National Cyber Security Centre
a part of GCHQ

Malware Analysis Report

UMBRELLA STAND

Malware targeting Fortinet devices

Version 1.0

18th June 2025

UMBRELLA STAND

Malware targeting Fortinet devices

Executive summary

- UMBRELLA STAND beacons to its C2 server using fake TLS on port 443
- UMBRELLA STAND has functionality to run shell commands
- UMBRELLA STAND has been observed targeting FortiGate 100D series firewalls produced by Fortinet
- UMBRELLA STAND has command configurable C2 and beacon frequency.
- UMBRELLA STAND has been observed deployed in conjunction with a set of publicly available tooling including: `BusyBox`, `nbtscan`, `tcpdump` and `openLDAP`
- UMBRELLA STAND is able to hook the reboot functionality of the device

Introduction

UMBRELLA STAND is a collection of actor binaries likely to be deployed by exploiting security vulnerabilities in the target device. It is believed that deployment is intended to facilitate long-term access into a given target network.

UMBRELLA STAND contains:

- remote shell execution functionality
- configurable beacon frequency
- configurable C2 IP address
- AES-Encrypted C2 communications

UMBRELLA STAND has been observed targeting internet-facing, FortiGate 100D series firewalls produced by Fortinet. Several of the tools described in the COATHANGER report[1] were also present on this device. A previously unknown piece of malware - SHOE RACK - was also identified on the device. SHOE RACK is beyond the scope of this report but is referenced in the Functionality (Associated tooling) section.

[1] https://www.ncsc.nl/binaries/ncsc/documenten/publicaties/2024/februari/6/mivd-aivd-advisory-coathanger-tlp-clear/TLP-CLEAR+MIVD+AIVD+Advisory+COATHANGER.pdf

Malware details

Metadata

Filename	blghtd
Description	UMBRELLA STAND – networking binary.
Size	873,696 bytes
MD5	79c8db07491b9049921b060ee059877a
SHA-1	8632487a9f19223b20d34c082a3077ca6ac6eac4
SHA-256	8bacd5df99476328321a7e8e2fc0124c20f7a7ebf3e8f151c050387038515b70

Filename	jvnlpe
Description	UMBRELLA STAND - used to run and monitor blghtd.
Size	857,024 bytes
MD5	ff82ea16717c6fa9fc8c07ae2de09c5f
SHA-1	3294257f9085a727a7723885e8431e8a036d3082
SHA-256	591d60c1d356da827a26f4141fa431d3663af91746d5371014695b1c89bac2b2

Filename	cisz
Description	UMBRELLA STAND - responsible for initialisation, launching injection, starting jvnlpe.
Size	10,624 bytes
MD5	586856B41EB0F101924CAEEAD5AEB4F0
SHA-1	7931678cdd0d143ca98c18a00b8d237583fa8d93
SHA-256	6a3abc19f324a475d4ce01fcc69797fc90e1a47970ed90e9cb01f540f3000b4e

Filename	libguic.so[2]
Description	UMBRELLA STAND - loaded using ldpreload and used to load cisz.
Size	12,292 bytes
MD5	6bdfac0500e164de8c6bbd13e05b3968
SHA-1	3a7d09c9ce2ff7f6026cda4a7f80945ee0952c95
SHA-256	190293440fce95f45eb8bf5d40334b41dd68c79578d06fe9b34670298daea7f3

Filename	*unknown* – for ease of reference this will be referred to as reboot_hooker
Description	UMBRELLA STAND - used to hook the reboot of the device.
Size	24,574 bytes
MD5	43f398bfe3eecd5584ebf3321000c3cd
SHA-1	73e2ae5b03e20eaa4e2cafb2000d57321bfa0b5f
SHA-256	a64b41e98e3e1066f41fbff5d4f99f6d34b792d35fe2be7e5d9fa8f3f8b93739

[2] Likely filename, derived from other artefacts on the device.

Filename	a
Description	UMBRELLA STAND - command line tool used to encrypt and decrypt files.
Size	849,056 bytes
MD5	615521d36644d11f53e17805a1422f29
SHA-1	d21e46856ffb344ed06a461efb554e5a490a9e3e
SHA-256	d1d5f502e2039b20269b562bbc1e5622a73bbecad54cb25ae5eaa7a91504e70e

Filename	lidwok
Description	Version 1.3.11 of BusyBox - used by the actors.
Size	2,765,328 bytes
MD5	d07a6070e4cee1716fc882bc1d51b7bc
SHA-1	99dde2df0b8b31fce5807d710c1b8d9018a56f58
SHA-256	d3b88b7f640e478d8d875e12b4561e8c794909e4954aebbc6fd1f5e79f381648

Filename	dskz
Description	Tool used to inject libraries into other processes - used by the actors.
Size	14,640 bytes
MD5	821B1158AA87A055030C99B93026485A
SHA-1	c8183d12c2070cf04cd03f080904ed1312a56911
SHA-256	65f1e17f7fa2e2fd9c57265f390484a7428c192f59ee41fc7c0d8386ea3b811a

Filename	tdp
Description	tcpdump version 4.99.1 (released June 2021) using libpcap version 1.10.1 (released June 2021) - used by the actors.
Size	2,571,376 bytes
MD5	c8c20c56c950eb291ce7902bf1c28485
SHA-1	c2a463d5091efb2be590fbfa5dba5a821d5625cf
SHA-256	38801caae26916367dd6cf6e8c55e50ed62526fe242cd0343dfe80a70564c28a

Filename	nbso
Description	Version 1.0.35 of nbtscan - used by the actors.
Size	133,256 bytes
MD5	50664a15bce874e53cdc99c220a10aab
SHA-1	28ff882baa02c646bcdddffacb75923490a3dcf7
SHA-256	881998c9864d2c7fe35f9b8071dbcf84386cb15da77e6f6a086cf605a4dd7823

Functionality

Overview

UMBRELLA STAND is a set of tooling designed to allow shell commands to be run on compromised embedded devices. The actors use it in conjunction with public domain utilities also deployed to the victim device. A description of the tooling is described in the Functionality (Associated tooling) section.

UMBRELLA STAND is composed of a large number of interconnected components.

A diagram of the interactions between known components is shown below:

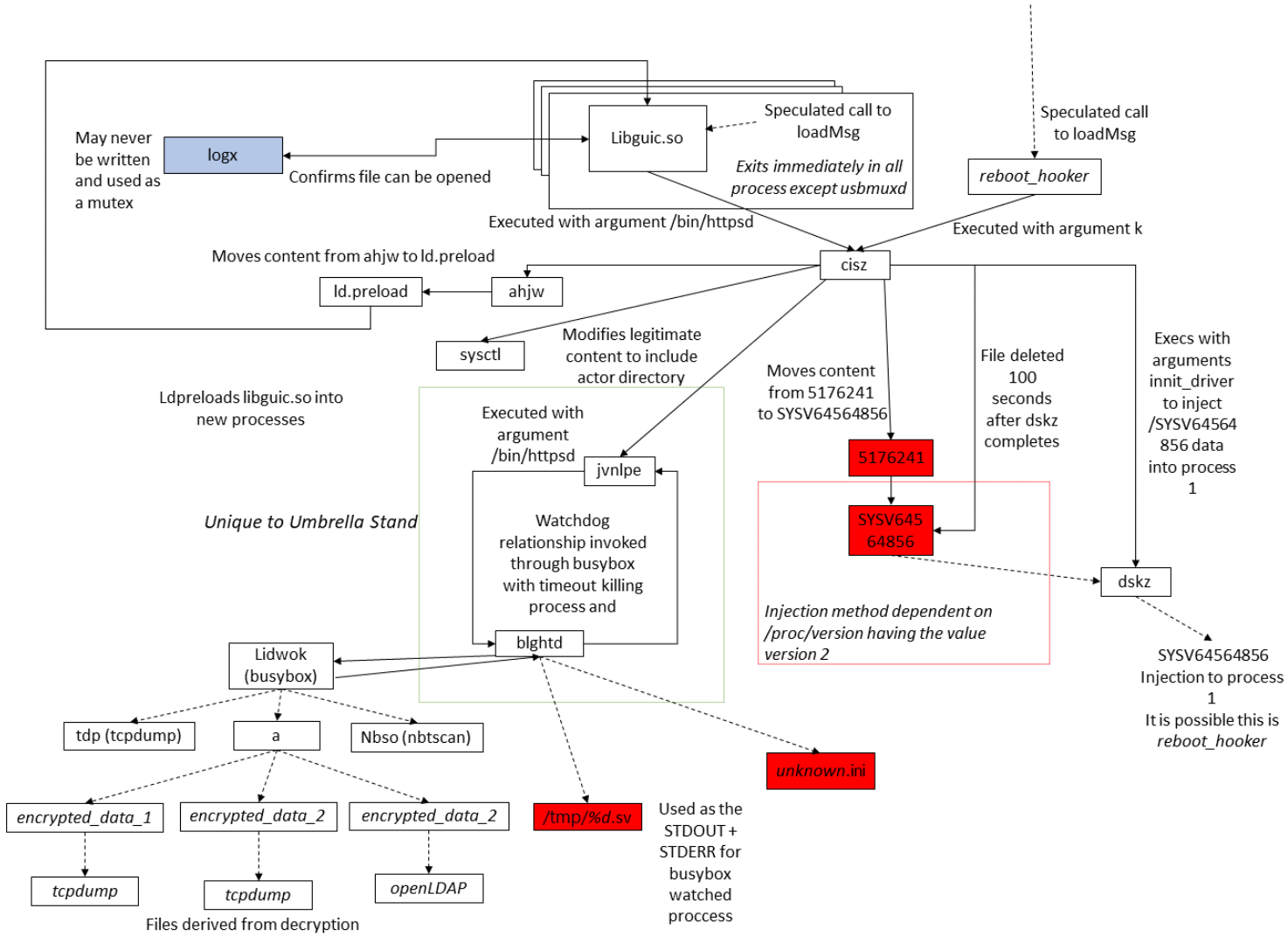

Figure 1: shows the different components of UMBRELLA STAND. Red shows files that were not recovered.

Associated tooling

Filename	File source	Purpose
`jvlpe`	File system `.ztls` directory	This is actor tooling used as a watchdog to ensure the main `blghtd` binary is running.
`blghtd`	File system `.ztls` directory	This is the main actor tooling responsible for networking and processing tasking.
`a`	File system `.ztls` directory	This is actor tooling used for encrypting and decrypting files. It is described in more detail in the Functionality (File encryptor) section.
`cisz`	Unallocated space	This is actor tooling used to initialise setup.
`libguic.so`	Unallocated space	This is actor tooling loaded using `ldpreload` into each process running on the device. If the process is `usbmux` it will execute `cisz`, otherwise, it will exit.
unknown	Unallocated space	This is actor tooling used to hook the reboot function of the device. Unfortunately, the file name was not recovered, so it will be referred to as `reboot_hooker`.
unknown	Unallocated space, data encrypted by the file encryptor `a`.	Partial file recovery suggests that this is `openLDAP`. An exact overlap between this file and public files has not been seen.
`ldnet`	File system `.legacy` directory	This tooling is not directly linked and a description of the functionality of this binary is beyond the scope of this report. It is referred to as SHOE RACK.
`dskz`	Unallocated space	This is used to carry out process injection.
`tdp`	File system `.ztls` directory	This is version `4.99.1` of `tcpdump` with `libpcap` version `1.10.1` (with `TPACKET_V3`) Evidence of this having been run by the actor and associated traffic capture was also identified on the device. Multiple versions of this file were identified on the device, encrypted by the file encryptor `a`.
`nbso`	File system `.ztls` directory	This is version `1.0.35` of `nbtscan`.

`lidwok`	File system `.ztls` directory	This is version `1.3.11` of `BusyBox` (`2022-09-13 17:36:18 PDT`) multi-call binary. A list of commands implemented can be found in the Appendix (BusyBox functionality).

Note: *tcpdump is a commonly used network traffic capture tool. Source code can be found at www.github.com/the-tcpdump-group/tcpdump*

nbtscan is a scanning tool used to identify NetBIOS name information. Source code can be found at https://github.com/charlesroelli/nbtscan

OpenLDAP is an open-source implementation of Lightweight Directory Access Protocol (LDAP). For more information see https://www.openldap.org

---

It is suspected that the majority of the actor specific tooling was deployed at the same time, with additional command line utilities being deployed at a later point. How these tools were deployed is not understood. Potential avenues could include the use of SHOE RACK or creating a file transfer service using shell commands.

Process injection

`dskz` is a tool used to inject shared objects into a specified process. It has the following help string:

```
usage: %s [-n process-name] [-p pid] [library-to-inject]
```

It is used by `cisz` with the following hardcoded command line:

```
dsckz init_drivercheck -p 1 /SYSV664564856
```

This will inject `SYSV664564856` into the process with a process id (pid) of `1`. This is the `init` process. A file with the name `SYSV664564856` was not recovered from the device but it is possible that this is the same file referred to as *reboot_hooker*.

Watchdog

`jvnlpe` acts as a watchdog for `blghtd` (the binary responsible for networking). If `blghtd` is not running, then it is started by `jvnlpe`. Every ~9 days, `jvnlpe` will use `BusyBox killall blghtd`, to terminate `blghtd` and then restart it.

It should be noted that restarting `blghtd` will reset the values of the beacon IP and any configured sleep time as described in the Functionality (Commands) section.

File encryptor

The binary `a` is a command line tool that AES encrypts files. An example of its command line is shown below:

```
a <input_file> <flag> <hex_key> <output_file>
```

The `<input_file>` is read from disk and AES encrypted (`<flag>` `1`) or decrypted (`<flag>` `2`) using the hex string provided on the command line, with the following IV:

```
67 C6 69 73 51 FF 4A EC 29 CD BA AB F2 FB E3 46
```

The key must be an even number of bytes in length, it should be noted that if an invalid hex character is used, it is replaced with a `0` nibble.

The result is then written to the `<output_file>`, and the IV is prepended to the encrypted data as follows:

File encryptor example output

```
67 C6 69 73 51 FF 4A EC 29 CD BA AB F2 FB E3 46
A0 9C 6F D9 41 2A C0 18 4A F6 D2 08 F2 E0 AA 43
```

Fixed IV	Encrypted data

Three chunks of encrypted data and associated keys were identified. The keys identified were short, low-entropy keys.

We were able to use these values to decrypt the data and identify two copies of the same file `tdp` (`tcpdump`) and an initial file chunk believed to be from the open-source tool `openLDAP`. This may suggest these additional utilities are deployed per use and deleted when no longer required.

Malware ID

A CRC32 of the hostname is generated then endian flipped. This value is used consistently in communications with the C2. It is suspected that this value is

used as a unique ID for the C2 to use to keep track of the different malware instances it can task.

As an example, the CRC32 of `debian` is `73b0cd4c`, so a value of `4ccdb073` would therefore be used in communications with the C2 server.

Sleep pattern

UMBRELLA STAND will sleep for 11 seconds the first 30 times it beacons. Following this, it will use the configured sleep time. It should be noted that 2 is added to any value configured by tasking.

If a configured value is not set, a default value of 2145 seconds (~35 minutes) is used.

Commands

UMBRELLA STAND possesses functionality to run commands on an infected device. The available `<commandid>` values are outlined in the following Functionality (Command IDs) section.

Command IDs

Command IDs are grouped together into sets of 10, the decimal values of which are shown in the table below. The functionality of each command ID in the group of 10 is the same unless specifically elaborated.

Command ID (decimal)	Behaviour	Description
00 – 09	Default case	Keep-alive beacon to configured IP address.
10 - 19	Execute shell command	Executes a shell command as described in: Functionality (Commands) - First shell implementation.
20 - 29	Read file	Reads a file from disk as described in: Functionality (Commands) - Read file.
30, 32 - 39	Read file execute chunk	This command is used to mark a chunk as collected and to identify the next chunk to read as described in: Functionality (Commands) - Second shell implementation.
31	Execute shell command	Executes a shell command as described in: Functionality (Commands) - Second shell implementation.

40 - 49	Register device	This command returns the CRC32 and the hostname of the compromised device.
50 - 59	Configure sleep time	This command overrides the default sleep time. This will only apply after the initial beacon patten has completed as described in Functionality (Sleep pattern). This change is not persistent if the binary is restarted.
60 - 69	Configure C2 IP address	This command overrides the default beacon IP. This change is not persistent if the binary is restarted.
70+	Same as default case.	Signifies a beacon to the configured IP address in the same way as the default case.

Read file

UMBRELLA STAND can read the contents of a file specified in a command message; this read file command is requested with the following parameters:

```
<commandid> <file> <offset>
```

This command will collect up to 6000-bytes from a `<file>` file starting at `<offset>`. To collect larger files this command would need to be called multiple times with the same `<file>` and an incrementing `<offset>` value.

Shell commands

UMBRELLA STAND has two different commands that provide a remote shell, the intended use of these is not known.

Note: *It is speculated that one is intended to directly pass shell commands and the other to run executables on the device.*

All shell commands are invoked using both the `ash` shell and the `BusyBox` binary (renamed to '`lidwok`') as follows:

```
ash -c lidwok timeout 900 <command>
```

This provides the actor with a `BusyBox` shell environment that will automatically stop long running commands (900 seconds+). This is likely a safety mechanism designed to prevent the actor from unintentionally

creating long running tasks that could be noticed by an admin or cause the malware to become unresponsive.

First shell implementation
This command is invoked with the following parameters:

```
<commandid> <filename> <offset> <command>
```

The `<filename>` is a decimal value used as the file name of the result file at the path `/tmp/%d.sv`.

The `<offset>` is the offset into the result file. If this is zero the `<command>` will be executed on the victim device with the results written into the results file. The malware will sleep for 2 seconds before reading and exfiltrating up to 6000-bytes from the results file. If the `<offset>` value was not zero, no execution occurs and instead, the malware reads and exfiltrates up to 6000-bytes starting from `<offset>` in the results file.

Once all data has been read from the results file, the malware will delete it.

This means that this command would likely be called multiple times per execution to ensure all output data is collected and cleared.

Second shell implementation
The second command implementation is invoked with the following parameters:

```
<commandid> <chunk> <command> <max_collect_size>
```

Any output generated is expected to be produced by the executable being run e.g. `tcpdump -w output`.

This command set is split into two behaviours – which will be referred to as ‘execute’ and ‘mark collected’.

When running as ‘execute’, the requested `<command>` is executed and the `<max_collect_size>` is used to initialise a collection information file, which is written too and read from `/data2/<command>.ini`.

In this file, each line contains a chunk number, an equals sign, and a 0 or 1 corresponding to a ‘collected’ or ‘uncollected’ state, respectively. An example of the content of the `.ini` file is shown below:

```
0=1

1=1

2=0

3=0
```

The 'mark collected' mode marks the requested chunk as collected in the `.ini` file and returns the next chunk number to be collected. This command does not itself return any of the output data written by the requested command. Instead, it is expected that each chunk would be collected using the read file command.

It is speculated that the actor would run the 'execute' file command then:

- Run the read command to read the first chunk of the `output` file.
- Run the 'mark collected' command to set the chunk as read and identify the number of the next chunk.
- Run the read command to collect the next chunk, and so on.

Once this reaches the final entry in the `ini` file, the `ini` file itself will be deleted.

Note: *The <max_collect_size> only defines how much data will be collected by the actor. If there is more data output by the command, this will not be returned and the file will still be deleted once <max_collect_size> is reached.*

It is unclear why the actor would choose to implement capability in this way. One possibility is that it allows them to see chunks of a command's result before it has finished running. Alternatively, it may be to support asynchronous file collection in the future, though there is no evidence of this at present.

Message structure

The message structure sent by the malware and the controller is shared.

Initial beacon

Example beacon

```
4C CD B0 73 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 F4 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
```

Malware ID	Command ID	Unused
Unused	Unused	Unused
Static bytes	Unused	Padding

Note, the values that are designated 'Unused' in the 'Example beacon' may be populated on a per command basis in response to tasking.

C2 response data

The C2 appears to respond with the following default tasking structure, regardless of the traffic that it receives. More information about this can be found in the Communications (Command and Control) section.

C2 server response

```
19 6C ED 06 6A 80 F1 38 B8 2E 88 85 C8 56 F1 47
00 00 00 00 28 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 F4 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F
```

Random data	Malware ID	Command ID/Error code
Filename	Endian flipped file length	Current chunk
Optional argument length	Static bytes	Optional argument
Padding		

The filename is a decimal value to be used as the output file as described in Functionality (Commands) - First shell implementation.

Persistence

The persistence mechanism for UMBRELLA STAND has not been captured in its entirety. Functionality that could enable persistence has been observed in two different places.

Reboot hooking

UMBRELLA STAND identifies and overwrites the `reboot` function of the Fortinet operating system. When the hooked function is called, the UMBRELLA STAND loading binary (`cizs`) is executed. It is possible that this binary is injected into the `init` process (PID 1) and that process is responsible for rebooting the device, though this has not been confirmed.

ldpreload

The presence of the string `/data2/.ztls/libguic.so` in `/etc/ld.so.preload` suggests that persistence is achieved through using `ldpreload` to load `libguic.so` into a new process, this then loads the other components.

Defence evasion

Hidden Folder

UMBRELLA STAND and associated tooling make use of the hidden folder `/data2/.ztls/`. This is likely to reduce the chances of being noticed on the device.

Naming

UMBRELLA STAND and associated tooling make use of generic sounding filenames that could plausibly be found on Linux systems. This is likely an attempt to blend in with files expected to be present on the device.

String encryption

UMBRELLA STAND makes use of AES encrypted stack strings. This is done by the actor to hinder detection and analysis.

The C2 IP is obfuscated in a similar way but the obfuscated string is not stored on the stack. It is possible that this is to allow the IP to be patched in post compilation without having to rebuild the binary.

Process impersonation

`blghtd` overwrites the name of the executable present on its command line with a hardcoded actor string `/bin/httpsd`. This means that in process listings the process name does not matchup with the executable in use. Any difference in length between executable name and the fake string, is filled with null bytes. This is more noticeable if a command line argument is also present, as shown below.

Before renaming:

```
root@debian:/home/user/Desktop# ps -aux | grep 21844
root     21844  0.0  0.0   1160   140 pts/1    t    13:08   0:00 /data2/.ztls/blghtd example_arg_string
root     21856  0.0  0.0  12784  1016 pts/0    S+   13:10   0:00 grep 21844
root@debian:/home/user/Desktop# readlink /proc/21844/exe
/data2/.ztls/blghtd
root@debian:/home/user/Desktop# xxd /proc/21844/cmdline
00000000: 2f64 6174 6132 2f2e 7a74 6c73 2f62 6c67  /data2/.ztls/blg
00000010: 6874 6400 6578 616d 706c 655f 6172 675f  htd.example_arg_
00000020: 7374 7269 6e67 00                        string.
root@debian:/home/user/Desktop# cat /proc/21844/cmdline
/data2/.ztls/blghtdexample_arg_stringroot@debian:/home/user/Desktop#
```

After renaming:

```
root@debian:/home/user/Desktop# ps -aux | grep 21844
root     21844  0.0  0.0   1160   140 pts/1    t    13:08   0:00 /bin/httpsd        example_arg_string
root     21868  0.0  0.0  12784   948 pts/0    S+   13:16   0:00 grep 21844
root@debian:/home/user/Desktop# readlink /proc/21844/exe
/data2/.ztls/blghtd
root@debian:/home/user/Desktop# xxd /proc/21844/cmdline
00000000: 2f62 696e 2f68 7474 7073 6400 0000 0000  /bin/httpsd.....
00000010: 0000 0000 6578 616d 706c 655f 6172 675f  ....example_arg_
00000020: 7374 7269 6e67 00                        string.
root@debian:/home/user/Desktop# cat /proc/21844/cmdline
/bin/httpsdexample_arg_stringroot@debian:/home/user/Desktop#
```

Abuse of device protection mechanism

FortiOS hides certain files and directories from the device admin, this includes:

`/data/etc/.ftgd_trusted/`

An example of this is shown below:

```
GW # fnsysctl ls -al /data/etc/cert
drwxr-xr-x    7 0        0       Tue May 30 18:11:59 2023            1024 .
drwxr-xr-x   12 0        0       Wed Oct 16 15:06:13 2024            2048 ..
drwxr-xr-x    2 0        0       Sat Jun  3 00:09:32 2023           11264 ca
-rwxrwxr-x    1 0        0       Tue Apr 11 22:32:55 2023             472 cert.conf
lrwxrwxrwx    1 0        0       Tue May 30 18:11:59 2023              12 crl -> /var
/log/crl
drwxr-xr-x    2 0        0       Tue May 30 18:11:23 2023            1024 factory
drwxr-xr-x    2 0        0       Tue May 30 18:11:14 2023            2048 local
drwxr-xr-x    2 0        0       Tue Apr 11 23:08:16 2023            1024 remote

GW # fnsysctl ls -al /data/etc/cert/.ftgd_trusted
drwxr-xr-x    7 0        0       Tue May 30 18:11:59 2023            1024 ..

GW # fnsysctl ls -al /data/etc/cert/.junk
ls: /data/etc/cert/.junk: No such file or directory

GW # fnsysctl ls -al /data/etc/
drwxr-xr-x   12 0        0       Wed Oct 16 15:06:13 2024            2048 .
drwxr-xr-x    8 0        0       Tue May 30 18:11:13 2023            1024 ..
drwx------    2 0        0       Tue May 30 18:11:14 2023            1024 .hidden
-rw-r--r--    1 0        0       Wed Jul 26 00:09:45 2023           58850 app.json.gz
-rw-r--r--    1 0        0       Wed Jul 26 00:09:25 2023          286767 application
```

Figure 2: shows the `.ftgd_trusted` directory not appearing in a directory listing on the device even though other hidden directories can be seen.

`cisz` modifies `/bin/sysctl` replacing the string `/data/etc/.ftgd_trusted/` with the actor's own directory `/data2/.ztls/`.

Note: *It is believed that this is an abuse of the legitimate Fortinet binary `/bin/sysctl` to hide the actor's directory.*

Communications

Command and control

UMBRELLA STAND beacons to its C2 server using communications prefixed with a fake TLS header before the encrypted message. Details of the message format can be found in the Functionality (Message structure) section.

The IP `89.44.194.32` is the hardcoded default C2 present in this sample.

It should be noted that despite using the TLS data bytes `17 03 01` which would be seen in legitimate TLS 1.0 traffic. UMBRELLA STAND and its C2 do not perform a TLS handshake. This stands out from legitimate TLS sessions, as they will begin with a handshake before sending encrypted application data.

Observing a server that responds with TLS data without performing a handshake may be indicative of suspicious activity.

The message data sent by UMBRELLA STAND is encrypted using AES in CBC mode using:

- Key:
  `2C 28 28 40 3D 2B 75 74 69 76 70 34 35 7A 74 76 6F 36 26 52 46 76 74 26 2A 5E 74 70 26 21 71 3B`
- IV:
  `67 C6 69 73 51 FF 4A EC 29 CD BA AB F2 FB E3 46`

It is believed that these values are changed on a per server basis.

Information about the underlying message format can be found in the Functionality (Message Structure) section.

An example of the encrypted traffic is shown below:

Fake TLS traffic		
`17 03 01 01 40 67 C6 69 73 51 FF 4A EC 29 CD BA AB F2 FB E3 46 3B E9…`		
Fake TLS header	Length	Random value
Encrypted message		

The 16 random bytes of random data is generated one byte at a time using `rand`. In the initial beacon this value will be the IV, it is possible that the actor intended to change the IV per message but in this implementation the initial message IV is shared with all future messages.

Conclusion

UMBRELLA STAND is a medium sophistication malware, displaying some operational security considerations and targeting embedded device environments.

It should be noted that there is considerable similarity between the loading components observed in UMBRELLA STAND and COATHANGER. Operational security has clearly been a consideration, with the more recent addition of string encryption, and filenames having been changed.

UMBRELLA STAND is a convoluted set of tooling. Not all of its components have been recovered. There also remains the possibility that further functionality was hidden as described in the COATHANGER reporting.

Detection

Indicators of compromise

Type	Description	Values
IPv4	C2 infrastructure	`89.44.194.32`
Path	Hidden directory actor tooling uses	`/data2/.ztls/`
Path	Paths used by the actors	`/tmp/%d.sv` `/data2/tmp/%s.ini`

Rules and signatures

Description	Identifies sample based on encrypted stack strings.
Precision	No false positives identified in VirusTotal retro-hunts.
Rule type	YARA

```
rule UMBRELLA_STAND_stack_constants_used_for_crypt
{
    meta:
        author = "NCSC"
        description = "Identifies UMBRELLA STAND sample based on
encrypted stack strings."
        date = "2025-06-18"
    strings:
        $ = {C6 45 D0 92 C6 45 D1 48 C6 45 D2 D6 C6 45 D3 E3 C6 45
D4 39 C6 45 D5 EE C6 45 D6 CE C6 45 D7 E4 C6 45 D8 59 C6 45 D9 58
C6 45 DA 97 C6 45 DB E5 C6 45 DC 2E C6 45 DD 27 C6 45 DE 98 C6 45
DF DE C6 45 E0 93 C6 45 E1 A8 C6 45 E2 77 C6 45 E3 9B C6 45 E4 4A
C6 45 E5 F0 C6 45 E6 EF C6 45 E7 74 C6 45 E8 80 C6 45 E9 2D C6 45
EA EB C6 45 EB 30 C6 45 EC 1F C6 45 ED B5 C6 45 EE D9 C6 45 EF C7}
// Key
        $ = {C6 45 C0 77 C6 45 C1 90 C6 45 C2 3D C6 45 C3 35 C6 45
C4 A4 C6 45 C5 9F C6 45 C6 0F C6 45 C7 5E C6 45 C8 F5 C6 45 C9 52
C6 45 CA 44 C6 45 CB 69 C6 45 CC B9 C6 45 CD C0 C6 45 CE BA C6 45
CF DC} // IV
    condition:
        uint32(0) == 0x464C457F and
        any of them
}
```

Description	Identifies sample based on strings, if string obfuscation is not present.
Precision	No false positives identified in VirusTotal retro-hunts.
Rule type	YARA

```
rule UMBRELLA_STAND_plaintext_of_encrypted_stings
{
    meta:
        author = "NCSC"
        description = "Identifies UMBRELLA STAND sample based on
strings, if string obfuscation is not present."
        date = "2025-06-18"
    strings:
        $ = {ED B8 83 20}
        $ = "%s/%d.sv"
        $ = "%s/%s.ini"
        $ = "%s>%s 2>&1"
        $ = "%[^=]=%[^=]"
        $ = "/data2/.ztls/"
        $ = "/proc/%s/status"
        $ = "%*s %s"
        $ = "./lidwok killall blghtd"
        $ = "89.44.194.32"
        $ = "/data2/.ztls/lidwok"
    condition:
        uint32(0) == 0x464C457F and
        5 of them
}
```

Description	Identifies sample based on strings present in injected tooling.
Precision	No false positives identified in VirusTotal retro-hunts.
Rule type	YARA

```
rule UMBRELLA_STAND_injected_tool_load_mechanism
{
    meta:
        author = "NCSC"
        description = "Identifies UMBRELLA STAND sample based on
strings present in tooling."
        date = "2025-06-18"
    strings:
        $ = "LoadMsg"
        $ = "/bin/httpsd"
        $ = "/proc/%d/cmdline"
        $ = "/tmp/logx"
        $ = "error"
    condition:
        uint32(0) == 0x464C457F and
        all of them
}
```

Description	Identifies sample based on custom Base64 alphabet and encryption key.
Precision	No false positives identified in VirusTotal retro-hunts.
Rule type	YARA

```
rule UMBRELLA_STAND_deobfuscated_custom_base64_and_encryption_key
{
    meta:
        author = "NCSC"
        description = "Identifies UMBRELLA STAND sample based on
custom base64 alphabet and encryption key reference."
        date = "2025-06-18"
    strings:

        $ = ",((@=+utivp45ztvo6&RFvt&*^tp&!q;"
        $ =
"ef345FGHIJKLMNOPWghijklmnoXYd67TUVABC89+/pqrstQRSDEZabcuvwxyz012"

    condition:
        uint32(0) == 0x464C457F and
        any of them
}
```

Description	Identifies sample based on string usage
Precision	No false positives identified in VirusTotal retro-hunts.
Rule type	YARA

```
rule UMBRELLA_STAND_strings_used_in_firewall_malware
{
    meta:
        author = "NCSC"
        description = "Identifies UMBRELLA STAND sample based on
strings."
        date = "2025-06-18"
    strings:
        $ = "/bin/httpsd"
        $ = "dxumbsu"
        $ = "/proc/%d/cmdline"
        $ = "/tmp/logx"
        $ = "/etc/ld.so.preload"
        $ = "loadMsg"
        $ ="/SYSV64564856"
    condition:
        uint32(0) == 0x464C457F and
        all of them
}
```

Description	Identifies sample based on strings used to configure injection.
Precision	No false positives identified in VirusTotal retro-hunts.
Rule type	YARA

```
rule UMBRELLA_STAND_injection_related_strings
{
    meta:
        author = "NCSC"
        description = "Identifies UMBRELLA STAND sample based on
strings used to configure injection."
        date = "2025-06-18"
    strings:
        $ = "/SYSV64564856"
        $ = "/etc/ld.so.preload"
        $ = "version 2"
        $ = "init_drivercheck"
    condition:
        uint32(0) == 0x464C457F and
        3 of them
}
```

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	T1059.004	Command and Scripting Interpreter: Unix Shell	UMBRELLA STAND contains functionality to receive and run Shell commands.
Defense Evasion	T1140	Deobfuscate/Decode Files or Information	UMBRELLA STAND contains AES-encrypted strings.
	T155	Process Injection	UMBRELLA STAND uses tooling to perform process injection.
	T1070.004	Indicator Removal: File Deletion	UMBRELLA STAND binaries had been deleted from the device.
	T1564.010	Hide Artifacts: Process Argument Spoofing	UMBRELLA STAND modifies it's process name
	T1564.001	Hide Artifacts: Hidden Files and Directories	UMBRELLA STAND and associated tooling use hidden directories.
Persistence	T1547.006	Boot or Logon Autostart Execution: Kernel Modules and Extensions	UMBRELLA STAND has functionality to modify `reboot` to achieve persistence.
	T1574.006	Hijack Execution Flow: Dynamic Linker Hijacking	UMBRELLA STAND has functionality to load via `ldpreload`.
Discovery	T1057	Process Discovery	UMBRELLA STAND performs process discovery to check that its main process is still executing.
	T1046	Network Service Discovery	UMBRELLA STAND was in use with `nbtscan` which can list NetBIOS computer names.
	T1040	Network Sniffing	UMBRELLA STAND was in use with `tcpdump` which can be,

			and was, used to perform packet captures.
Command and Control	T1001.003	Data Obfuscation: Protocol or Service Impersonation	UMBRELLA STAND uses fake TLS for communications with the C2 server.

Appendix

BusyBox functionality

The following commands are defined in this version of `BusyBox`.

```
Currently defined functions:
	[, [[, acpid, add-shell, addgroup, adduser, adjtimex,
arch, arp, arping, ash, awk, base64,
	basename, bc, beep, blkdiscard, blkid, blockdev,
bootchartd, brctl, bunzip2, bzcat, bzip2,
	cal, cat, chat, chattr, chgrp, chmod, chown, chpasswd,
chpst, chroot, chrt, chvt, cksum,
	clear, cmp, comm, conspy, cp, cpio, crond, crontab,
cryptpw, cttyhack, cut, date, dc, dd,
	deallocvt, delgroup, deluser, depmod, devmem, df,
dhcprelay, diff, dirname, dmesg, dnsd,
	dnsdomainname, dos2unix, dpkg, dpkg-deb, du, dumpkmap,
dumpleases, echo, ed, egrep, eject,
	env, envdir, envuidgid, ether-wake, expand, expr, factor,
fakeidentd, fallocate, false,
	fatattr, fbset, fbsplash, fdflush, fdformat, fdisk,
fgconsole, fgrep, find, findfs, flock,
	fold, free, freeramdisk, fsck, fsck.minix, fsfreeze,
fstrim, fsync, ftpd, ftpget, ftpput,
	fuser, getopt, getty, grep, groups, gunzip, gzip, halt,
hd, hdparm, head, hexdump, hexedit,
	hostid, hostname, httpd, hush, hwclock, i2cdetect,
i2cdump, i2cget, i2cset, i2ctransfer, id,
	ifconfig, ifdown, ifenslave, ifplugd, ifup, inetd, init,
insmod, install, ionice, iostat, ip,
	ipaddr, ipcalc, ipcrm, ipcs, iplink, ipneigh, iproute,
iprule, iptunnel, kbd_mode, kill,
	killall, killall5, klogd, last, less, link, linux32,
linux64, linuxrc, ln, loadfont,
	loadkmap, logger, login, logname, logread, losetup, lpd,
lpq, lpr, ls, lsattr, lsmod, lsof,
	lspci, lsscsi, lsusb, lzcat, lzma, lzop, makedevs,
makemime, man, md5sum, mdev, mesg,
	microcom, mkdir, mkdosfs, mke2fs, mkfifo, mkfs.ext2,
mkfs.minix, mkfs.vfat, mknod, mkpasswd,
	mkswap, mktemp, modinfo, modprobe, more, mount,
mountpoint, mpstat, mt, mv, nameif, nanddump,
	nandwrite, nbd-client, nc, netstat, nice, nl, nmeter,
nohup, nologin, nproc, nsenter,
	nslookup, ntpd, nuke, od, openvt, partprobe, passwd,
paste, patch, pgrep, pidof, ping, ping6,
	pipe_progress, pivot_root, pkill, pmap, popmaildir,
poweroff, powertop, printenv, printf, ps,
	pscan, pstree, pwd, pwdx, raidautorun, rdate, rdev,
```

readahead, readlink, readprofile,
realpath, reboot, reformime, remove-shell, renice, reset, resize, resume, rev, rm, rmdir,
rmmod, route, rpm, rpm2cpio, rtcwake, run-init, run-parts, runlevel, runsv, runsvdir, rx,
script, scriptreplay, sed, sendmail, seq, setarch, setconsole, setfattr, setfont,
setkeycodes, setlogcons, setpriv, setserial, setsid, setuidgid, sh, sha1sum, sha256sum,
sha3sum, sha512sum, showkey, shred, shuf, slattach, sleep, smemcap, softlimit, sort, split,
ssl_client, start-stop-daemon, stat, strings, stty, su, sulogin, sum, sv, svc, svlogd, svok,
swapoff, swapon, switch_root, sync, sysctl, syslogd, tac, tail, tar, taskset, tc, tcpsvd,
tee, telnet, telnetd, test, tftp, tftpd, time, timeout, top, touch, tr, traceroute,
traceroute6, true, truncate, ts, tty, ttysize, tunctl, ubiattach, ubidetach, ubimkvol,
ubirename, ubirmvol, ubirsvol, ubiupdatevol, udhcpc, udhcpc6, udhcpd, udpsvd, uevent, umount,
uname, unexpand, uniq, unix2dos, unlink, unlzma, unshare, unxz, unzip, uptime, users, usleep,
uudecode, uuencode, vconfig, vi, vlock, volname, w, wall, watch, watchdog, wc, wget, which,
who, whoami, whois, xargs, xxd, xz, xzcat, yes, zcat, zcip

Disclaimer

This report draws on information derived from the 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.