BRIEF: Raccoon Stealer Version 2.0

Archived: 2026-04-05 17:59:54 UTC

ZeroFox Intelligence has observed the following information as of June 28, 2022, and has released the following.

Executive Summary

On June 4, 2022, ZeroFox Intelligence discovered a then-unknown information stealer being distributed by ProCrackerz, a website distributing fake software cracks and key generators (keygens). The earliest known instance of this information stealer observed by ZeroFox Intelligence was a sample uploaded to VirusTotal on April 19, 2022. Twitter user @James_inthe_box suggested the name “Recordbreaker” for it based on the use of “record” as the User-Agent string in each sample. In May 2022, logs for sale with “Raccoon Stealer V2.0” branding were discovered that matched what ZeroFox Intelligence was observing with Recordbreaker. Due to this and multiple other private confirmations, ZeroFox Intelligence asserts with MEDIUM confidence that Raccoon Stealer has returned and that Recordbreaker is actually Raccoon Stealer version 2.0.

Details

Raccoon Stealer version 2.0 is capable of targeting Chromium and Mozilla-based browsers by looking for well-known file names in specific directories. For Chromium-based browsers, threat actors using Raccoon Stealer 2.0 have the ability to specify a list of Chrome extension IDs and associated files as well. In ZeroFox Intelligence’s observations, these consisted entirely of cryptocurrency extensions. Raccoon Stealer 2.0 attempts to collect credentials, cookies, autofill data, credit cards, and data associated with specified Chrome extensions. For Mozilla browsers such as Firefox, only credentials, cookies, and autofill data were targeted. Other applications like Telegram and specific cryptocurrency applications can be targeted as well. To ensure that all cryptocurrency wallets are collected, a separate function exists solely to collect “wallet.dat” files. For any applications or files without specific support, a generic “grbr_” function exists to allow actors to specify files by path and a name or pattern.

Technical Analysis

ZeroFox Intelligence first discovered Raccoon Stealer version 2.0 disguised as a crack for Microsoft Office on the ProCrackerz website. Clicking on any of the download links redirected the viewer through various advertisements and click trackers until they were eventually shown a set of directions and a Discord CDN link hosting the fake crack. The download links on ProCrackerz change regularly as the Discord links are removed.

Figure 1. ProCrackerz listing for a Microsoft Office crack

Source: ZeroFox Intelligence

Figure 2. Instructions on downloading a compressed Raccoon Stealer version 2.0 sample

Source: ZeroFox Intelligence

The compressed files are small in size but inflate to hundreds of megabytes when decompressed. This is due to the samples being padded with large amounts of repeating bytes.

Samples ZeroFox Intelligence observed distributed in this way were obfuscated or packed; the unique string "edinayarossiya" was visible and used to pivot to other samples uploaded to VirusTotal. This allowed ZeroFox Intelligence to download much smaller (~56KB) unprotected samples and greatly sped up our analysis. Translated, Edinaya Rossiya means "United Russia," which is currently the largest political party in Russia. Later analysis of the string decryption routine determined this is an encryption key for the protected strings used by the stealer.

Raccoon Stealer version 2.0 begins by importing all of the Windows API calls it needs (and some it does not). Importing API calls at runtime is a common tactic used by malware to avoid adding them to the import table to be used as a signature.

```
00401000  void resolve_winapi_imports()

00401000  {
0040100b      HINSTANCE h_kernel32_dll = LoadLibraryW("kernel32.dll");
00401016      if (h_kernel32_dll != 0)
00401014      {
0040102b          FARPROC eax = GetProcAddress(h_kernel32_dll, "LoadLibraryW");
00401032          LoadLibraryW = eax;
00401037          HINSTANCE h_shlwapi_dll = eax(0x40c08c);  // {"Shlwapi.dll"}
00401046          int32_t h_ole32_dll = LoadLibraryW("Ole32.dll");
00401056          int32_t h_wininet_dll = LoadLibraryW("WinInet.dll");
00401066          int32_t h_advapi32_dll = LoadLibraryW("Advapi32.dll");
00401076          int32_t h_user32_dll = LoadLibraryW("User32.dll");
00401086          int32_t h_crypt32_dll = LoadLibraryW("Crypt32.dll");
00401096          int32_t h_shell32_dll = LoadLibraryW("Shell32.dll");
0040109f          LoadLibraryW("Bcrypt.dll");
004010ad          FARPROC eax_1 = GetProcAddress(h_kernel32_dll, "GetProcAddress");
004010b8          GetProcAddress = eax_1;
004010c4          GetCurrentProcess = eax_1(h_kernel32_dll, 0x40c15c);  // {"GetCurrentProcess"}
004010d6          GetEnvironmentVariableW = GetProcAddress(h_kernel32_dll, "GetEnvironmentVariableW");
004010e8          GetFileSize = GetProcAddress(h_kernel32_dll, "GetFileSize");
004010fa          GetDriveTypeW = GetProcAddress(h_kernel32_dll, "GetDriveTypeW");
0040110c          GetLastError = GetProcAddress(h_kernel32_dll, "GetLastError");
0040111e          GetLocaleInfoW = GetProcAddress(h_kernel32_dll, "GetLocaleInfoW");
00401130          GetLogicalDriveStringsW = GetProcAddress(h_kernel32_dll, "GetLogicalDriveStringsW");
0040113d          GetModuleFileNameW = GetProcAddress(h_kernel32_dll, "GetModuleFileNameW");
00401154          GetSystemWow64DirectoryW = GetProcAddress(h_kernel32_dll, "GetSystemWow64DirectoryW");
00401166          GetUserDefaultLocaleName = GetProcAddress(h_kernel32_dll, "GetUserDefaultLocaleName");
00401178          GetTimeZoneInformation = GetProcAddress(h_kernel32_dll, "GetTimeZoneInformation");
```

Figure 3. Raccoon Stealer version 2.0 resolves Windows API calls at runtime

Source: ZeroFox Intelligence

Afterwards, all protected strings are base64 decoded and RC4 decrypted as shown in **Figure 4**.

```
decrypt_strings:
00404036  55              push    ebp {__saved_ebp}
00404037  8bec            mov     ebp, esp {__saved_ebp}
00404039  51              push    ecx {var_8}
0040403a  8365fc00        and     dword [ebp-0x4 {var_8}], 0x0
0040403e  8d55fc          lea     edx, [ebp-0x4 {var_8}]
00404041  56              push    esi {var_c}
00404042  57              push    edi {var_10}
00404043  b954c84000      mov     ecx, data_40c854  {"fVQMox8c"}
00404048  e8b9d7ffff      call    base64_decode
0040404d  bf44c84000      mov     edi, string_rc4_key  {"edinayarossiya"}
00404052  8d4dfc          lea     ecx, [ebp-0x4 {var_8}]
00404055  57              push    edi {var_14}  {string_rc4_key, "edinayarossiya"}
00404056  51              push    ecx {var_8} {var_18}
00404057  be28e24000      mov     esi, data_40e228
0040405c  50              push    eax {var_1c}
0040405d  8bce            mov     ecx, esi  {data_40e228}
0040405f  e8e2460000      call    rc4_decrypt
00404064  8d55fc          lea     edx, [ebp-0x4 {var_8}]
00404067  a3f8eb4000      mov     dword [data_40ebf8], eax
```

Figure 4. Strings are protected by RC4 encryption and base64 encoding

Source: ZeroFox Intelligence

The RC4 key “edinayarossiya” was consistent across most samples, though some also used “credit19” instead. ZeroFox Intelligence is currently unsure if this is specified by each actor deploying the stealer or if this is decided by the authors for each build.

Unlike the RC4 key used to decrypt strings, the RC4 key used to decrypt command and control (C2) servers is a fixed length and changes with every sample. Aside from this, C2 servers are protected in much the same way the

other strings are. Up to five C2s can be configured per sample, with each C2 slot hardcoded to be 65 bytes long. Addresses that are shorter than 65 bytes after being encrypted and base64 encoded are padded with spaces.

Figure 5. C2s are RC4 encrypted, base64 encoded, and padded with spaces

Source: ZeroFox Intelligence

The locale on the victim's machine is checked against two locales that can be hard-coded in the binary. ZeroFox did observe a check for a "ru" locale, but the language check does not affect the execution in any way. A second locale was not configured in the samples we observed.

Figure 6. Checking the victim's locale

Source: ZeroFox Intelligence

Raccoon Stealer 2.0 also ensures that only one instance is running at a time by checking and creating a mutex. ZeroFox Intelligence observed this to be "8724643052" with every sample obtained. If it cannot open a handle to the mutex, Raccoon Stealer 2.0 will exit with Error Code 2.

Figure 7. Raccoon Stealer 2.0 ensures that only one instance is running at a time

Source: ZeroFox Intelligence

The victim's security identifier (SID) is checked against the value "S-1-5-18" to determine if the process happens to be running as the SYSTEM or LOCAL SYSTEM user. If so, Raccoon Stealer 2.0 will enumerate the list of running processes on the infected machine.

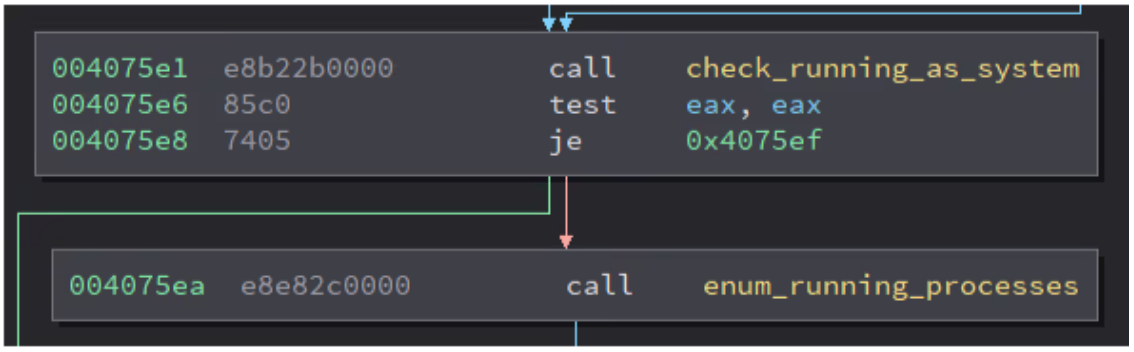


Figure 8. Enumerate running processes if running as SYSTEM

Source: ZeroFox Intelligence

The first real action Raccoon Stealer 2.0 takes is to get the machine GUID and username, which are then sent as an HTTP POST request to the C2. As seen in **Figure 9** below, the GUID and username are sent together in the URL parameter "machineId" separated by a pipe character. The "configId" parameter shown is the RC4 key used to decrypt C2 addresses.

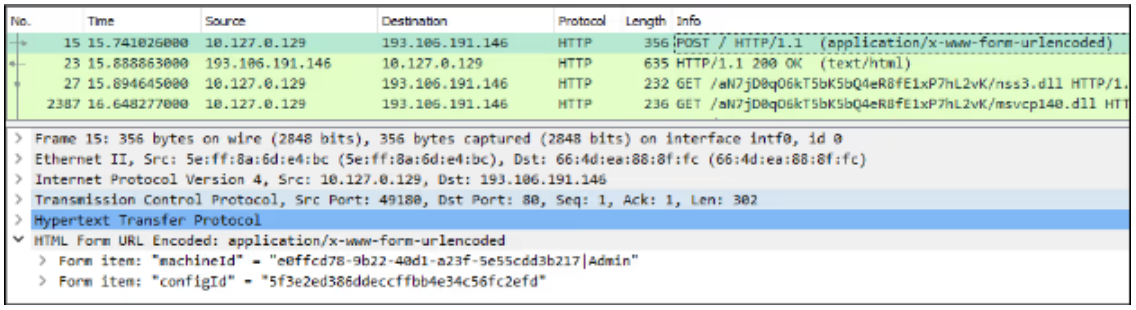


Figure 9. Sending a unique identifier to a Raccoon Stealer C2 server

Source: ZeroFox Intelligence

If the C2 is still available, the server will respond with a simple, newline-separated configuration. If no C2 is available, Raccoon Stealer 2.0 simply exits.

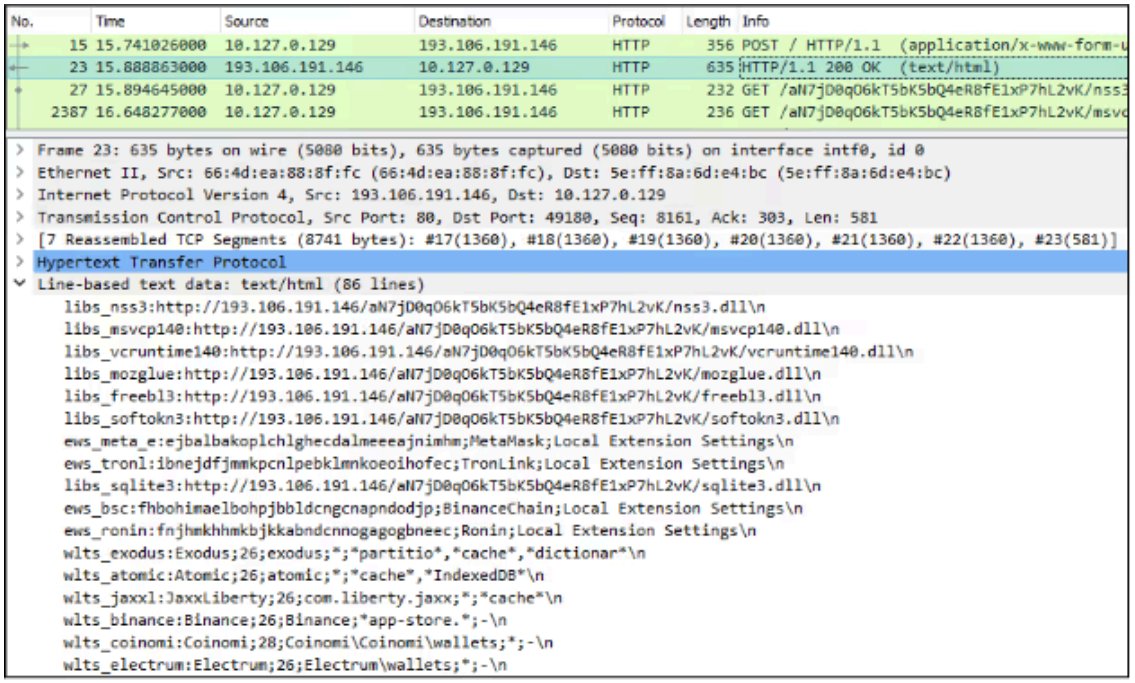


Figure 10. A Raccoon Stealer 2.0 C2 responds with a configuration

Source: ZeroFox Intelligence

There are currently nine options that can be processed from the settings shown in **Figure 10**. A sample configuration returned by one of the C2 servers can be found here.

Option	Description
ews_	Targeted Chrome browser extensions
grbr_	Targeted files to steal
ldr_	A command, DLL, or executable to run
libs_	DLLs to download
scrnsht_	Screenshot file name
sstmnfo_	Send system information to the C2 with this file name and add this template text
tlgrm_	Telegram-specific files and folders to target
token	URL path to POST stolen data
wlts_	Cryptocurrency wallets and associated files and folders to target

DLL files downloaded using the “libs_” option are saved to the AppData\LocalLow directory. Raccoon Stealer 2.0 attempts to add this directory to the PATH environment variable but does not verify if it was successful. In our observations, this actually failed, and the sample continued to run without issue.

The following system information is collected during a run:

- User locale
- System time zone
- Operating system
- System architecture (32-bit or 64-bit)
- CPU core count
- Installed RAM
- Screen resolution
- All display devices (GPUs)
- Installed software and versions

Once each of these functions has run, another POST request is made to /<token>.

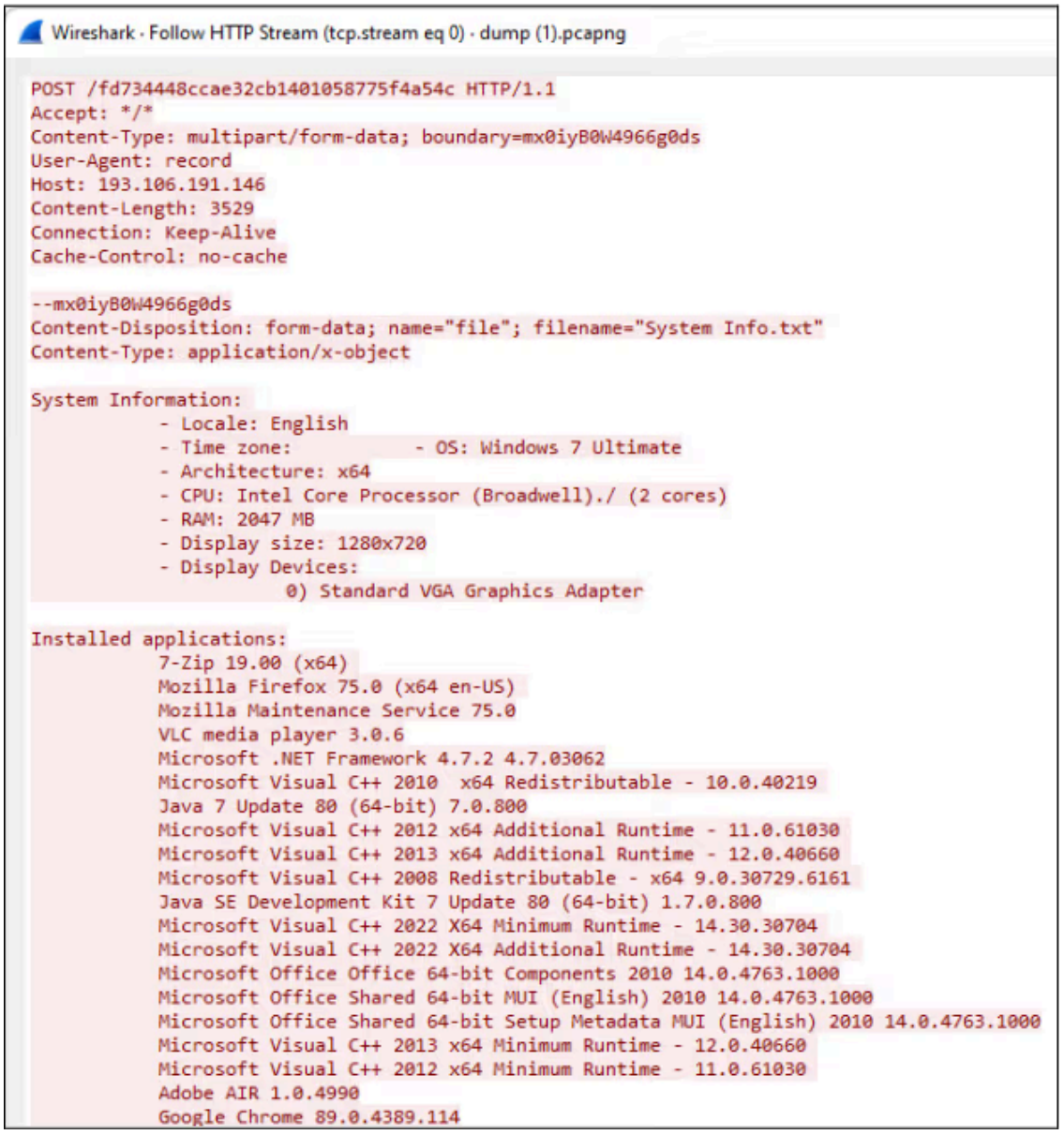


Figure 11. System information being sent to the C2 server

Source: ZeroFox Intelligence

Rather than look for specific browsers, Raccoon Stealer version 2.0 targets any Chromium or Mozilla-based browsers by the name of the directories in which each respective browser stores its data. For Chromium, this is “User Data” while Mozilla/Gecko uses “Profiles.”

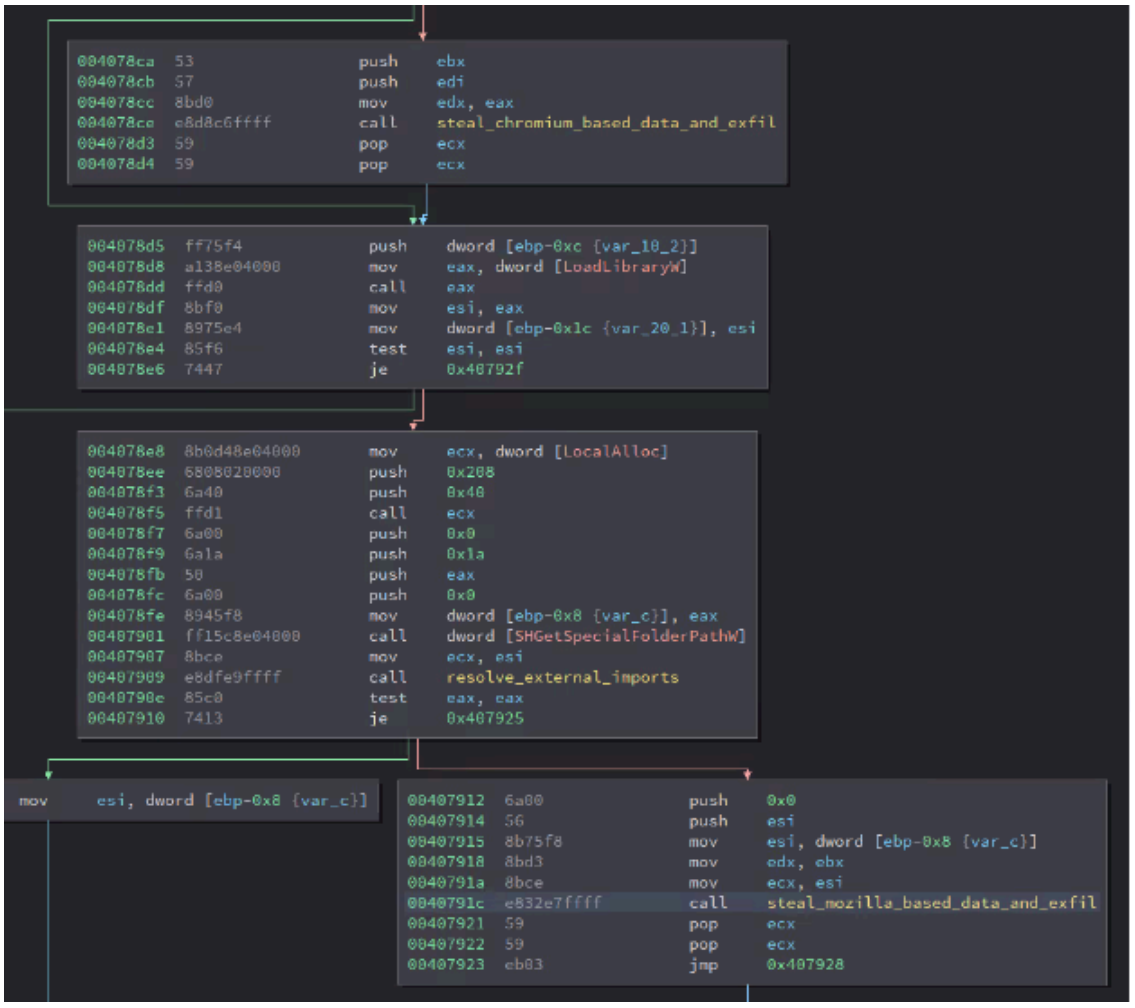
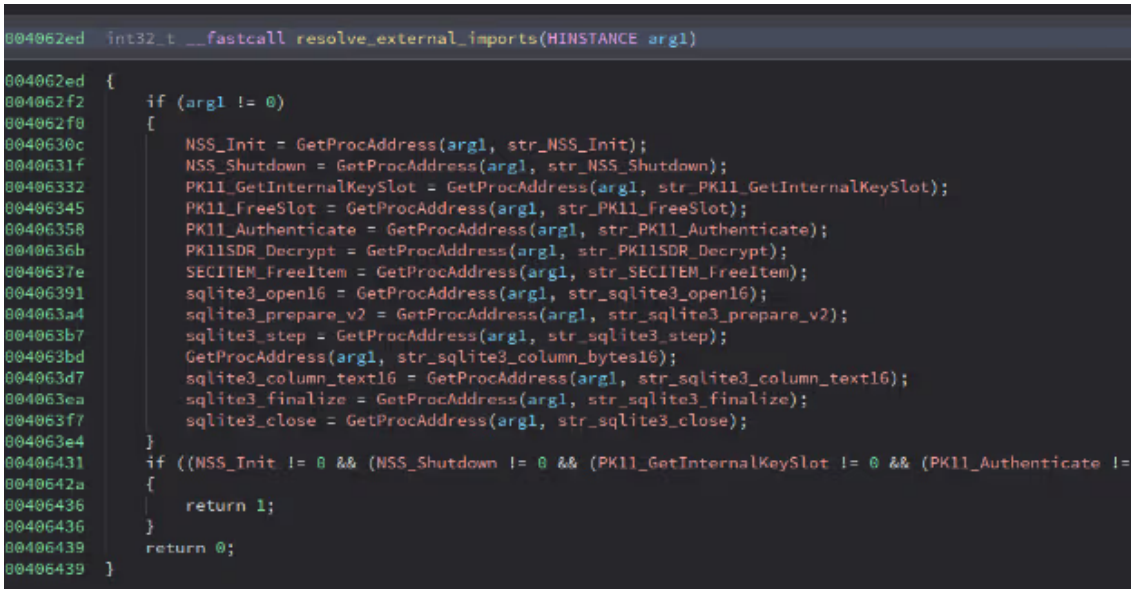


Figure 12. Raccoon Stealer 2.0 targets browsers based on Chromium and Mozilla’s Gecko
Source: ZeroFox Intelligence

In order for Raccoon Stealer 2.0 to be able to read the data threat actors are interested in, it must load the DLLs from the “libs_” options earlier.



```
004062ed  int32_t __fastcall resolve_external_imports(HINSTANCE arg1)

004062ed  {
004062f2      if (arg1 != 0)
004062f8      {
0040630c          NSS_Init = GetProcAddress(arg1, str_NSS_Init);
0040631f          NSS_Shutdown = GetProcAddress(arg1, str_NSS_Shutdown);
00406332          PK11_GetInternalKeySlot = GetProcAddress(arg1, str_PK11_GetInternalKeySlot);
00406345          PK11_FreeSlot = GetProcAddress(arg1, str_PK11_FreeSlot);
00406358          PK11_Authenticate = GetProcAddress(arg1, str_PK11_Authenticate);
0040636b          PK11SDR_Decrypt = GetProcAddress(arg1, str_PK11SDR_Decrypt);
0040637e          SECITEM_FreeItem = GetProcAddress(arg1, str_SECITEM_FreeItem);
00406391          sqlite3_open16 = GetProcAddress(arg1, str_sqlite3_open16);
004063a4          sqlite3_prepare_v2 = GetProcAddress(arg1, str_sqlite3_prepare_v2);
004063b7          sqlite3_step = GetProcAddress(arg1, str_sqlite3_step);
004063bd          GetProcAddress(arg1, str_sqlite3_column_bytes16);
004063d7          sqlite3_column_text16 = GetProcAddress(arg1, str_sqlite3_column_text16);
004063ea          sqlite3_finalize = GetProcAddress(arg1, str_sqlite3_finalize);
004063f7          sqlite3_close = GetProcAddress(arg1, str_sqlite3_close);
004063e4      }
00406431      if ((NSS_Init != 0 && (NSS_Shutdown != 0 && (PK11_GetInternalKeySlot != 0 && (PK11_Authenticate !=
0040642a      {
00406436          return 1;
00406436      }
00406439      return 0;
00406439  }
```

Figure 13. Raccoon stealer resolving external imports to read browser data later
Source: ZeroFox Intelligence

Data targeted from Chromium-based browsers includes:

- Credentials
- Cookies
- Autofill data
- Credit cards
- Extensions listed in the configuration retrieved from the C2

Data targeted from Mozilla/Gecko-based browsers includes:

- Credentials
- Cookies
- Autofill data

Although the “ews_” option is not necessarily limited to only cryptocurrency-related browser extensions, ZeroFox Intelligence has only observed this to be the use case. **Figure 14** below shows two more functions dedicated to stealing cryptocurrency wallets. The first, “wlts_”, exfiltrates files based on the configuration option of the same name. Other cryptocurrency wallets may still be stolen by the next function, which looks for “wallet.dat” files.

Figure 14. The last several functions of Raccoon Stealer version 2.0

Source: ZeroFox Intelligence

The “grbr_” function uses directory paths, file names or patterns, and other options such as file size specified in the configuration to decide which files it should exfiltrate.

The “tlgrm_” function is similar to “grbr_” but has fewer options. It is meant to target Telegram data, though the same functionality could have been achieved with the generic file grabber.

Taking a screenshot is separated into two functions. The first, “scrnsht_” checks to see if the configuration wants it to take one, and the second function actually takes and sends the screenshot.

Finally, the “ldr_” function is capable of allowing Raccoon Stealer version 2.0 to act as a loader for secondary payloads or execute commands. Each “ldr_” option contains multiple parts. It could contain a command to execute

or the URL of a file to download, and if a URL is given a directory is specified to which the file should be downloaded. The last part specifies which action should be taken (e.g., execute a command, run EXE or DLL).

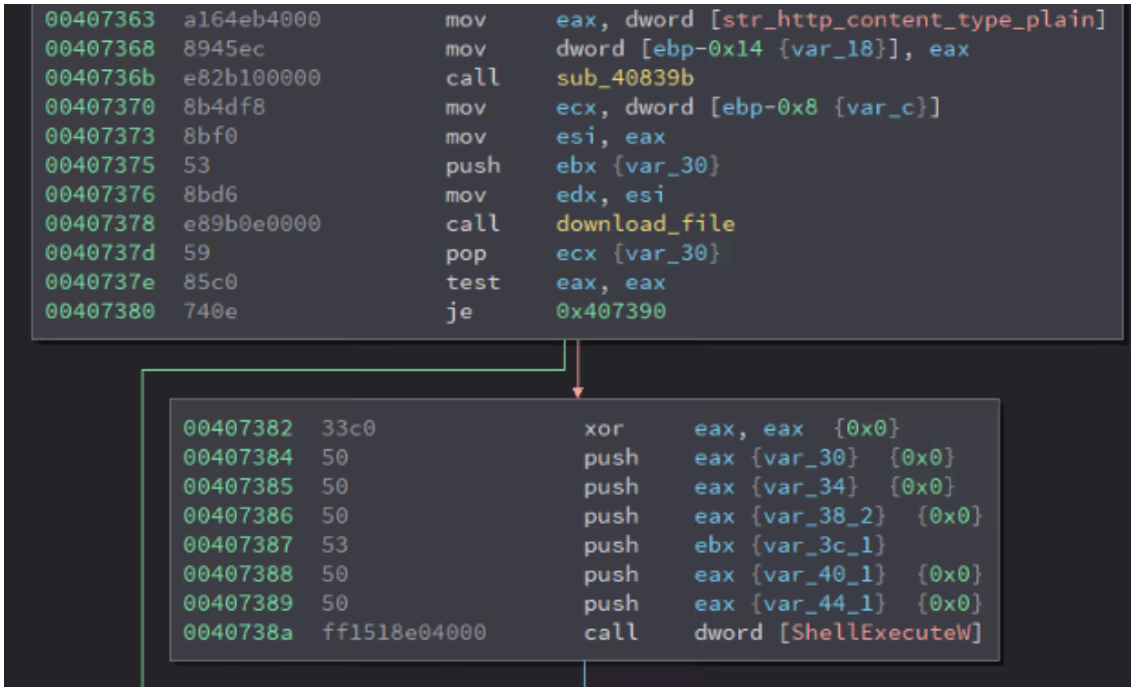


Figure 15. The "ldr_" function can download and execute secondary payloads
Source: ZeroFox Intelligence

Recommendations

ZeroFox Intelligence highly discourages seeking out pirated software of any kind. As in this case, such downloads are often completely fake and will not install the software the victim wanted. In some cases, the download may contain the actual software—as well as a hidden malicious component to infect the victim.

ZeroFox Intelligence also highly recommends that organizations take reports of pirated software on corporate machines seriously. With each download, the risk of infection increases.

YARA

ZeroFox Intelligence has created a public YARA rule that can be found on GitHub.

MITRE ATT&CK

Tactic	Technique	Comments
Reconnaissance	T1592.001Gather Victim Host Information: Hardware	The sstmnfo_ function collects information about the infected system's CPU, installed RAM, and display devices.

Tactic	Technique	Comments
Reconnaissance	T1592.002Gather Victim Host Information: Software	The sstmnfo_ function collects installed applications and their version numbers.
Reconnaissance	T1589.001 Gather Victim Identity Information: Credentials	Raccoon Stealer 2.0 retrieves stored credentials from targeted web browsers.
Execution	T1059 Command and Scripting Interpreter	The ldr_ function can be used to run commands.
Execution	T1559.001Inter-Process Communication: Component Object Model	Raccoon Stealer 2.0 makes use of COM objects in the grbr_ function.
Execution	T1204 User Execution	Samples discovered so far relied on victims seeking out pirated software.
Defense Evasion	T1027.002 Software Packing	Raccoon Stealer 2.0 can be found packed in the wild.
Defense Evasion	T1140 Deobfuscate/Decode Files or Information	Strings and hosts to reach out to are RC4 encrypted and base64 encoded.
Defense Evasion	Path Interception by PATH Environment Variable	Raccoon Stealer 2.0 attempts to add AppData\LocalLow to the PATH variable.
Defense Evasion	T1070.004 Indicator Removal on Host: File Deletion	Several files are copied into the AppData\LocalLow directory and subsequently deleted after use.
Credential Access	T1539Steal Web Session Cookie	Raccoon Stealer 2.0 steals cookies from targeted web browsers.
Discovery	T1057 Process Discovery	If the process is running as SYSTEM, it will enumerate running processes.
Discovery	T1012Query Registry	The registry is used to gather system info, such as the operating system and currently-installed software.
Discovery	T1082 System Information	Raccoon Stealer 2.0 gathers system information, such as the victim operating

Tactic	Technique	Comments
	Discovery	system, system architecture, user locale, installed applications, and more.
Discovery	T1614.001System Location Discovery: System Language Discovery	User locale is checked, but no specific action is taken.
Discovery	T1124System Time Discovery	The victim's time zone is checked and compared to GMT/UTC.
Collection	T1005 Data from Local System	Raccoon Stealer 2.0 offers configurable file-stealing capabilities for actors to choose based on their interests.
Collection	T1113 Screen Capture	Raccoon Stealer 2.0 takes a screenshot near the end of its execution.
Command and Control	T1071.001Application Layer Protocol: Web Protocols	Raccoon Stealer 2.0 uses standard HTTP requests to exfiltrate data and download files.
Command and Control	T1105Ingress Tool Transfer	Raccoon Stealer 2.0 downloads a set of legitimate DLL files to read browser data.
Exfiltration	T1020Automated Exfiltration	Data exfiltration is customizable by the actor through specified directories and file name patterns.
Exfiltration	T1030Data Transfer Size Limits	Actors have the ability to only steal files within a configurable size limit.
Exfiltration	T1041Exfiltration Over C2 Channel	Data is exfiltrated over HTTP and in plain text.

IOCs

Type	IOC
IPv4	5.252.22.62
IPv4	45.142.212.100
IPv4	51.81.143.169
IPv4	51.195.166.171

Type	IOC
IPv4	51.195.166.175
IPv4	51.195.166.176
IPv4	51.195.166.184
IPv4	51.195.166.201
IPv4	62.113.255.110
IPv4	80.92.206.126
IPv4	80.92.206.215
IPv4	85.202.169.112
IPv4	188.215.229.203
IPv4	193.106.191.146
IPv4	194.156.98.151
Domain	wiwirdo.ac[.]ug
URL	hxxp://<c2 address>/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nss3.dll
URL	hxxp://<c2 address>/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/msvcp140.dll
URL	hxxp://<c2 address>/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/vcruntime140.dll
URL	hxxp://<c2 address>/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/mozglue.dll
URL	hxxp://<c2 address>/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/freebl3.dll
URL	hxxp://<c2 address>/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/softokn3.dll
URL	hxxp://<c2 address>/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/sqlite3.dll
URL	hxxp://<c2 address>/aN7jD0qO6kT5bK5bQ4eR8fE1xP7hL2vK/nssdbm3.dll
URL	hxxp://wiwirdo.ac[.]ug/azne.exe
URL	hxxp://wiwirdo.ac[.]ug/pm.exe
URL	hxxp://wiwirdo.ac[.]ug/cc.exe
URL	hxxp://wiwirdo.ac[.]ug/rc.exe
SHA256	048c0113233ddc1250c269c74c9c9b8e9ad3e4dae3533ff0412d02b06bdf4059
SHA256	0661dbb6a0ff7d84c25ae7dd840cefd470279346bd476f1cff5d766f0902a277

Type	IOC
SHA256	0b7d9b786726641c28afda4f641baa4811e0d4c8937748075e61611843e94234
SHA256	0c722728ca1a996bbb83455332fa27018158cef21ad35dc057191a0353960256
SHA256	263c18c86071d085c69f2096460c6b418ae414d3ea92c0c2e75ef7cb47bbe693
SHA256	27e02b973771d43531c97eb5d3fb662f9247e85c4135fe4c030587a8dea72577
SHA256	516c81438ac269de2b632fb1c59f4e36c3d714e0929a969ec971430d2d63ac4e
SHA256	5e614758b6344d6aa9619a75c110b9af4ea2dc1b1103c542e5d10e8d5fc2d66a
SHA256	7299026b22e61b0f9765eb63e42253f7e5d6ec4657008ea60aad220bbc7e2269
SHA256	79103532395036c14b755d90f9cacfdec6b588f1b031a7cba936c1b9d2ef3b51
SHA256	8655a544a26bade11fbda736c2af2a475ed12f2150efce7f0500b6fc6d317cb8
SHA256	89a718dacc3cfe4f804328cbd588006a65f4dbf877bfd22a96859bf339c6f8bc
SHA256	960ce3cc26c8313b0fe41197e2aff5533f5f3efb1ba2970190779bc9a07bea63
SHA256	99f510990f240215e24ef4dd1d22d485bf8c79f8ef3e963c4787a8eb6bf0b9ac
SHA256	9e239b12c8cc7f5f7fc0a46121aa5dbfd82306f08c4b04a6ac9f61495ecd410b
SHA256	9ee50e94a731872a74f47780317850ae2b9fae9d6c53a957ed7187173feb4f42
SHA256	bc15f011574289e46eaa432f676e59c50a9c9c42ce21332095a1bd68de5f30e5
SHA256	e514d7ee18dbe841e411f03dd6e0f498b509599d81d15c0945325070cdc8c687
SHA256	f20dcb9477e356e91e1b00abc351f749739f98ca395840ae3062d6cebc72f85b
SHA256	f9c4d451f8c9d4e546e67348c4cc2e8810aa5c39d4fabe1ee891408a0bc53043
SHA256	fb26544d45a1166e15e37853786f0b98ff876d1ce94c240a0f3bc2f9a8fb258f
SHA256	fba1005e8c248ec460e6c13cb38759bd70d9db4882f88f651b194ab1800e656c

SCOPE NOTE

ZeroFox Intelligence is derived from a variety of sources, including—but not limited to—curated open-source accesses, vetted social media, proprietary data sources, and direct access to threat actors and groups through covert communication channels. Information relied upon to complete any report cannot always be independently verified. As such, ZeroFox applies rigorous analytic standards and tradecraft in accordance with best practices and includes caveat language and source citations to clearly identify the veracity of our Intelligence reporting and substantiate our assessments and recommendations. All sources used in this particular Intelligence product were

identified prior to 12:00 PM (EDT) on June 28, 2022; per cyber hygiene best practices, caution is advised when clicking on any third-party links.
