SOPHOS
Cybersecurity made simple.

“VPNFilter” botnet: a SophosLabs analysis

By **Sergei Shevchenko**, Threat Research Manager, SophosLabs

A SophosLabs technical paper May 2018

Contents

A technical investigation of the malicious components involved in the attack that infected over 500,000 routers and network storage devices.

By Sergei Shevchenko, Threat Research Manager, SophosLabs

Date: May 23rd, 2018

Thanks to the Cyber Threat Alliance, the SophosLabs researchers were provided early access to malware samples collected by Cisco TALOS team in their research of the VPNFilter botnet activity. Besides updating our protection data, we also had a chance to take a closer look at the attack components and the 3 stages of the attack. Here is our findings.

First stage implant

The First stage implant (0e0094d9bd396a6594da8e21911a3982cd737b445f591581560d766755097d92) is compiled as a x86 ELF executable.

This executable was first submitted to VirusTotal on June 12 , 2017 from a user in Taiwan. According to VirusTotal, the submitted file has a filename:

C:\Users\chli\Documents\qsync.php

Possibly, the file was fetched from a remotely hosted script called qsync.php, using a Windows system. However, it's not clear how this sample was used to compromise the devices.

When run, the implant schedules itself to be executed periodically, by modifying crontab (cron table) file.

The cron format[1] has five time and date fields: minute, hour, day of month, month, and day of week. If a value is specified as ***/step**, execution takes place at every interval or step through the unrestricted range.

By appending the schedule argument "*/5 * * * *" to the crontab, the implant is scheduled to be activated every five minutes:

```
fd = open_file("/etc/config/crontab", "a");
_fd = fd;
if (fd)
{
   format_sys_write(fd, "*/5 * * * * %s\n", (int)&fname);
   fd = close(_fd);
}
```

1 https://www.ibm.com/support/knowledgecenter/SSEPGG_9.5.0/com.ibm.db2.luw.sql.rtn.doc/doc/c0054381.html

The implant keeps its critical strings encrypted. For that, it relies on a modified RC4 algorithm. In a normal RC4 implementation, the RC4 initialization routine calculates an index into the state table, using the key. Next, two bytes in the state table are swapped in place, where the first byte is pointed by the incremented index i, and the second one - by the newly calculated index index2, as shown below:

```
#define swap_byte(a, b) {swapByte = a; a = b; b = swapByte;}
for (i = 0; i < 256; i++)state[i] = i;
key_index = 0;
index2 = 0;
for (i = 0; i < 256; i++)
{
  index2 = (key[key_index] + state[i] + index2) & 0xFF;
  swap_byte(state[i], state[index2]);
  if (++key_index == key_size)key_index = 0;
}
```

The implant however, initializes the state table differently. Instead of permuting the state table by swapping the bytes in it, it simply applies XOR to the state table, using the same RC4 key.

Apparently, this flavor of RC4 initialization is known to be used by BlackEnergy[2].

```
for (i = 0; i < 256; i++)state[i] = i;
key_index = 0;
for (i = 0; i < 256; i++)
{
  state[i] ^= key[key_index];
  if(++key_index == key_size)key_index = 0;
}
```

The RC4 key is a four character string hard-coded as **"%^:d"**. The rest of the RC4 implementation is identical to the standard algorithm.

In total, the body of the implant contains 12 encrypted strings. Each encrypted string is stored as a string length that takes one byte, followed with the encrypted string itself.

Once decrypted these strings become:

- **/var/run/client.crt**
- **/var/run/client.key**
- **/var/run/client_ca.crt**
- **0.3.9qa**
- **/var/run/msvf.pid**
- **http[://]toknowall.com/manage/content/update.php**
- **/var/vpnfilter**
- **/update/test**
- **http[://]photobucket.com/user/nikkireed11/library**
- **http[://]photobucket.com/user/kmila302/library**
- **http[://]photobucket.com/user/lisabraun87/library**
- **http[://]photobucket.com/user/katyperry45/library**

The first three strings are the filenames where the implant saves three client certificates, hard-coded within its own body. These client-side SSL certificates are used for authenticated requests to the C2 server, over HTTPS (port 443).

The version number "**0.3.9qa**" is saved into the file **/var/run/msvf.pid**.

The **/var/vpnfilter** is used as a temporary filename for the downloaded files.

The implant relies either on hard-coded Photobucket URLs or Toknowall C2 website to fetch the images. The images are used to extract a second stage server IP from the images' EXIF metadata.

Next, a payload module is fetched from the second stage server, using a URL path **/update/test**. The downloaded module is saved as **/var/vpnfilter**, assigned execution permission with the **chmod(511)** command, then executed with the **sys_execve()** system call.

2 https://www.secureworks.com/research/blackenergy[2]

Second stage payload: a backdoor trojan

The second stage payload fetched by the implant **(8a20dc9538d639623878a3d3d18d88da8b635ea52e5e2d0c2cce4a8c5a703db1)** is a backdoor trojan compiled as x86 ELF executable.

Just like the first stage implant, its critical strings are encrypted using the same method. The RC4 key is different this time: "**g&*kdj$dg0_@@7'x**".

The decrypted strings expose backdoor commands, IP addresses of C2, and some other configuration parameters.

For example, the backdoor is able to accept and execute the following remote commands:

- **download** - download remote file, save it as **/var/tmp/vpn.tmp**
- **reboot** - terminate current process with **sys_exit()** system call
- **restart** - reboot the device with **sys_reboot()** system call;
- **delay** - appears to invoke delayed reboot
- **copy** - read local file contents
- **exec** - execute command or another plugin, using sys_ execve() system call with the following shells:

  /bin/sh

  /bin/ash

  /bin/bash

  /bin/shell

- **kill** - terminate process(es) with the sys_kill() system call, delete own files and directories, such as:

  /var/run/tord

  /var/run/

  /var/run/vpn.pid

  /var/tmp/vpn.tmp

  etc.

- **pxs** - set C2 proxy, i.e. the module contains 2 hard-coded proxies in it:

  217.12.202.40

  91.121.109.209

- **port** - set proxy port
- **tr, mds, tor, me** - set other configuration parameters

In its communications, the backdoor relies on a user agent string randomly selected from a list of nine strings:

user_agent = user_agents[PRNG() % 9];

where user_agents table consists of:

- **Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:52.0) Gecko/20100101 Firefox/52.0**
- **Mozilla/5.0 (Windows NT 6.1; rv:52.0) Gecko/20100101 Firefox/52.0**
- **curl/7.47.0**
- **Wget/1.17.1 (linux-gnu)**
- **git/2.7.4**
- **Google Chrome/64.0.3282.140 Windows**
- **Google Chrome/64.0.3282.140 Linux**
- **Lynx/2.8.8pre.4 libwww-FM/2.14**
- **python-requests/2.18.4**

Just like the implant, the backdoor communicates with its proxies via a SSL connection, relying on client-side SSL certificates.

The module tries to determine the presence of TOR by parsing socket info from **/proc/net/tcp**. For each enumerated socket descriptor, it then tries to find a file descriptor (by using a method described here[3]) to a socket that has open connection on port **9050**, that is used by TOR.

With the TOR module installed as a third stage plugin, the communication will take place via the following .onion domains:

- **6b57dcnonk2edf5a.onion/bin32/update.php**
- **tljmmy4vmkqbdof4.onion/bin32/update.php**

Backdoor modules built for different platforms are almost identical in their functionality. The strings are encrypted using the same key.

A subtle difference exists in the internal platform ID parameters. For example, x86 module uses IDs:

- pDJOSERi686QNAPX86 or pPRXi686QNAPX86
- i686

Backdoor module built for ARM CPU may have these parameters set to:

- pDJOSERarmv5lQNAP_ARM
- armv5l

Backdoor compiled for MIPS:

- pDJOSERmipsDGN2200V4
- mips

Third stage plugin

A Third stage plugin **(afd281639e26a717aead65b1886f98d6d6c258736016023b4e59de30b7348719)** is a TOR client. Compiled as x86 ELF executable, it shares the same code as known open-source TOR client implementations[5].

Another Third stage plugin is built for MIPS architecture **(f8286e29faa67ec765ae0244862f6b7914fcdde10423f96595cb84ad5cc6b344)**. The plugin represents itself a sniffer that looks for several interesting traffic patterns, such as:

- **"/tmUnblock.cgi" -** a vulnerable CGI script in some Cisco/Linksys router firmware; this executable is linked to several exploits and malicious executables, such as "Moon Worm", a malicious Bitcoin miner that has infected Linksys routers in the past[6]
- **"*modbus*\n%s:%uh->%s:%hu"** - a packet used in Modbus, a standard communication protocol, commonly used for connecting industrial electronic devices, such as PLC
- **"Basic Og=="** - part of HTTP authentication packet, that means[7] "Empty username and empty password"

Other patterns related to HTTP authentication packets:

- **"Password required"**
- **"Authorization: Basic"**
- **"User="**
- **"user="**
- **"Name="**
- **"name="**
- **"Usr="**
- **"usr="**
- **"Login="**
- **"login="**
- **"Pass="**
- **"pass="**
- **"Password="**
- **"password="**
- **"Passwd="**
- **"passwd="**

3 https://stackoverflow.com/questions/3319521/how-can-i-match-each-proc-net-tcp-entry-to-each-opened-socket

4 https://people.torproject.org/~nickm/tor-auto/doxygen/microdesc_8c_source.html

5 https://github.com/kaist-ina/SGX-Tor/

The intercepted data is stacked into the files, formatted as:

%DIR%/rep_%NUMBER%.bin

where **%DIR%** is a working directory, such as **/var/run/vpnfilterw**.

Conclusion

VPNFilter malware is another clear demonstration of a rather philosophical paradigm: the more IoT devices we have helping us out in our daily lives, the more advanced the CPUs become, driving our routers, cars, refrigerators - you name it - the bigger an attack surface becomes.

The type of CPU during the grunt work within all those devices, whether it's ARM or MIPS or Intel x86, doesn't matter much, as long as they are powerful enough, and they are becoming more powerful each day. That's the whole gist of the evolution, and this process won't stop.

VPNFilter also demonstrates how the cybercriminals achieve a high degree of portability by building their code so that it targets different architectures.

What's still interesting in this case however, is the very possibility for an organization or a home to become compromised by allowing a backdoor access via one of the least suspicious devices in its possession: a little black box quietly sitting on a shelf, blinking with its friendly green eyes.

VPNFilter isn''t the first zombie malware to target everyday devices on everyone''s network, and it won''t be the last.

6 https://isc.sans.edu/forums/diary/Whatever+Happened+to+tmUnblockcgi+Moon+Worm/19999/

7 https://chromium.googlesource.com/chromium/src/net/+/master/http/http_auth_handler_basic_unittest.cc

United Kingdom and Worldwide Sales
Tel: +44 (0)8447 671131
Email: sales@sophos.com

North American Sales
Toll Free: 1-866-866-2802
Email: nasales@sophos.com

Australia and New Zealand Sales
Tel: +61 2 9409 9100
Email: sales@sophos.com.au

Asia Sales
Tel: +65 62244168
Email: salesasia@sophos.com

18-05-24 TP-NA (PS)

SOPHOS