Redline Stealer - Writeup

Redline Stealer

Did this for a course, this is the writeup I submitted there. Thought I'd post it on here, enjoy.

Short Summary

Malware Name: `Redline Stealer`

Sample Hash: `d50203cdee12951fa87f5e2eb1428dde3dedb16257db97ef4ef27201a9267c85`

URL: `https://bazaar.abuse.ch/sample/d50203cdee12951fa87f5e2eb1428dde3dedb16257db97ef4ef27201a9267c85/`

This report summarizes the findings regarding the Redline Stealer sample with the above SHA256 hash. Redline Stealer is an information stealer, which gathers information about the infected system, for instance saved browser login credentials, system configuration and installed programs among others [1]. The information is then exfiltrated to the attackers in control of the configured C2 server. In the case of this particular sample, an initial stage contains a packed second stage. The initials stage unpacks this second stage, which is the stealer component, to %USER%/AppData/Local/Temp and executes it. As a result, a connection to the C2 is established where the malware sends all collected information after it is obtained from the system.

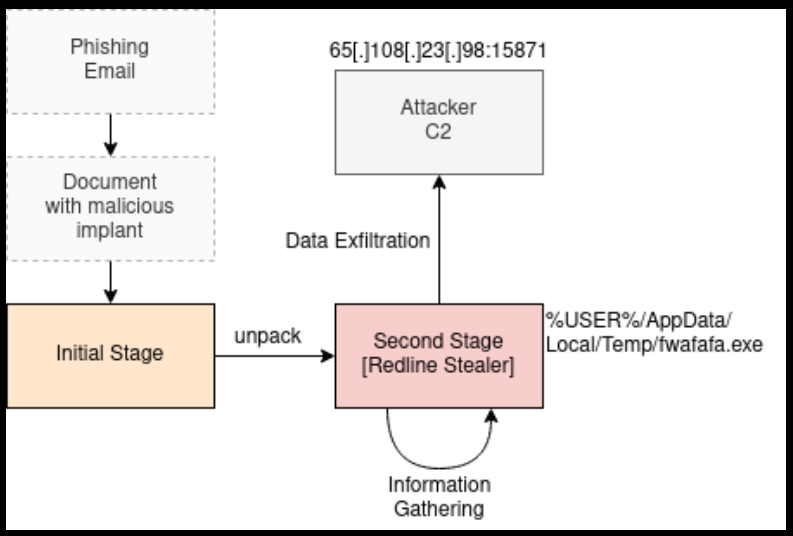
High Level Technical Summary

Although the infection vector by which the malware reaches the victim system is not discernable via the given sample alone, past Redline Stealer campaigns have used phishing with a malicious document [2] or disguised the malware as software cracks [3]. In the case of this sample,

the exectable with the initial stage unpacks the seconds stage as "fwafafa.exe" to "%USER%/AppData/Local /Temp" and executes it.

The second stage is Redline Stealer, which exfiltrates the collected information to the attacker IP "65[.]108[.]23[.]98" via port 15871. As the C2 is not active anymore, this can only be ascertained from the static and dynamic analysis of the sample but not be corroborated by a real-world test.

The below block diagram summarizes the above.



Composition

The two parts of the malware are:

- the initial stage SHA256: d50203cdee12951fa87f5e2eb1428dde3dedb16257db97ef4ef27201a9267c85
- the second stage, Redline Stealer SHA256: f4bf9e7bebb1eb9645a4c15d8575837f586fec95f729359d9b9c550051fcddef

Basic Static Analysis

In a first step, the initial stage of the malware is examined. An analysis of the strings contained in the malware provides the following selection of items of interest:

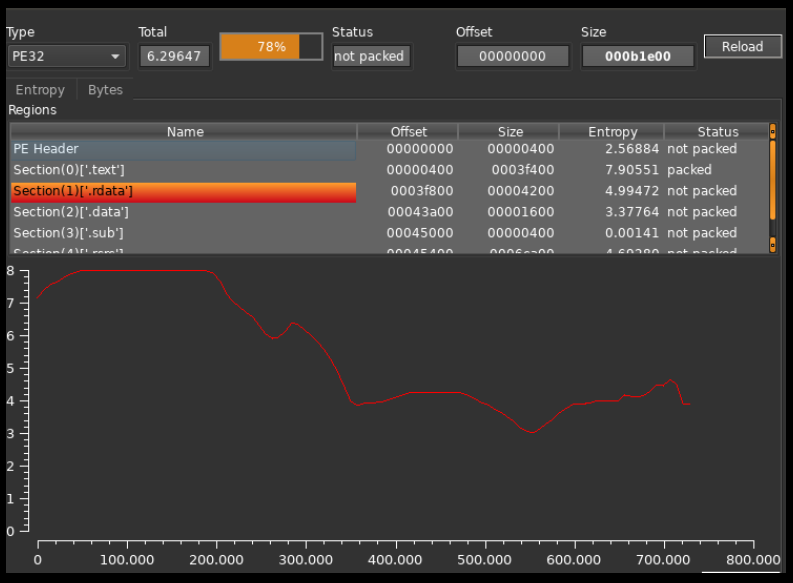
```
C:\yoxu\dexu\zamuluku\xekecojezox.pdb
Vowoyezelu tahuvuputif laje
VirtualAlloc
```

```
VirtualProtect
GetProcAddress
LoadLibraryA
...
```

The two first strings may allow identification of the malware via automated tools, while the function names give clues of its inner workings.

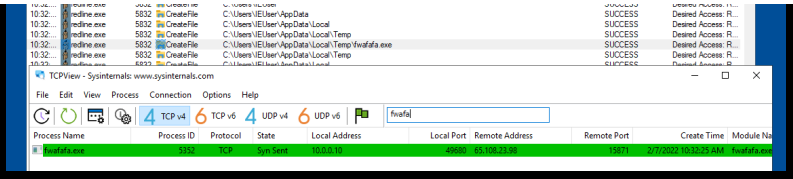
An initial analysis shows the sample is a PE32 executable, compiled with Microsoft Visual C/C++. Examining the imports, only KERNEL32.dll is imported, however, via GetProcAddress and LoadLibraryA, further components may be loaded at runtime.

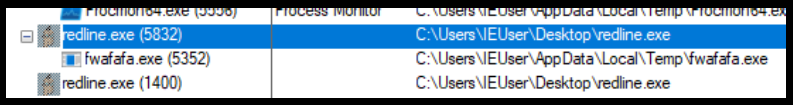
The entropy shown below and the fact that VirtualProtect / VirtualAlloc are used suggest this executable may contain a further stage which was confirmed in the course of the analysis. The sections do not indicate a standard packer such as UPX is in use, neither do the comparison of raw and virtual section sizes.

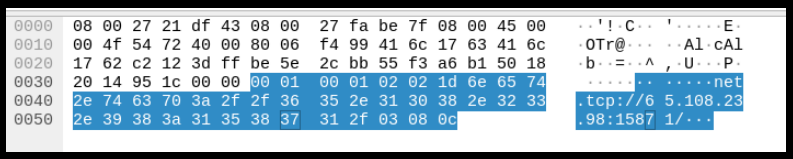


Basic Dynamic Analysis

The malware drops the file `fwafafa.exe` to `%USER%\AppData\Local\Temp`. This executable attempts to contact the IP address `65[.]108[.]23[.]98` via port 15871. Wireshark shows the indicator net.tcp://65.108.23.98:15871



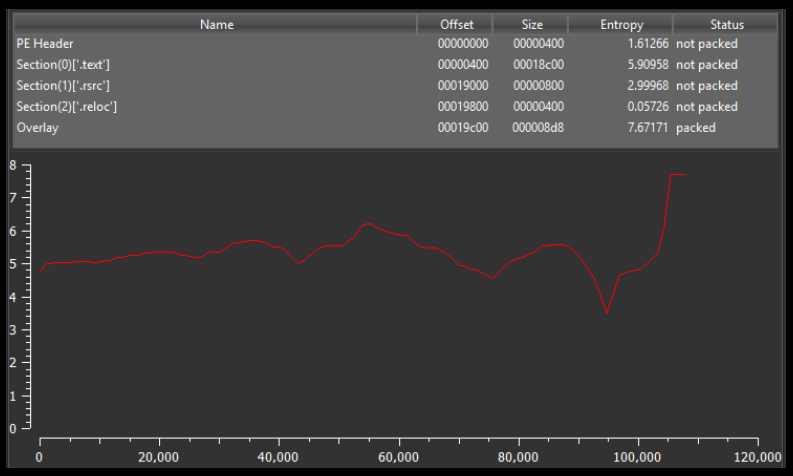




Advanced Static and Dynamic Analysis

Examination of the Second Stage

The unpacked second stage `fwafafa.exe` has a lower overall entropy than the packed initial stage.



This already hints at this being the malware's payload. Closer examination reveals this is a .NET PE32 executable, thus it is loaded into dnSpy to perform a deeper analysis.

Using the debugger to reach the entry point of the application, the first step in execution is the decryption of the configuration shown below.

```
public static class Arguments
{
    // Token: 0x04000003 RID: 3
    public static string IP = "DwQyGSgmAForBiwIDQQMWCoYBF8oKAYK";

    // Token: 0x04000004 RID: 4
    public static string ID = "G1wxGT8qCwYFLVhP";

    // Token: 0x04000005 RID: 5
    public static string Message = "";

    // Token: 0x04000006 RID: 6
    public static string Key = "Angler";

    // Token: 0x04000007 RID: 7
    public static int Version = 1;
}
```

Decryption is realized via a simple XOR cipher, in combination with the hard-coded key.

```
// Token: 0x0600002A RID: 42 RVA: 0x00003E8C File Offset: 0x0000228C
public static string Xor(string input, string stringKey)
{
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = 0; i < input.Length; i++)
    {
        stringBuilder.Append(Convert.ChangeType((int)(input[i] ^ stringKey[i % stringKey.Length]), TypeCode.Char));
    }
    return stringBuilder.ToString();
}
```

The `IP` variable decrypts to `65[.]108[.]23[.]98:15871`.

Capabilities

This section describes the key capabilities of the information stealer.

Enumeration of Installed Browsers

Browsers are enumerated via the registry, as shown in the below screenshot. The `StartMenuInternet` subkey is where browsers add an entry in order to be displayed as internet clients in the start menu.

```
RegistryKey registryKey = Registry.LocalMachine.OpenSubKey("SOFTWARE\\WOW6432Node\\Clients\\StartMenuInternet");
if (registryKey == null)
{
    registryKey = Registry.LocalMachine.OpenSubKey("SOFTWARE\\Clients\\StartMenuInternet");
}
```

Enumeration of Installed Programs

Installed software is also enumerated via the registry, here the key `SOFTWARE\Microsoft\Windows \CurrentVersion\Uninstall` is used. The malware authors attempt to make analysis harder by storing the key as a character array.

```
        's',
        'i',
        'o',
        'n',
        '\\',
        'U',
        'n',
        'i',
        'n',
        's',
        't',
        'a',
        'l',
        'l'
    });
```

In a later method, the directories

- Windows
- Program Data
- Program Files
- Program Files (x86)

are checked for directories containing the (obfuscated) strings

- Login Data
- Web Data
- Cookies

```
foreach (string text in FileCopier.FindPaths(baseDirectory, 1, 1, new string[]
{
    "LEnvironmentogiEnvironmentn DatEnvironmenta".Replace("Environment", string.Empty),
    "WSystem.Texteb DatSystem.Texta".Replace("System.Text", string.Empty),
    "CoCryptographyokieCryptographys".Replace("Cryptography", string.Empty)
}))
```

Enumeration of Antivirus

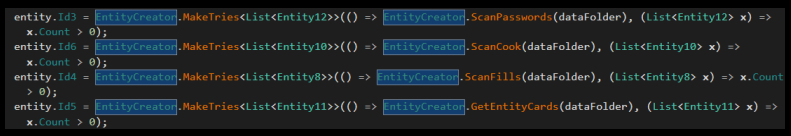
A further method searches for the strings

```
AntivirusProduct
AntiSpywareProduct
FirewallProduct
```

using a `ManagementObjectSearcher` class to enumerate security software installed on the system. This is explained further in [4].

Stealing Web Browser Data

Using the information about the identified browsers, the malware collects login data, autofill information, cookies and targets the string *credit_cards* as well.



Encoded Wallet Names

The malware contains a long base64 encoded string (approx. 15k characters) which was decoded using a python script, the resulting strings are shown below.

```
ffnbelfdoeiohenkjibnmadjiehjhajb|YoroiWalle
t
ibnejdfjmmkpcnlpebklmnkoeoihofec|Tronlink
jbdaocneiiinmjbjlgalhcelgbejmnid|NiftyWalle
t
nkbihfbeogaeaoehlefnkodbefgpgknn|Metamask
afbcbjpbpfadlkmhmclhkeeodmamcflc|MathWallet
hnfanknocfeofbddgcijnmhnfnkdnaad|Coinbase
fhbohimaelbohpjbbldcngcnapndodjp|BinanceCha
in
odbfpeeihdkbihmopkbjmoonfanlbfcl|BraveWalle
t
hpglfhgfnhbgpjdenjgmdgoeiappafln|GuardaWall
et
blnieiiffboillknjnepogjhkgnoapac|EqualWalle
t
cjelfplplebdjjenllpjcblmjkfcffne|JaxxxLiber
ty
fihkakfobkmkjojpchpfgcmhfjnmnfpi|BitAppWall
et
kncchdigobghenbbaddojjnnaogfppfj|iWallet
amkmjjmmflddogmhpjloimipbofnfjih|Wombat
fhilaheimglignddkjgofkcbgekhenbh|AtomicWall
et
nlbmnnijcnlegkjjpcfjclmcfggfefdm|MewCx
nanjmdknhkinifnkgdcggcfnhdaammmj|GuildWalle
t
nkddgncdjgjfcddamfgcmfnlhccnimig|SaturnWall
et
fnjhmkhhmkbjkkabndcnnogagogbneec|RoninWalle
```

```
t
aiifbnbfobpmeekipheeijimdpnlpgpp|TerraStati
on
fnnegphlobjdpkhecapkijjdkgcjhkib|HarmonyWal
let
aeachknmefphepccionboohckonoeemg|Coin98Wall
et
cgeeodpfagjceefieflmdfphplkenlfk|TonCrystal
pdadjkfkgcafgbceimcpbkalnfnepbnk|KardiaChai
n
bfnaelmomeimhlpmgjnjophhpkkoljpa|Phantom
fhilaheimglignddkjgofkcbgekhenbh|Oxygen
mgffkfbidihjpoaomajlbgchddlicgpn|PaliWallet
aodkkagnadcbobfpggfnjeongemjbjca|BoltX
kpfopkelmapcoipemfendmdcghnegimn|LiqualityW
allet
hmeobnfnfcmdkdcmlblgagmfpfboieaf|XdefiWalle
t
lpfcbjknijpeeillifnkikgncikgfhdo|NamiWallet
dngmlblcodfobpdpecaadgfbcggfjfnm|MaiarDeFiW
allet
bhghoamapcdpbohphigoooaddinpkbai|Authentica
tor
```

In combination with further enumeration in the Application Data directory, this suggests the stealer also attempts to collect cryptocurrency wallet data.

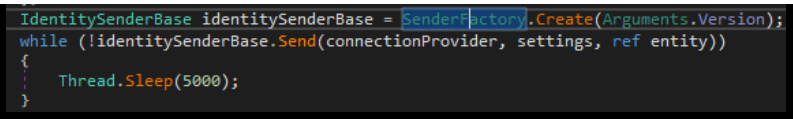
Further Points

The malware also contains references to the applications:

- Discord
- Telegram
- Filezilla

Exfiltration

Exfiltration takes place via port 15871 to the encoded IP address. After a connection to the attacker system is established, the SenderFactory.Create method, shown below, collects all information as described in the previous sections. This data is then sent to the attackers.



Appendix

IOCs

Host Based IOCs

Initial Stage (SHA256 Hash):

```
d50203cdee12951fa87f5e2eb1428dde3dedb16257d
b97ef4ef27201a9267c85
```

Second Stage (Filename and SHA256 Hash):

```
fwafafa.exe
f4bf9e7bebb1eb9645a4c15d8575837f586fec95f72
9359d9b9c550051fcddef
```

Network Based IOCs

```
65.108.23.98:15871
net.tcp://65.108.23.98:15871
```

YARA Rules

Rule for Initial Stage

```
rule
redline_d50203cdee12951fa87f5e2eb1428dde3de
db16257db97ef4ef27201a9267c85 {

    meta:
        last-updated: "2022-02-08"
        author: "secret"
        description: "yara rule for Redline
Stealer
(SHA256:d50203cdee12951fa87f5e2eb1428dde3de
db16257db97ef4ef27201a9267c85)"

    strings:
        // $VARNAME = "VALUE" TYPE

        // add strings from the malware
        $string1 = "Vowoyezelu tahuvuputif
```

```
laje" ascii
        $pdb = "C:\yoxu\dexu\zamuluku
\xekecojezox.pdb" ascii

    condition:

        // is a PE, so MZ is at position 0
of the file
        $PE_MAGIC_BYTE at 0 and
        // AND must contains string1 and
string2
        ($string1 and $pdb)
}
```

References

[1] https://malpedia.caad.fkie.fraunhofer.de/details/win.redline_stealer

[2] https://www.fortinet.com/blog/threat-research/omicron-variant-lure-used-to-distribute-redline-stealer

[3] https://asec.ahnlab.com/en/30445/

[4] https://social.msdn.microsoft.com/Forums/vstudio/en-US/d555f390-dd75-4604-b653-df0a9f4c2fa3/wmi-retrieving-antivirus-product-info-with-wmi-and-c-i-need-help-cant-find-the-solution?forum=csharpgeneral

<= back