

Cyclops Blink

Malware Analysis Report

Version 1.0



Cyclops Blink

Modular malware framework targeting SOHO network devices

Executive summary

- Cyclops Blink is a malicious Linux ELF executable, compiled for the 32-bit PowerPC (bigendian) architecture.
- Persistence is maintained throughout the legitimate device firmware update process.
- Implements a modular framework consisting of a core component and additional modules that are executed as child processes.
- Modules to download/upload files, extract device information, and update the malware have been built-in and are executed at startup.
- Command and control (C2) communication uses a custom binary protocol underneath TLS, and messages are individually encrypted.

Introduction

Cyclops Blink is a malicious Linux ELF executable, compiled for the 32-bit PowerPC (big-endian) architecture. NCSC, FBI, CISA, NSA and industry analysis has associated it with a large-scale botnet targeting Small Office/Home Office (SOHO) network devices. This botnet has been active since at least June 2019, affecting WatchGuard Firebox and possibly other SOHO network devices.

This report covers the analysis of two samples recently acquired by the FBI from WatchGuard Firebox devices known to have been incorporated into the botnet.



Malware details

Metadata

Filename	cpd	
Description	Cyclops Blink - Linux ELF PowerPC big-endian. The size corresponds to the complete file, but the hash values correspond to the executable code segment only.	
Size	2494940 bytes	
MD5	d01e2c2e8df92edeb8298c55211bc4b6	
SHA-1	3adf9a59743bc5d8399f67cab5eb2daf28b9b863	
SHA-256	50df5734dd0c6c5983c21278f119527f9fdf6ef1d7e808a29754ebc5253e9a86	

Filename	cpd	
Description	Cyclops Blink - Linux ELF PowerPC big-endian. The size corresponds to the complete file, but the hash values correspond to the executable code segment only.	
Size	2494940 bytes	
MD5	bbb76de7654337fb6c2e851d106cebc7	
SHA-1	c59bc17659daca1b1ce65b6af077f86a648ad8a8	
SHA-256	c082a9117294fa4880d75a2625cf80f63c8bb159b54a7151553969541ac35862	

The above Cyclops Blink samples are loaded into memory as two program segments. The first of these program segments has read/execute permissions and contains the Linux ELF header and executable code for the malware. The second has read/write permissions and contains the data, including victim-specific information, used by the malware. To make the sample hashes as useful as possible for comparison purposes, they have been calculated over the executable (first) program segments only. The file sizes correspond to those of the original files.

Filename	install_upgrade	
Description	Cyclops Blink embedded ELF - Linux ELF PowerPC big-endian	
Size	964556 bytes	
MD5	3c9d46dc4e664e20f1a7256e14a33766	
SHA-1	7d61c0dd0cd901221a9dff9df09bb90810754f10	
SHA-256	4e69bbb61329ace36fbe62f9fb6ca49c37e2e5a5293545c44d155641934e39d1	

Filename	install_upgrade	
Description	Cyclops Blink embedded ELF - Linux ELF PowerPC big-endian	
Size	964556 bytes	
MD5	3f22c0aeb1eec4350868368ea1cc798c	
SHA-1	438cd40caca70cafe5ca436b36ef7d3a6321e858	
SHA-256	ff17ccd8c96059461710711fcc8372cfea5f0f9eb566ceb6ab709ea871190dc6	



MITRE ATT&CK®

This report has been compiled with respect to the MITRE ATT&CK® framework, a globally accessible knowledge base of adversary tactics and techniques based on real-world observations.

Tactic	ID	Technique	Procedure
Execution	<u>T1059.004</u>	Command and Scripting Interpreter: Unix Shell	Cyclops Blink executes downloaded files using the Linux API function execlp.
Persistence	<u>T1037.004</u>	Boot or Logon Initialization Scripts: RC Scripts	Cyclops Blink is executed on device startup, using a modified S51armled RC script.
Persistence	T1542.001	Pre-OS Boot: System Firmware	Cyclops Blink maintains persistence throughout the legitimate device firmware update process. This is achieved by patching the firmware when it is downloaded to the device.
Defence Evasion	T1562.004	Impair Defenses: Disable or Modify System Firewall	Cyclops Blink modifies the Linux iptables firewall to enable C2 communication via a stored list of port numbers.
Defence Evasion	<u>T1036.005</u>	Masquerading: Match Legitimate Name or Location	Cyclops Blink renames its running process to masquerade as a Linux kernel thread.
Discovery	<u>T1082</u>	System Information Discovery	Cyclops Blink regularly queries device information.
Command And Control	T1132.002	Data Encoding: Non- Standard Encoding	Cyclops Blink command messages use a custom binary scheme to encode the specific command to be executed, as well as any command parameters required.
Command And Control	<u>T1008</u>	Fallback Channels	Cyclops Blink randomly selects a C2 server from contained lists of IPv4 addresses and port numbers.
Command And Control	<u>T1071.001</u>	Application Layer Protocol: Web Protocols	Cyclops Blink can download files via HTTP or HTTPS.
Command And Control	T1573.002	Encrypted Channel: Asymmetric Cryptography	Cyclops Blink C2 messages are individually encrypted using AES-256-CBC and sent underneath TLS. OpenSSL library functions are used to encrypt each message using a randomly generated key and IV, which are then encrypted using a hard-coded RSA public key.
Command And Control	<u>T1571</u>	Non-Standard Port	Cyclops Blink contains a list of port numbers used for C2 communication. This list includes non-standard ports not typically associated with HTTP or HTTPS traffic.
Exfiltration	<u>T1041</u>	Exfiltration Over C2 Channel	Cyclops Blink is capable of uploading files to a C2 server.



Functionality

Overview

Cyclops Blink is a malicious Linux ELF executable, compiled for the 32-bit PowerPC (big-endian) architecture. It consists of a core component and additional modules that are executed as child processes using the Linux API function fork. Linux pipes are used for inter-process communication between the core component and modules.

Both analysed samples included the same four built-in modules that are executed on startup and provide basic malware functionality including: file upload/download, system information discovery and malware version update. Further modules can be added via tasking from a C2 server. The malware expects these modules to be Linux ELF executables that can be executed using the Linux API function <code>execlp</code>.

The malware contains a hard-coded RSA public key, which is used for C2 communications, as well as a hard-coded RSA private key and X.509 certificate. The hard-coded RSA private key and X.509 certificate do not appear to be actively used within the analysed samples, so it is possible that these are intended to be used by a separate module.

Cyclops Blink also contains an initial list of C2 server IPv4 addresses, and a hard-coded list of port numbers to use for C2 communications. The content of these lists is different for each of the analysed samples.

C2 messages include what appears to be a hard-coded ID value, which is set to 0xe2bb2797 and 0x2831bee1 in the analysed samples.

Core component

The core component starts by testing whether it is currently running as a process named [kworker:0/1]. If this is not the case then Cyclops Blink reloads itself by creating a child process, running the Linux API function execl("/proc/self/exe", ["[kworker:0/1]"], NULL), and then exiting the parent process.

At this point the malware is running as a process named [kworker:0/1]. This is masquerading as a kernel thread and has most likely been chosen to blend into the list of running processes.

Note: The Linux kernel creates a number of threads for running various system tasks e.g. scheduling, disk I/O, etc. When a process listing is viewed, using tools such as ps, these kernel threads are denoted with square brackets around them.

The core component then modifies the Linux iptables firewall to allow TCP traffic via the hard-coded list of port numbers used for C2 communications, and starts each of the four built-in modules.

Once the initialisation of the malware is complete the core component enters a main C2 loop where it:

- Receives messages containing data from running modules and queues them up ready to be sent to a C2 server.
- Beacons, consisting of queued messages, are sent to a C2 server at regular intervals. The
 intervals are specified by what appears to be a timeout variable, initially configured as 3600
 seconds.
- Decrypts and parses tasking received in response to beacons, either handling them directly or passing to the appropriate module.



Modules

Module ID 0x8 (system reconnaissance)

The purpose of this module is the discovery of system information from the WatchGuard device. The module gathers a wide variety of system information, at regular intervals, by running Linux commands and reading system files. The intervals are specified by what appears to be a module-specific timeout variable, initially configured as 600 seconds.

The gathered system information, as well as information about Cyclops Blink, is sent to the core component (where it is queued to be sent to a C2 server).

The gathered system information includes the output of the following Linux API functions:

- uname gathers name and information about the Linux kernel.
- sysinfo gathers memory statistics and swap space usage.
- statvfs gathers statistics for the filesystem containing the current working directory.
- if nameindex gathers network interface names.

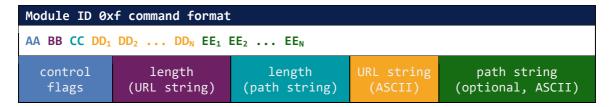
The gathered system information also includes network configuration information for the identified network interfaces, as well as the content of the following Linux system files: /etc/issue, /etc/passwd, /etc/group, /proc/mounts, /proc/partitions, /proc/net/arp.

The Cyclops Blink information includes:

- A value that appears to refer to the current version (set to 0x8036994d and 0x4ba9dc2c in the analysed samples).
- A list of the currently installed module ID values.

Module ID 0xf (file download/upload)

The purpose of this module is to enable the download and upload of files to/from the WatchGuard device. The module receives commands from the core component, formatted as follows:



Each received command is handled in a child process, thus enabling the module to handle multiple commands concurrently. The control flags are used to control the operation of the module and to indicate status, and are defined as follows:



Control flags value	Description
0x80	If this bit is set, then the download has been redirected to an absolute URL.
0x40	If this bit is set, then the download has been redirected to a relative URL.
0x20	If this bit is set, then the download has been completed.
0x10	If this bit is set, then this is an upload operation and URL string specifies a file to be uploaded to a C2 server. Otherwise, if this bit is not set, then this is a download operation.
0x8	If this bit is set, then the module downloads from the list of C2 server IPv4 addresses. Otherwise, if this bit is not set, then the module downloads from the remote server specified in URL string.
0x4	If this bit is set, then data is downloaded directly into memory and executed as shellcode. Otherwise, if this bit is not set, then data is downloaded to the file specified by path string .
0x2	If this bit is set, then the downloaded file is added to Cyclops Blink as a new module.
0x1	If this bit is set, then the downloaded file is executed as a child process.

If a download operation does not specify the **path string**, then data is written to the default location /var/tmp/a.tmp.

An upload operation reads the contents of the file specified by **URL string**. The file contents are formatted as follows and sent to the core component (where it is queued to be sent to a C2 server):

Module 0xf upload message format			
AA AA AA AA file: BB BB BB \n CC CC CC			
total size of message (bytes)	hard-coded format string	full path to uploaded file	Uploaded file contents

Module ID 0x39 (store C2 server IPv4 addresses)

The purpose of this module is to maintain the current list of C2 server IPv4 addresses on the device filesystem. When started, the module reads the current list of C2 server IPv4 addresses from the file ${\tt rootfs_cfg}$ and sends the data to the core component (where it is queued to be sent to a C2 server).

The location for the <code>rootfs_cfg</code> file is identified by searching for the first entry in <code>/proc/mounts</code> with: read/write permissions, either the <code>relatime</code> or <code>noatime</code> mount option set, and the mounted device contains either of the strings <code>/dev</code> or <code>ubi</code>.

Note: The file /proc/mounts contains a list of filesystems mounted by the WatchGuard device. The string ubi most likely refers to UBIFS, a flash filesystem for unmanaged flash memory devices.



When an updated list of C2 server IPv4 addresses is received from the core component these are written to the file rootfs cfg, which is created if it does not already exist.

Module ID 0x51 (malware update and persistence)

The purpose of this module is to update the Cyclops Blink Linux ELF executable and to maintain persistence of the malware throughout the legitimate device firmware update process. When started, the module sends the contents of the files /etc/wg/configd-hash.xml and /etc/wg/config.xml to the core component (where they are queued to be sent to a C2 server).

Note: The files /etc/wg/configd-hash.xml and /etc/wg/config.xml are legitimate WatchGuard files relating to the configuration of the WatchGuard device.

The module responds to the following commands from the core component:

Command ID	Description
0x0	Remount the root filesystem with read-only permissions.
0x1	Remount the root filesystem with read/write permissions.
0x2	Update the Cyclops Blink Linux ELF executable. The command includes the new Cyclops Blink version number and the server address from which to download the updated Linux ELF executable.
0x3	Send the contents of the file /etc/wg/configd-hash.xml to the core component (where it is queued to be sent to a C2 server).

When the update command (ID 0x2) is received, the module checks whether the new Cyclops Blink version matches the current Cyclops Blink version. If they match, then the update command is silently ignored. Otherwise, a command is sent to module ID 0xf to download the update, with command-specific data as follows:

- The URL string is set to <server address>/<Cyclops Blink version number>, where <server address> and <Cyclops Blink version number> are the values specified by the original update command received from the core component.
- The path string is set to /usr/bin/cpd.

The device filesystem is checked for the presence of the file / usr/bin/cpd every second, for 60 seconds. If the update is successfully downloaded during this period then command ID 0x0 is sent to the core component, causing the Cyclops Blink process to be terminated. The full set of command IDs supported by the core component are described in the 'Communications (Command and control)' section of this report. The new version of Cyclops Blink will then be executed on device startup via the RC script S51armled (as described in 'Cyclops Blink persistence') or must be started manually. If the download fails then the old version of Cyclops Blink will continue to run.



Cyclops Blink persistence

Cyclops Blink persistence throughout the legitimate device firmware update process is handled by a child process of module ID 0x51. Figure 1 summarises how this persistence works.

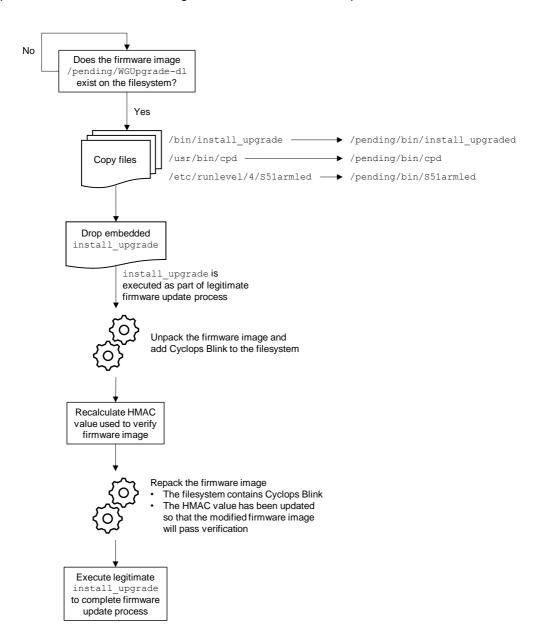


Figure 1: Cyclops Blink persistence throughout the legitimate update process



When the file /pending/WGUpgrade-dl is found on the device filesystem the module first copies the following files:

- The file /bin/install upgrade is copied to /pending/bin/install upgraded
- The file /usr/bin/cpd (the Cyclops Blink executable) is copied to /pending/bin/cpd
- The file /etc/runlevel/4/S51armled is copied to /pending/bin/S51armled

Note: The file /pending/WGUpgrade-dl is the legitimate updated firmware image to be applied to a WatchGuard device. The file /bin/install_upgrade is a legitimate WatchGuard executable that is responsible for installing an upgraded device firmware image. The file /etc/runlevel/4/S51armled is an RC script executed during device startup and is most likely used to execute Cyclops Blink.

The module then overwrites the legitimate /bin/install_upgrade executable with a Linux ELF executable embedded within Cyclops Blink. This modified version of /bin/install_upgrade is then used to install the upgraded device firmware image.

The modified version of <code>/bin/install_upgrade</code> unpacks the firmware image <code>/pending/WGUpgrade-dl</code>. The Cyclops Blink executable and the RC script <code>S5larmled</code> are added to the firmware image. The access and modification timestamps for these files are changed, using the <code>Linux API</code> function <code>utime</code>, to match those of <code>/pending/sysa_code_dir/bin/busybox-rel</code>. The HMAC value, used to verify the firmware image, is recalculated to take account of the added files. Finally, the modified firmware image is repacked ready for installation.

Note: It is possible to recalculate the HMAC value for the modified firmware image because the WatchGuard FireBox devices use a hard-coded key to initialise the hash calculation.

Once the modified firmware image has been repacked, it is installed using the legitimate $\begin{tabular}{l} \begin{tabular}{l} \begin{tabular}{l$



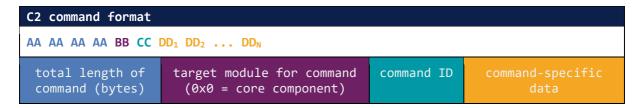
Communications

Command and control

Cyclops Blink uses OpenSSL (version 1.0.1f) to support C2 communication underneath TLS. Each time the malware beacons it randomly selects a destination from the current list of C2 server IPv4 addresses and hard-coded list of C2 ports.

Beacons consist of queued messages containing data from running modules. Each message is individually encrypted using AES-256-CBC. The <code>OpenSSL_EVP_SealInit</code> function is used to randomly generate the encryption key and IV for each message, and then encrypt them using the hard-coded RSA public key.

The <code>OpenSSL_RSA_public_decrypt</code> function is used to decrypt tasking, received in response to beacons, using the hard-coded RSA public key. The decrypted command is expected to be formatted as follows:



The core component responds to the following commands:

Command ID	Description	
0x0	Terminate the process running Cyclops Blink.	
0x1	Force a beacon to be sent to a C2 server on the next iteration of the main C2 loop.	
0x2	Update the list of C2 server IPv4 addresses. The command-specific data contains the number of C2 server IPv4 addresses, followed by the C2 server IPv4 addresses in binary format.	
0x3	Set the time at which the next beacon will be sent. The command-specific data contains the time (number of seconds since the epoch) as a 4-byte value.	
0x4	Set the beaconing interval. The command-specific data contains the beaconing interval (seconds) as a 4-byte value.	
0x5	Add a new module to Cyclops Blink. The command-specific data contains the path to a Linux ELF executable that will be loaded.	
0x6	Restart Cyclops Blink.	
0x7	Set an unknown 4-byte value.	
0x8 – 0xa	Resend the current Cyclops Blink configuration to all running modules.	
0xb	Send the hard-coded RSA public key to a specified module. The command- specific data contains the module ID to which the data should be sent.	
Охс	Send the hard-coded RSA private key to a specified module. The command- specific data contains the module ID to which the data should be sent.	
0xd	Send the hard-coded X.509 certificate to a specified module. The command- specific data contains the module ID to which the data should be sent.	



Conclusion

Cyclops Blink appears to have been professionally developed, given its modular design approach. A comparison of the core component functionality between the analysed samples indicates that they have most likely been developed from a common code base.

A significant amount of attention has been given to ensuring that the C2 communications are difficult to detect and track (for example, use of TLS, AES-256-CBC encryption, multiple redundant C2 servers etc.).

The developers have clearly reverse engineered the WatchGuard Firebox firmware update process and have identified a specific weakness in this process, namely the ability to recalculate the HMAC value used to verify a firmware update image. They have taken advantage of this weakness to enable them to maintain the persistence of Cyclops Blink throughout the legitimate firmware update process.

It is of note that Cyclops Blink has read/write access to the device filesystem, enabling legitimate files to be replaced with modified versions (e.g. install_upgrade). Even if the specific weakness highlighted above were fixed, it is expected that the developers would be capable of deploying new capability to maintain the persistence of Cyclops Blink.

These factors, combined with the professional development approach, lead to the NCSC conclusion that Cyclops Blink is a highly sophisticated piece of malware.

Whilst the samples of Cyclops Blink described in this report have been compiled for the 32-bit PowerPC (big-endian) architecture, WatchGuard devices cover a wide range of architectures, and it is highly likely that these are also targeted by the malware. The weakness in the firmware update process is also highly likely to be present in other WatchGuard devices. It is therefore recommended that users follow the WatchGuard mitigation advice for all relevant devices.



Detection

Indicators of compromise

Туре	Description	Values
Path	Path location of Cyclops Blink executable	/usr/bin/cpd
Path	Path location to backed-up legitimate install_upgrade executable	/pending/bin/install_upgraded
Path	Default path location for downloaded files	/var/tmp/a.tmp
Filename	Name of file used to persist C2 server IP addresses on the device filesystem	rootfs_cfg
IPv4 address	C2 server IP address	100.43.220[.]234
IPv4 address	C2 server IP address	96.80.68[.]193
IPv4 address	C2 server IP address	188.152.254[.]170
IPv4 address	C2 server IP address	208.81.37[.]50
IPv4 address	C2 server IP address	70.62.153[.]174
IPv4 address	C2 server IP address	2.230.110[.]137
IPv4 address	C2 server IP address	90.63.245[.]175
IPv4 address	C2 server IP address	212.103.208[.]182
IPv4 address	C2 server IP address	50.255.126[.]65
IPv4 address	C2 server IP address	78.134.89[.]167
IPv4 address	C2 server IP address	81.4.177[.]118
IPv4 address	C2 server IP address	24.199.247[.]222
IPv4 address	C2 server IP address	37.99.163[.]162
IPv4 address	C2 server IP address	37.71.147[.]186
IPv4 address	C2 server IP address	105.159.248[.]137
IPv4 address	C2 server IP address	80.155.38[.]210
IPv4 address	C2 server IP address	217.57.80[.]18
IPv4 address	C2 server IP address	151.0.169[.]250
IPv4 address	C2 server IP address	212.202.147[.]10
IPv4 address	C2 server IP address	212.234.179[.]113
IPv4 address	C2 server IP address	185.82.169[.]99
IPv4 address	C2 server IP address	93.51.177[.]66
IPv4 address	C2 server IP address	80.15.113[.]188
IPv4 address	C2 server IP address	80.153.75[.]103
IPv4 address	C2 server IP address	109.192.30[.]125



Rules and signatures

Description	Detects notable strings identified within the Cyclops Blink executable	
Precision	No false positives have been identified during VT retrohunt queries	
Rule type	YARA	

```
rule CyclopsBlink notable strings
    meta:
           author = "NCSC"
          description = "Detects notable strings identified within the Cyclops
Blink executable"
           hash1 = "3adf9a59743bc5d8399f67cab5eb2daf28b9b863"
          hash2 = "c59bc17659daca1b1ce65b6af077f86a648ad8a8"
           // Process names masqueraded by implant
           $proc name1 = "[kworker/0:1]"
           $proc name2 = "[kworker/1:1]"
           // DNS query over SSL, used to resolve C2 server address
           $dns query = "POST /dns-query HTTP/1.1\x0d\x0aHost:
dns.google\x0d\x0a"
           // iptables commands
           $iptables1 = "iptables -I %s -p tcp --dport %d -j ACCEPT &>/dev/null"
           $iptables2 = "iptables -D %s -p tcp --dport %d -j ACCEPT &>/dev/null"
           // Format strings used for system recon
           $sys recon1 = "{\"ver\":\"%x\",\"mods\";["
           $sys_recon2 = "uptime: %lu mem_size: %lu mem free: %lu"
           $sys_recon3 = "disk_size: %lu disk free: %lu"
           $sys_recon4 = "hw: \(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}\)02x:\(\frac{1}{6}
           // Format string for filepath used to test access to device
filesystem
           $testpath = "%s/214688dsf46"
           // Format string for implant configuration filepath
           $confpath = "%s/rootfs cfg"
           // Default file download path
           $downpath = "/var/tmp/a.tmp"
     condition:
           (uint32(0) == 0x464c457f) and (8 of them)
```



DescriptionDetects the code bytes used to initialise the modules built into Cyclops BlinkPrecisionNo false positives have been identified during VT retrohunt queriesRule typeYARA



Description

Detects notable strings identified within the modified install_upgrade executable, embedded within Cyclops Blink

Precision

No false positives have been identified during VT retrohunt queries

Rule type

```
rule CyclopsBlink modified install upgrade
 meta:
   author = "NCSC"
   description = "Detects notable strings identified within the modified
install upgrade executable, embedded within Cyclops Blink"
   hash1 = "3adf9a59743bc5d8399f67cab5eb2daf28b9b863"
   hash2 = "c59bc17659daca1b1ce65b6af077f86a648ad8a8"
   hash3 = "7d61c0dd0cd901221a9dff9df09bb90810754f10"
   hash4 = "438cd40caca70cafe5ca436b36ef7d3a6321e858"
  strings:
    // Format strings used for temporary filenames
    $ = "/pending/%010lu %06d %03d p1"
    $ = "/pending/sysa code_dir/test_%d_%d_%d_%d_%d_%d"
    // Hard-coded key used to initialise HMAC calculation
    $ = "etaonrishdlcupfm"
    \ensuremath{//} Filepath used to store the patched firmware image
    $ = "/pending/WGUpgrade-dl.new"
    // Filepath of legitimate install_upgrade executable
    $ = "/pending/bin/install upgraded"
    // Loop device IOCTL LOOP SET FD
    $ = {38 80 4C 00}
    // Loop device IOCTL LOOP_GET_STATUS64
    $ = {38 80 4C 05}
    // Loop device IOCTL LOOP SET STATUS64
    $ = {38 80 4C 04}
    // Firmware HMAC record starts with the string "HMAC"
    $ = {3C 00 48 4D 60 00 41 43 90 09 00 00}
  condition:
    (uint32(0) == 0x464c457f) and (6 of them)
```



Description

Detects the code bytes used to test the command ID being sent to the core component of Cyclops Blink

Precision

No false positives have been identified during VT retrohunt queries

Rule type

YARA

```
rule CyclopsBlink core command check
 meta:
   author = "NCSC"
   description = "Detects the code bytes used to test the command ID
being sent to the core component of Cyclops Blink"
   hash1 = "3adf9a59743bc5d8399f67cab5eb2daf28b9b863"
   hash2 = "c59bc17659daca1b1ce65b6af077f86a648ad8a8"
  strings:
    // Check for command ID equals 0x7, 0xa, 0xb, 0xc or 0xd
    $cmd check = {81 3F 00 18 88 09 00 05 54 00 06 3E 2F 80 00
                  (07|0A|0B|0C|0D)}
  condition:
    (uint32(0) == 0x464c457f) and (\#cmd check == 5)
```

Description

Detects the initial characters used to identify Cyclops Blink configuration data

Precision

No false positives have been identified during VT retrohunt queries

Rule type

```
rule CyclopsBlink config identifiers
 meta:
    author = "NCSC"
    description = "Detects the initial characters used to identify
Cyclops Blink configuration data"
    hash1 = "3adf9a59743bc5d8399f67cab5eb2daf28b9b863"
   hash2 = "c59bc17659daca1b1ce65b6af077f86a648ad8a8"
  strings:
    // Main config parameter data starts with the string "<p: "
    $ = "<p: " fullword
    // RSA public key data starts with the string "<k: "
    $ = {3C 00 3C 6B 60 00 3A 20 90 09 00 00}
    // X.509 certificate data starts with the string "<c: "
    $ = {3C 00 3C 63 60 00 3A 20 90 09 00 00}
    // RSA private key data starts with the string "<s: "
    $ = {3C 00 3C 73 60 00 3A 20 90 09 00 00}
  condition:
    (uint32(0) == 0x464c457f) and (all of them)
}
```



Description

Detects the code bytes used to check module ID 0xf control flags and a format string used for file content upload

Precision

No false positives have been identified during VT retrohunt queries

Rule type

```
rule CyclopsBlink handle mod 0xf command
 meta:
    author = "NCSC"
    description = "Detects the code bytes used to check module ID 0xf
control flags and a format string used for file content upload"
    hash1 = "3adf9a59743bc5d8399f67cab5eb2daf28b9b863"
    hash2 = "c59bc17659daca1b1ce65b6af077f86a648ad8a8"
  strings:
    // Tests execute flag (bit 0)
    $ = {54 00 06 3E 54 00 07 FE 54 00 06 3E 2F 80 00 00}
    // Tests add module flag (bit 1)
    $ = {54 \ 00 \ 06 \ 3E \ 54 \ 00 \ 07 \ BC \ 2F \ 80 \ 00 \ 00} 
    // Tests run as shellcode flag (bit 2)
    $ = {54 \ 00 \ 06 \ 3E \ 54 \ 00 \ 07 \ 7A \ 2F \ 80 \ 00 \ 00} 
    // Tests upload flag (bit 4)
    $ = {54 \ 00 \ 06 \ 3E \ 54 \ 00 \ 06 \ F6 \ 2F \ 80 \ 00 \ 00}
    // Upload format string
    $ = "file:%s\n" fullword
  condition:
    (uint32(0) == 0x464c457f) and (all of them)
```



DescriptionDetects the code bytes used to set default Cyclops Blink configuration valuesPrecisionNo false positives have been identified during VT retrohunt queries

Rule type YARA

```
rule CyclopsBlink default config values
 meta:
    author = "NCSC"
    description = "Detects the code bytes used to set default Cyclops
Blink configuration values"
    hash1 = "3adf9a59743bc5d8399f67cab5eb2daf28b9b863"
    hash2 = "c59bc17659daca1b1ce65b6af077f86a648ad8a8"
  strings:
    // Unknown config value set to 0x19
    $ = {38 \ 00 \ 00 \ 19 \ 90 \ 09 \ 01 \ A4}
    // Unknown config value set to 0x18000
    $ = {3C 00 00 01 60 00 80 00 90 09 01 A8}
    // Unknown config value set to 0x4000
    $ = {38 \ 00 \ 40 \ 00 \ 90 \ 09 \ 01 \ AC}
    // Unknown config value set to 0x10b
    $ = {38 \ 00 \ 01 \ 0B \ 90 \ 09 \ 01 \ B0}
    // Unknown config value set to 0x2711
    $ = {38 \ 00 \ 27 \ 11 \ 90 \ 09 \ 01 \ C0}
  condition:
    (uint32(0) == 0x464c457f) and (3 of them)
}
```



Description

Detects the code bytes used to check commands sent to module ID 0x51 and notable strings relating to the Cyclops Blink update process

Precision

No false positives have been identified during VT retrohunt queries

Rule type

```
rule CyclopsBlink handle mod 0x51 command
 meta:
    author = "NCSC"
    description = "Detects the code bytes used to check commands sent to
module ID 0x51 and notable strings relating to the Cyclops Blink update
process"
    hash1 = "3adf9a59743bc5d8399f67cab5eb2daf28b9b863"
    hash2 = "c59bc17659daca1b1ce65b6af077f86a648ad8a8"
  strings:
    // Check for module command ID equals 0x1, 0x2 or 0x3
    \ cmd check = {88 1F [2] 54 00 06 3E 2F 80 00 (01|02|03)}
    // Legitimate WatchGuard filepaths relating to device configuration
    $path1 = "/etc/wg/configd-hash.xml"
    $path2 = "/etc/wg/config.xml"
    // Mount arguments used to remount root filesystem as RW or RO
    mnt arg1 = "ext2"
    $mnt arg2 = "errors=continue"
    mnt_{arg3} = {38 C0 0C 20}
    mnt arg4 = {38 C0 OC 21}
  condition:
    (uint32(0) == 0x464c457f) and (\#cmd check == 3) and
    ((@cmd check[3] - @cmd check[1]) < 0x200) and
    (all of ($path*)) and (all of ($mnt arg*))
```



Disclaimer

This report draws on information derived from NCSC and industry sources. Any NCSC findings and recommendations made have not been provided with the intention of avoiding all risks and following the recommendations will not remove all such risk. Ownership of information risks remains with the relevant system owner at all times.

This information is exempt under the Freedom of Information Act 2000 (FOIA) and may be exempt under other UK information legislation.

Refer any FOIA queries to ncscinfoleg@ncsc.gov.uk.

All material is UK Crown Copyright ©