SystemBC: RIG & Fallout Exploit Kits Campaign Analysis | Proofpoint US

By Kade Harmon | Kafeine | Dennis Schwarz | The Proofpoint Threat Insight Team

Published: 2019-08-01 · Archived: 2026-04-05 14:38:25 UTC

Overview

SystemBC is a previously undocumented malware that we have recently observed as a payload in both RIG and Fallout exploit kit (EK) campaigns. While EK activity has remained quite low relative to its peak in early 2016, exploit kits remain important vectors for malware distribution, particularly in regions where Windows piracy is common. The new malware utilizes SOCKS5 proxies to mask network traffic to and from Command and Control (C&C) infrastructure using secure HTTP connections for well-known banking Trojans such as Danabot, which we have also observed distributed in the same EK campaigns.

A related sample of this malware may have been identified by other Infosec researchers on Twitter [2] in mid-October of 2018 distributing AZORult instead of Danabot. SystemBC may also have connections to Brushaloader and related malware.

Campaign Analysis

While analyzing a Fallout EK campaign on June 4, 2019, Proofpoint researchers observed the distribution of a previously unseen proxy malware. Most recently, the malvertising-based Fallout exploit kit chain has been used to deliver instances of Maze ransomware (Figure 1).

Clusters	Tags	Date
Fallout	exploit-kit Fallout [redacted] Keitaro Malvertising PopCash DEU Ransomware Maze	2019-05-26
Fallout	exploit-kit Fallout [redacted] Keitaro Malvertising PopCash CHE SystemBC	2019-06-04
Fallout	exploit-kit Fallout [redacted] Keitaro Malvertising PopCash AUS Maze Ransomware	2019-06-12
Fallout	exploit-kit Fallout [redacted] Keitaro Malvertising PopCash TWN Maze Ransomware	2019-07-08

Figure 1: Malvertising-based Fallout EK chain that previously delivered Maze ransomware

On June 6, 2019, Proofpoint researchers observed the new proxy malware in the wild again [3]. This time it was being delivered via a Fallout EK and PowerEnum campaign (Figure 2) alongside an instance of the Danabot banking Trojan (affiliate ID 4).

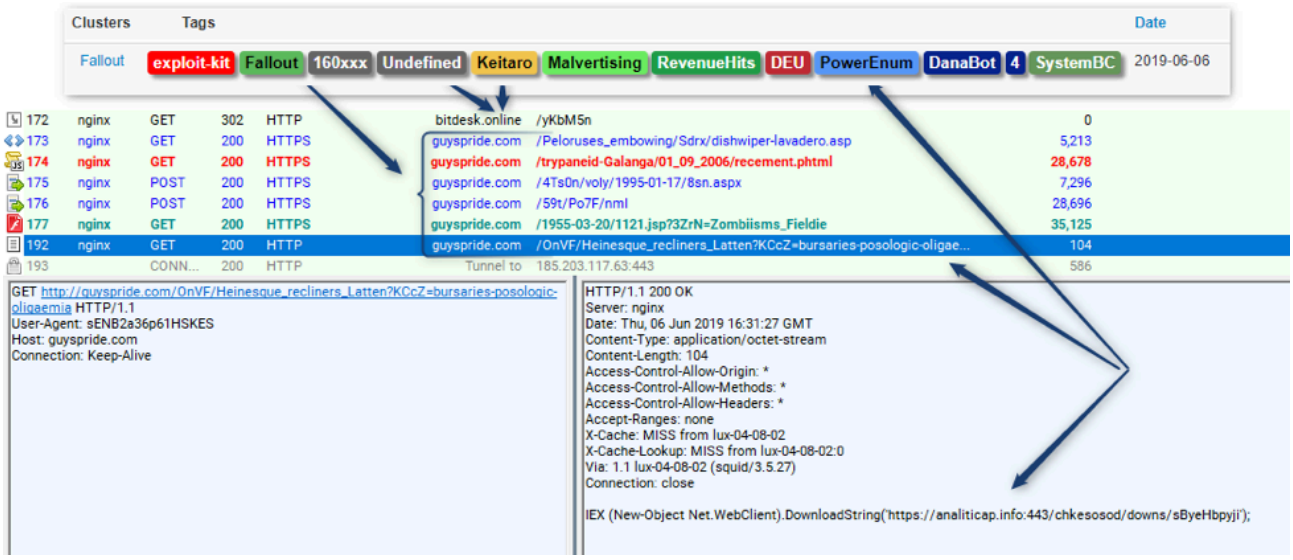

Figure 2: Fallout EK dropping PowerEnum, which has been observed instructing the download of Danabot Affid 4 and a proxy malware DLL

Between July 18 and 22, 2019, Proofpoint researchers observed the proxy malware a third time. This time it was being distributed by the Amadey Loader, which itself was being distributed in a RIG EK campaign.

Other security researchers have also observed the malware being used in the wild. Notably, Vitali Kremez saw a sample of the malware on May 2, 2019 [3], and @nao_sec observed it in connection with in a third Fallout EK campaign on July 13, 2019 [4].

Marketplace Analysis

Since this proxy malware was being used in multiple separate campaigns, Proofpoint researchers believe it was very likely that it was being sold in an underground marketplace. Moreover, we found an advertisement from April 2, 2019, on an underground forum that described a malware named “socks5 backconnect system” (Figure 3) that matched the functionality of the malware seen in the above campaigns. To differentiate from other malware leveraging SOCKS5, we dubbed the new malware “SystemBC” based on the URI path shown in the advertisement’s panel screenshots (Figures 4 and 5).

```
Topic updated 04/02/2019

I sell socks5 backconnect system

consists of:

client part

- socks.exe - does not hide from the dispatcher. minimum load on av detekty. XP support and above
- socks.dll - separate assembly as dll

dll is a bit better embedded in your bot and uses all its capabilities (hiding from the controller, bypasses the firewalls)

there is autorun. after rebooting the pc, the socks are returned.

Otstuk about 70% after the standards crypt.

the system works in multi-threaded mode, which gives a high increase in the speed of socks

server part

supports installation both on win servers and on Linux (server requirements 400mb free RAM for 1 000 socks)

- server.exe to run on win servers. supports up to 40,000 incoming connections
- server.out to run on Linux
- php admin

For software, a dedicated (non-shared) 1 gbit channel is recommended.
if they just hang and are not used the internet is not consumed. each sock consumes ~ 3 mbit when used

features

- loader with update function every N hours (for long survivability it is necessary to update the crypts)
- firewall (access to socks only from trusted ip)
- authorization on socks by login and password
- GeoIP

The bot also works at integrity level low. only in autorun in such cases will not be added

GeoIP can be configured via maxmind online service (weekly database updates. latest data)
just insert id and key from maxmind

The system is developed in assembler. high speed minimum size

file weight

socks.exe 12kb
socks.dll 10kb
server.exe 14kb
server.out 10kb (for Linux)

supports regular domains and ip + .bit domains (via your dns or public)
After the purchase I give a link to the builder (10 attempts)

screen builder hxxp://i66.tinypic[.com/5wcuax.jpg
admin screen
hxxp://i63.tinypic[.com/j7w4zd.jpg
hxxp://i68.tinypic[.com/szv9za.jpg

set cost $ 250 in bitcoin
```

Figure 3: Original forum advertisement for SystemBC (translated from Russian)

The advertisement also contains screenshots of the C&C panel (Figure 4-6). The simple C&C panel boasts a list of victim computers, automated updating, and built-in authentication. The builder allows users to create a set number of samples with custom configurations.

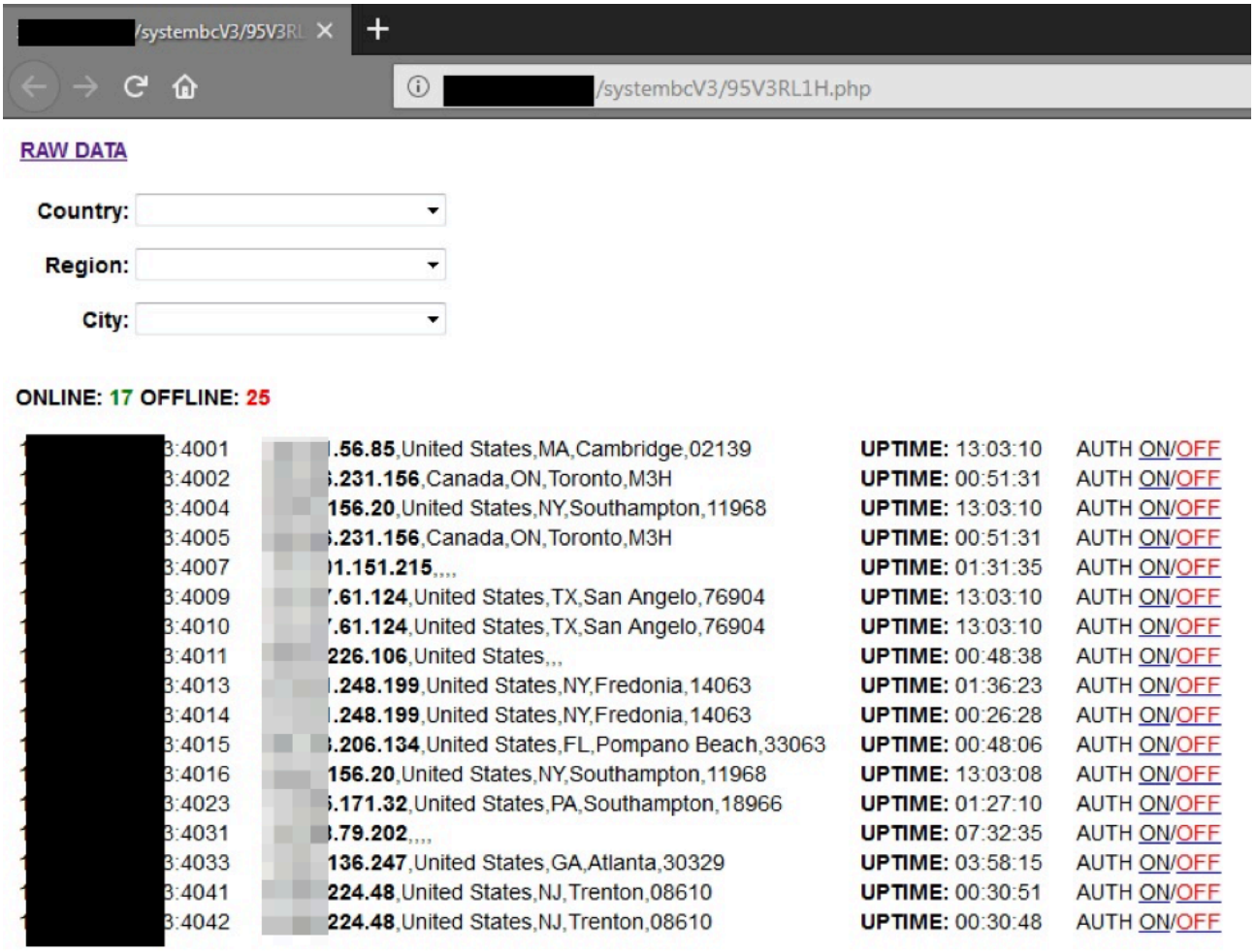

Figure 4: SystemBC Administrator Panel (as observed in an underground advertisement)

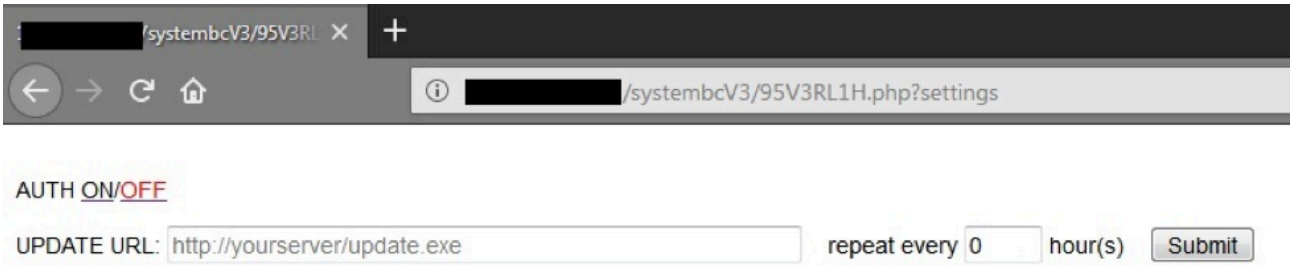

Figure 5: Another section of the SystemBC Administrator Panel (as observed in an underground advertisement)

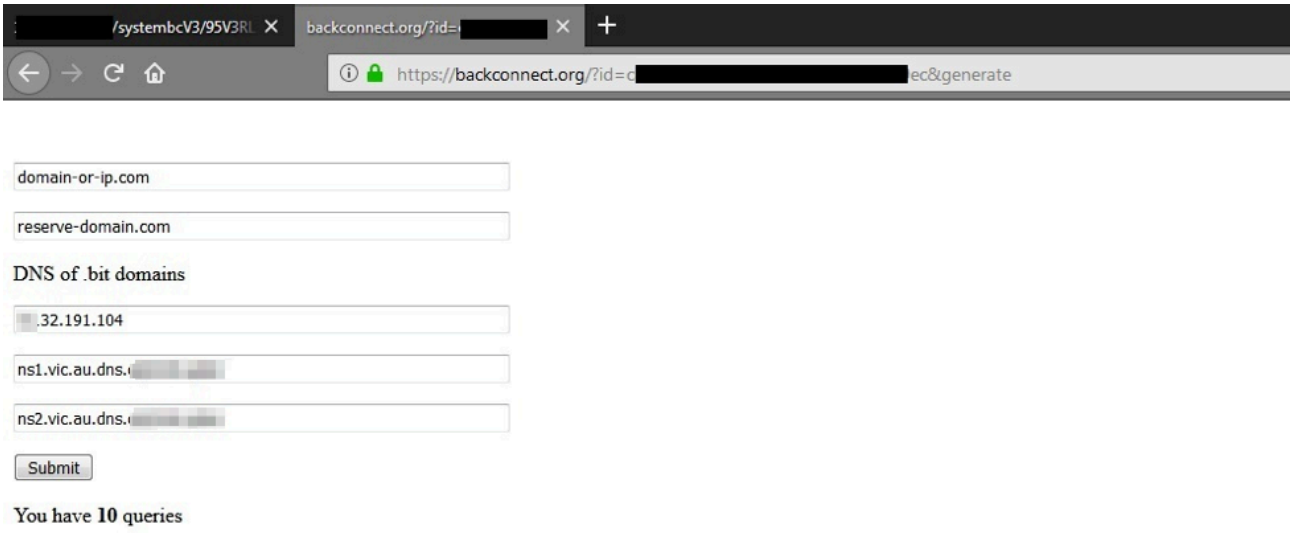

Figure 6: SystemBC Builder (as observed in the advertisement noted above)

Malware Analysis

SystemBC is written in C++ and primarily sets up SOCKS5 proxies on victim computers that can then be used by threat actors to tunnel/hide the malicious traffic associated with other malware.

Configuration

Important strings such as the C&C servers, DNS servers, and port number are encrypted with a 40-byte XOR key that is stored in memory. Reference [5] is a GitHub-hosted Ghidra Python script that can be used to decrypt the configuration from the analyzed sample (Figure 7):

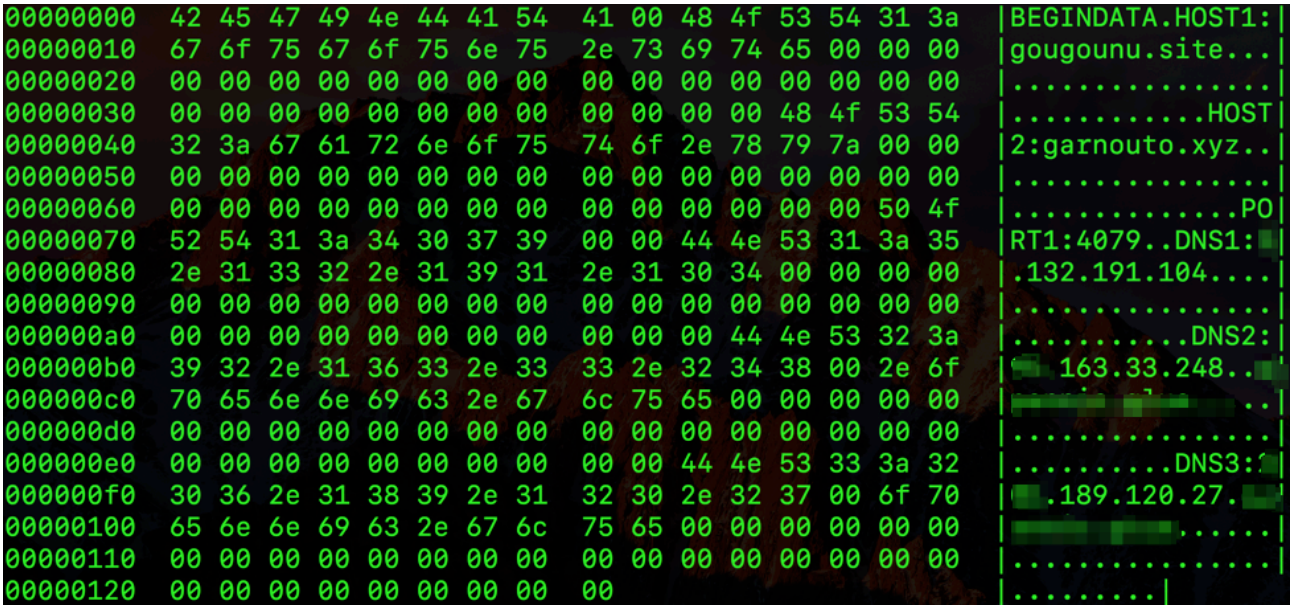

Figure 7: Decrypted malware configuration

The DNS servers are used to resolve ".bit" domains. The malware calls a function to check whether the server name ends in ".bit." If it does, a DNS query will be generated by iterating through the list of DNS hostnames until

the malware finds a valid server (Figures 8 and 9). It is worth mentioning that although both the screenshots and samples reference OpenNIC, they are no longer resolving ".bit" domains [6].

```
decrypt_str(uParm1,pbParm2,(byte *)serverName,(char *)0xffffffff,(byte *)0x0);
pcVar2 = null_terminate_string(serverName);
cp = serverName;
if (pcVar2 == (char *)0x0) {
  in = (in_addr).bitCheck(serverName,2);
  cp = inet_ntoa(in);
}
```

Figure 8: SystemBC performing the initial ".bit" check

```
if (((int)(pcVar1 + -4) < 0) || (*(int *)(pcVar1 + -4 + (int)param_1) != 0x7469622e)) {
```

Figure 9: The malware checks to see if the last four characters of the server name are ".bit"

Command and Control

All packets are encrypted using standard RC4, but the S-Box is initialized in a novel way (Figure 10):

```
iVar3 = -0x10204;
i = 64;
do {
  S[i] = iVar3;
  iVar3 = iVar3 + -0x4040404;
  i = i - 1;
} while (i != 0);
```

```
for i from 0 to 255
    S[i] := i
endfor
```

Figure 10: Side-by-side comparison of the SystemBC RC4 S-Box initialization (left) and the more common implementation (right)

An example of the hex-encoded (for visibility) C&C communications are available in Figure 11:

```
c9b36aec884ac19ccefa68453cb14ea8758668c6c96aeca6385bacb06891eb3144561ecda0df696fdad20
0000000000000003083ea94fd5f506bb0905729ff801c32dca3502bed85a09e27ea148f712dc21c2fe55b
cedd1d955642e4b99f38e761e86cf3
819ff095849fea96be7d1e3c9ed50068c99c4c1ef1986f1a91a795bd6c18
8094ea849eea94ad1c7d5ff1a0
819cfc95849fea96a26b0a28dfc71d68dbc64212b09c6c12be7f
8394ea849eea94ad1c7d5ff1a0
809fef97979deb97ad1d7d5e0da36df8fd9b93d6201524901b8ad89773bf24900b2d52ea4b7dd7a36d85e
f30188c99bfb423f88034321f894911ec6d0e1fdb540a79a4a203714fa9a3b074954629473a8fa4206e3a
0c73ecd9df8a2d76cb8276813512c7c3e90f2e1fa786a4aa7f55cf6fdcfcbb17bc6cf44a3e46309c2d6d8
91b86a3042f0d26a67878864bedd052d7ba1a52d0216bb616e04f152736e6f50e0c78c3fe22c877e0b27c
34178573389843e56626ac17cc3a2782ae9764cc5e6c1eb9d1b652444087ed7d1ede11aea52c386f06c54
fd88f87c8fddfb01a1fb866196e6a24a85571a2640a8e62b52dbdda5976a5bdc9a455256ba31142792d7c
376781c27b98989d31dcb9d33f88657cb98e82163fbce531811731f4b6c5fd1c75a44d0bfcb8267223dac
9af6a55a050efafba23be2ac4e6b9854fdd9be3fd856be779b7056912e29a828313ab706f2eb4db03579c
159976aedd7ff94c69c940ac8970affb1dc2cb1a77689da69c3d277e95964aa5960b1b9c8fd02554b6255
40516f6c34cca0697b85679c4a9072a7d28232420797259f234eb3594d3c37259daa232a411490acc3991
eaeb10d9f23905d88c9e7f6f58701a0c2ef784ac72d1821a22733cd2fca914db96511c1dae4d4540594b2
5f423d5b2afb78d4a2b4fb9e5d6fdc13c59b81e09d0bb27beff8a44fc24adf731816bce739c4e1ed40cdb
c14adc47735adb7f19042d
80b2e1979de995d71e7d5f87a36d8b81ab9a941b1dc3fa9c78a943511894770773b162dbf0a3cb7339d1a
67582ecbfb423f88034321f894911ec6d0e1fdb540a79a4a203714fa9a3b074954629473a9c874a04070d
53eeeedc5706968606a9c8d2c99d0587dfae40beef0c4d87023ec981b1688131e437d4154d26e52d46911
993a7103f6f45c4161aeb3c959faeb879b5cb9f530d47d120330e2ed2c498f431fe14ca6771c38d2009a7
```

Figure 11: Example C&C communications (hex-encoded for visibility)

The client begins the communication by sending a 100-byte packet to the C&C address.

The packet contains four elements:

Bytes 0-49	Plaintext RC4 key
Bytes 50-51	Windows build ID
Bytes 52-53	Boolean determining if the client is running on an x64 processor
Bytes 53-99	Client machine’s account name, with trailing zeroes

It was derived from the following decompiled code (Figure 12):

```
                /* Create Packet */
GetUserNameExA(2,&packet->username,packet);
copyArg1toArg2(&key,(undefined *)packet,50);
uVar3 = rtlGetVers();
*(undefined4 *)&packet->rtlCheck = uVar3;
lpParameter = isWow64Proc();
packet->isWow64 = (char)lpParameter;
                /* RC4 last 50 bytes of packet with key */
RC4_implementation(&key,50,&packet->rtlCheck,50);
                /* Send Packet */
sendingData(the_socket,(char *)packet,100,(HANDLE)0x0);
```

Figure 12: Logic of initial packet creation

This is verified by decrypting the last 50 bytes of the packet using the first 50 bytes as the RC4 key. The result is plaintext data (Figure 13).

```
>>> key
'\xc9\xb3j\xec\x88J\xc1\x9c\xce\xfahE<\xb1N\xa8u\x86h\xc6\xc9j\xec\xa68[\xac\xb0h\x91\xeb1DV\x
1e\xcd\xa0\xdfio\xda\xd2\x00\x00\x00\x00\x00\x00\x00\x00'
>>> rc4 = ARC4(key)
>>> rc4.decrypt(data)
'\xb1\x1d\x00\x01                                    \x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00'
```

Figure 13: Decryption of the initial packet data

The return packet from the C&C server contains two main segments, a header and data, which are decrypted separately using the RC4 key from the first packet. The header, which makes up the initial 4 bytes of the packet, has a type, index, and length field. The data segment takes up the remaining bytes in the packet and contains details for the creation of a SOCKS5 proxy connection. A breakdown of the packet is as follows:

Byte 0	Type 1: Indicates the following “data” packet is SOCKS5 proxy traffic and is associated with the identified index number. 0: Create a new proxy and will assign it the given index number. The index number is assigned by the C&C server and is used to associate traffic to a particular proxy.

	-1: Update malware by downloading an executable, save it with a randomly generated name in the TEMP directory, then run the file
Byte 1	Index: Tells the infected machine which proxy to use
Bytes 2-3	Length: records the number of bytes in the following data chunk
Bytes 3-	SOCKS5 packet information

```
>>> rc4.decrypt(header)
'\x00\x01\x1a\x00'
>>> rc4 = ARC4(key)
>>> rc4.decrypt(data)
'\x05\x01\x00\x03\x13accounts.google.com\x01\xbb'
```

Figure 14: Decryption of the first response packet

Referencing source [7], we can map these values to a structured SOCKS5 client connection request packet to yield (for example):

Version: 5

Command code: 1

Reserved: 0

Address type: 3

Length of domain: 19

Domain name: accounts.google.com

Port number: 443

The second packet sent from the client to the server contains a 3-byte RC4-encrypted header consisting of an index number and data length. Repeating the above steps, but instead assuming a 3-byte header, we can decrypt a typical SOCKS5 server acknowledgment:

Version: 5

Status: 0

Reserved: 0

Address type: 1

IPv4 Address: 00 00 00 00

Port number: 00 00

This sample goes on to initialize another proxy with a different domain name and an incrementing index value.

With the proxies initialized, the client now begins to retrieve data requested from the C&C via HTTPS. We can discern that data with a 3-byte header contains response data sent from the proxy while data sent with a 4-byte header are commands from the C&C server.

Conclusion

Proofpoint researchers have identified a previously undocumented proxy malware, dubbed "SystemBC", being distributed by the Fallout and RIG exploit kits.

In the most recently tracked example, the Fallout exploit is used to download the Danabot banking Trojan and a SOCKS5 proxy which is used on the victim's Windows system to evade detection of command and control (C&C) traffic. The synergy between SystemBC as a malicious proxy and mainstream malware creates new challenges for defenders relying on network edge detections to intercept and mitigate threats like banking Trojans.

Proofpoint recommends that organizations continue to remain vigilant in keeping their Windows client and server operating systems as well as infrastructure devices patched with vendor-recommended updates and patches, to retire the use of legacy systems which use susceptible browser plugins such as Adobe Flash Player, and to retire legacy Windows systems that may be susceptible to exploit kits such as Fallout.

References

[1] https://www.proofpoint.com/us/threat-insight/post/brushaloader-still-sweeping-victims-one-year-later

[2] https://twitter.com/James_inthe_box/status/1150397404916543488

[3] https://twitter.com/VK_Intel/status/1123867031709863937

[4] https://twitter.com/nao_sec/status/1150038665013235717

[5] https://github.com/EmergingThreats/threatresearch/blob/master/SystemBC/XORscript.py

[6] https://wiki.opennic.org/votings/drop_namecoin

[7] https://samsclass.info/122/proj/how-socks5-works.html

Indicators of Compromise (IOCs)

IOC	IOC Type	Description
e8627abf6b2e9ccebbc544d485b4e2bccd22580b4dc7ba8510d4e4e8bba63fc9	SHA256	June 4, 2019 SystemBC Malware
mie[.crypto-crypto[.site	Hostname	June 4, 2019 SystemBC C&C
893305fd80eb324b262406c60496163ed4ff73dad679f1bd543ff703de457f91	SHA256	June 6, 2019 SystemBC Malware
gougounu[.site	Domain	June 6, 2019 SystemBC C&C
3261f0e45d867236d4794b2a3dce38663bb319a6fabec7ae07fac3237e474689	SHA256	July 18, 2019 Amadey
dsntu[.top elienne[.net amnsns[.com	Domains	July 18, 2019 Amadey C2 hosted behind Sandiflux
hxxp://mmasl[.com/s1.exe hxxp://calacs-laurentides[.com/s1.exe	URLs	July 18-22, 2019 Amadey Tasks (SystemBC)
9024a3ec7df6ef51f69c2e452da26d3a45743fd1c49b2d59beeb83be0949fe06	SHA256	July 18, 2019 SystemBC

		Malware
20a7cfcaf76890ad5e959e5662f421f41126d3ee1edace8f5531f8effecb6051	SHA256	July 22, 2019 SystemBC Malware
146.0.75[.34	IP	July 18-22, 2019 SystemBC C&C
6269d9ce2adb19a46bffefe50c9b3e00974c4dc8f4c2dc0156545707efb4f453	SHA256	July 24, 2019 SystemBC Malware

---

Source: https://www.proofpoint.com/us/threat-insight/post/systembc-christmas-july-socks5-malware-and-exploit-kits