a server to send information. In the next phase, I will either confirm my suspicions or be proven wrong.

Phase 2 – Extended Analysis

In order to confirm my suspicions, I needed to look at the code of the application. Using a disassembly program called IDA, I was able to look through the imports/functions the program calls to try to find out more. The first thing I thought to do was to search for the phrase "password", as we saw in the strings found in Phase 1. Unfortunately, this did not show the results I was hoping for. I did, however, find the password dump in the .data section. This showed me where all those passwords were

being used. Figure 2-1 shows a small portion of these passwords.

```
.data:004140D4 aFootball      db 'football',0
.data:004140DD aAngel        db 'angel',0
.data:004140E3 aJesus1       db 'jesus1',0
.data:004140EA a123123      db '123123',0
.data:004140F1 aWhatever    db 'whatever',0
.data:004140FA aFreedom     db 'freedom',0
.data:00414102 aKiller      db 'killer',0
.data:00414109 aAsdf        db 'asdf',0
.data:0041410E aSoccer      db 'soccer',0
.data:00414115 aCupcake     db 'cupcake',0
```

Figure 2-1

This peaked my curiosity as to what these passwords were being used for. I tried to find any evidence of my suspicions from before, but I unfortunately had no luck. After searching for quite a while, I found that the program tried to install other programs, which would explain some of the registry changes mentioned in phase 1. (Figure 2-2)

```
.text:00408B68      push     offset aWiseftpsrvsBin ; "wiseftpsrvs.bin"
.text:00408B6D      push     offset aAcebit       ; "\\AceBIT"
.text:00408B72      push     [ebp+arg_0]          ; int
.text:00408B75      call     sub_403FA2
.text:00408B7A      push     offset aSoftwareAcebit ; "Software\\AceBIT"
.text:00408B7F      push     hKey                  ; hKey
.text:00408B85      push     [ebp+arg_0]          ; int
.text:00408B88      call     sub_40891D
.text:00408B8D      push     offset aSoftwareAcebit ; "Software\\AceBIT"
.text:00408B92      push     80000002h             ; hKey
.text:00408B97      push     [ebp+arg_0]          ; int
.text:00408B9A      call     sub_40891D
.text:00408B9F      push     offset aSoftwareClasse ; "SOFTWARE\\Classes\\TypeLib\\{CB1F2C0F-8"...
.text:00408BA4      push     80000002h             ; hKey
```

Figure 2-2

After this finding, I concluded that the password dump was used to grant the program administrative control of the windows command prompt to run programs. My suspicions were incorrect. Initially, I thought the program was trying to gather information about the user's habits, which is not entirely wrong. The purpose of this reconnaissance was to gain access to the system. I also noticed that these registry changes occurred after the connection to the server was initiated, and that it tried to run abcd.bat. (Figure 2-3) The fact that the program tried to run this, tells me that the goal of the program

was to run commands as an administrator, not to solely gather information of the victim.

```
data:004173E3 aStatusImportOk db 'STATUS-IMPORT-OK',0
data:004173F4 aShellexecutea db 'ShellExecuteA',0
data:00417402 aAbcdBat db 'abcd.bat',0
```

Figure 2- 3

If you look above the abcd.bat line, you will also see that 'ShellExecuteA' is included. This should only be included if the program is trying to gain remote access and execute commands.

Conclusion and Recommendations

Initially, I thought the malware was simply gathering information to send to a server. After running an extended analysis, however, I found that this was not the program's intent. Yes, it did gather information, but that information was only used to achieve the program's further goals. The main reason I came to the conclusion I did, was that the server connection was established before the commands were run. This implies that the program wanted to connect the attackers to the machine remotely and install programs. If an organization is affected by this malware, I would suggest analyzing network traffic for any suspicious packets that go to the servers specified in the programs.

I learned a lot during this project. The first thing being that your assumptions from a basic analysis may be wildly off and further investigation may be needed to reach a final conclusion. I also learned that malware can sometimes be very complicated and use processes as a stepping stone for achieving bigger goals. I struggled frequently as well. During Phase 1, it seemed like all of my leads were dead ends, and I was getting nowhere. It wasn't until phase 2 that I was able to see the malware for what it is, a shell for attackers to come in and install whatever they want onto the victims machine.