Who is calling? CDRThief targets Linux VoIP softswitches

By Anton Cherepanov

Archived: 2026-04-05 17:01:18 UTC

ESET Research

ESET researchers have discovered and analyzed malware that targets Voice over IP (VoIP) softswitches

10 Sep 2020 • , 6 min. read

This new malware that we have discovered and named CDRThief is designed to target a very specific VoIP platform, used by two China-produced softswitches (software switches): Linknat VOS2009 and VOS3000. A softswitch is a core element of a VoIP network that provides call control, billing, and management. These softswitches are software-based solutions that run on standard Linux servers.

The primary goal of the malware is to exfiltrate various private data from a compromised softswitch, including call detail records (CDR). CDRs contain metadata about VoIP calls such as caller and callee IP addresses, starting time of the call, call duration, calling fee, etc.

To steal this metadata, the malware queries internal MySQL databases used by the softswitch. Thus, attackers demonstrate a good understanding of the internal architecture of the targeted platform.

Linux/CDRThief analysis

We noticed this malware in one of our sample sharing feeds, and as entirely new Linux malware is a rarity, it caught our attention. What was even more interesting was that it quickly became apparent that this malware targeted a specific Linux VoIP platform. Its ELF binary was produced by the Go compiler with the debug symbols left unmodified, which is always helpful for the analysis.

To hide malicious functionality from basic static analysis, the authors encrypted all suspicious-looking strings with XXTEA and the key fhu84ygf8643, and then base64 encoded them. Figure 1 shows some of the code the malware uses to decrypt these strings at runtime.

```
.text:0000000000678080 ; =============== S U B R O U T I N E =======================================
.text:0000000000678080
.text:0000000000678080
.text:0000000000678080 main_strDec proc near           ; CODE XREF: main_resolver_vosVersion+35↑p
.text:0000000000678080                                 ; main_getVosMysqlInfo+46↑p ...
.text:0000000000678080
.text:0000000000678080 var_48  = qword ptr -48h
.text:0000000000678080 var_40  = qword ptr -40h
.text:0000000000678080 var_38  = qword ptr -38h
.text:0000000000678080 var_30  = qword ptr -30h
.text:0000000000678080 var_28  = qword ptr -28h
.text:0000000000678080 var_20  = qword ptr -20h
.text:0000000000678080 var_8   = qword ptr -8
.text:0000000000678080 arg_0   = qword ptr  8
.text:0000000000678080 arg_8   = qword ptr  10h
.text:0000000000678080 arg_10  = qword ptr  18h
.text:0000000000678080 arg_18  = qword ptr  20h
.text:0000000000678080
.text:0000000000678080         mov     rcx, fs:0FFFFFFFFFFFFFFF8h
.text:0000000000678089         cmp     rsp, [rcx+10h]
.text:000000000067808D         jbe     short loc_6780E8
.text:000000000067808F         sub     rsp, 48h
.text:0000000000678093         mov     [rsp+48h+var_8], rbp
.text:0000000000678098         lea     rbp, [rsp+48h+var_8]
.text:000000000067809D         mov     rax, [rsp+48h+arg_0]
.text:00000000006780A2         mov     [rsp+48h+var_48], rax
.text:00000000006780A6         mov     rax, [rsp+48h+arg_8]
.text:00000000006780AB         mov     [rsp+48h+var_40], rax
.text:00000000006780B0         lea     rax, unk_721E60 ; fhu84ygf8643
.text:00000000006780B7         mov     [rsp+48h+var_38], rax
.text:00000000006780BC         mov     [rsp+48h+var_30], 0Ch
.text:00000000006780C5         call    github_com_xxtea_xxtea_go_xxtea_DecryptString
.text:00000000006780CA         mov     rax, [rsp+48h+var_28]
.text:00000000006780CF         mov     rcx, [rsp+48h+var_20]
.text:00000000006780D4         mov     [rsp+48h+arg_10], rax
.text:00000000006780D9         mov     [rsp+48h+arg_18], rcx
.text:00000000006780DE         mov     rbp, [rsp+48h+var_8]
.text:00000000006780E3         add     rsp, 48h
.text:00000000006780E7         retn
.text:00000000006780E8 ; ---------------------------------------------------------------------------
.text:00000000006780E8
.text:00000000006780E8 loc_6780E8:                     ; CODE XREF: main_strDec+D↑j
.text:00000000006780E8         call    runtime_morestack_noctxt
.text:00000000006780ED         jmp     short main_strDec
.text:00000000006780ED main_strDec endp
```

Figure 1. The routine used to decrypt the binary's strings

To access internal data stored in the MySQL database, the malware reads credentials from Linknat VOS2009 and VOS3000 configuration files that it attempts to locate in the following paths:

- /usr/kunshi/vos2009/server/etc/server_db_config.xml
- /usr/kunshi/vos3000/server/etc/server_db_config.xml
- /home/kunshi/vos2009/server/etc/server_db_config.xml
- /home/kunshi/vos3000/server/etc/server_db_config.xml
- /home/kunshi/vos2009/etc/server_db_config.xml
- /home/kunshi/vos3000/etc/server_db_config.xml
- /usr/kunshi/vos2009/server/etc/serverdbconfig.xml
- /usr/kunshi/vos3000/server/etc/serverdbconfig.xml

Interestingly, the password from the configuration file is stored encrypted. However, Linux/CDRThief malware is still able to read and decrypt it. Thus, the attackers demonstrate deep knowledge of the targeted platform, since the algorithm and encryption keys used are not documented as far as we can tell. It means that the attackers had to reverse engineer platform binaries or otherwise obtain information about the AES encryption algorithm and key used in the Linknat code.

As seen in Figure 2, CDRThief communicates with C&C servers using JSON over HTTP.

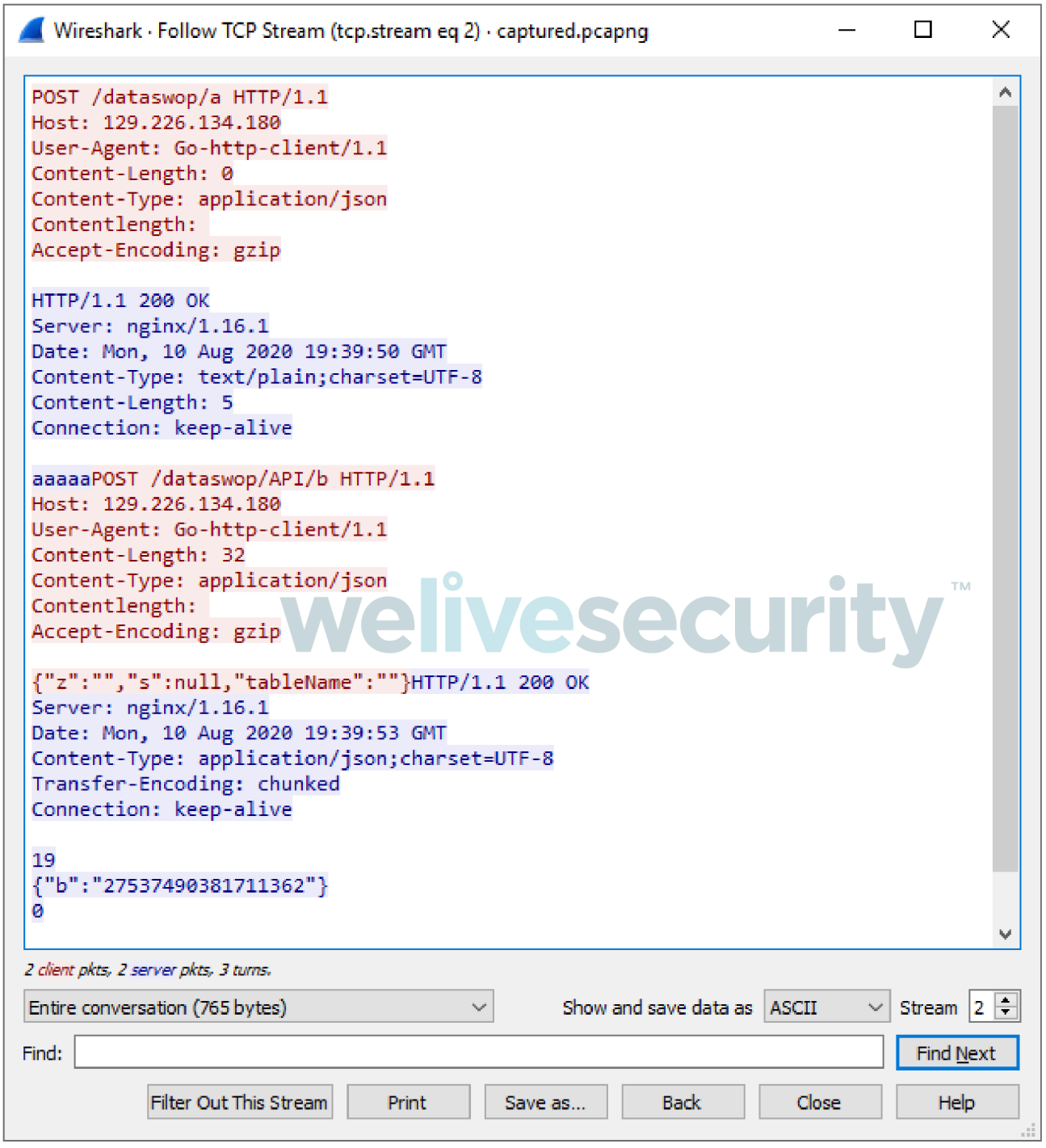

Figure 2. Captured network communication of the Linux/CDRThief malware

There are multiple functions in Linux/CDRThief's code used for communication with C&C servers. Table 1 contains the original names of these functions used by the malware authors.

Table 1. Functions used for communication with C&C

Function name	C&C path	Purpose
main.pingNet	/dataswop/a	Checks if C&C is alive

Function name	C&C path	Purpose
main.getToken	/dataswop/API/b	Obtains token
main.heartbeat	/dataswop/API/gojvxs	Main C&C loop, called every three minutes
main.baseInfo	/dataswop/API/gojvxs	Exfiltrates basic information about compromised Linknat system:
	#rowspan#	• MAC address
	#rowspan#	• cat /proc/version
	#rowspan#	• whoami
	#rowspan#	• cat /etc/redhat-release
	#rowspan#	• UUID from /bin/ibus_10.mo (or /home/kunshi/base/ibus_10.mo)
main.upVersion	/dataswop/Download/updateGoGoGoGoGo	Updates itself to the latest version
main.pushLog	/dataswop/API/gojvxs	Uploads malware error log
main.load	/dataswop/API/gojvxs	Exfiltrates various information about the platform:
	#rowspan#	• SELECT SUM(TABLE_ROWS) FROM information_schema.TABLES WHERE table_name LIKE 'e_cdr_%'
	#rowspan#	• cat /etc/motd

Function name	C&C path	Purpose
	#rowspan#	• username, encrypted password, IP address of the database
	#rowspan#	• ACCESS_UUID from server.conf
	#rowspan#	• VOS software version
main.syslogCall	/dataswop/API/gojvxs	Exfiltrates data from e_syslog tables
main.gatewaymapping	/dataswop/API/gojvxs	Exfiltrates data from e_gatewaymapping tables
main.cdr	/dataswop/API/gojvxs	Exfiltrates data from e_cdr tables

In order to exfiltrate data from the platform, Linux/CDRThief executes SQL queries directly to the MySQL database. Mainly, the malware is interested in three tables:

- e_syslog – contains log of system events
- e_gatewaymapping – contains information about VoIP gateways (see Figure 3)
- e_cdr – contains call data records (metadata of calls)

```
.text:0000000000679A10 main_s_gatewaymapping proc near
.text:0000000000679A10                                 ; CODE XREF: main_s_gatewaymapping+154↓j
.text:0000000000679A10                                 ; main_gatewaymapping+32↓p
.text:0000000000679A10                                 ; DATA XREF: ...
.text:0000000000679A10
.text:0000000000679A10 var_70  = qword ptr -70h
.text:0000000000679A10 var_68  = qword ptr -68h
.text:0000000000679A10 var_60  = qword ptr -60h
.text:0000000000679A10 var_58  = qword ptr -58h
.text:0000000000679A10 var_50  = qword ptr -50h
.text:0000000000679A10 var_48  = qword ptr -48h
.text:0000000000679A10 var_40  = qword ptr -40h
.text:0000000000679A10 var_38  = qword ptr -38h
.text:0000000000679A10 var_30  = qword ptr -30h
.text:0000000000679A10 var_28  = qword ptr -28h
.text:0000000000679A10 var_20  = qword ptr -20h
.text:0000000000679A10 var_18  = qword ptr -18h
.text:0000000000679A10 var_10  = qword ptr -10h
.text:0000000000679A10 var_8   = qword ptr -8
.text:0000000000679A10 arg_0   = qword ptr  8
.text:0000000000679A10 arg_8   = qword ptr  10h
.text:0000000000679A10 arg_10  = qword ptr  18h
.text:0000000000679A10 arg_18  = qword ptr  20h
.text:0000000000679A10 arg_20  = qword ptr  28h
.text:0000000000679A10 arg_28  = qword ptr  30h
.text:0000000000679A10 arg_30  = qword ptr  38h
.text:0000000000679A10 arg_38  = qword ptr  40h
.text:0000000000679A10
.text:0000000000679A10         mov     rcx, fs:0FFFFFFFFFFFFFF8h
.text:0000000000679A19         cmp     rsp, [rcx+10h]
.text:0000000000679A1D         jbe     loc_679B5F
.text:0000000000679A23         sub     rsp, 70h
.text:0000000000679A27         mov     [rsp+70h+var_8], rbp
.text:0000000000679A2C         lea     rbp, [rsp+70h+var_8]
.text:0000000000679A31         lea     rax, aIyaftlzb2bsr0n ; "iYAfTlzb2bSR0N/0SwC8IMldwVaX1yYfDr6oaxl"
.text:0000000000679A38         mov     [rsp+70h+var_70], rax
.text:0000000000679A3C         mov     [rsp+70h+var_68], 40h ; '@'
.text:0000000000679A45         call    main_strDec   ; SELECT * FROM `e_gatewaymapping` WHERE id>'
.text:0000000000679A4A         mov     rax, [rsp+70h+var_60]
.text:0000000000679A4F         mov     [rsp+70h+var_10], rax
.text:0000000000679A54         mov     rcx, [rsp+70h+var_58]
.text:0000000000679A59         mov     [rsp+70h+var_18], rcx
.text:0000000000679A5E         lea     rdx, aA9kd2gibrphayl ; "A9KD2gibRphaYlT6dRswyFh7NtIzsxnc"
.text:0000000000679A65         mov     [rsp+70h+var_70], rdx
.text:0000000000679A69         mov     [rsp+70h+var_68], 20h ; ' '
.text:0000000000679A72         call    main_strDec   ; ' ORDER BY id ASC ;
.text:0000000000679A77         mov     rax, [rsp+70h+var_60]
```


Figure 3. Disassembled code of the function that initializes an SQL query

Data to be exfiltrated from the e_syslog, e_gatewaymapping, and e_cdr tables is compressed and then encrypted with a hardcoded RSA-1024 public key before exfiltration. Thus, only the malware authors or operators can decrypt the exfiltrated data.

Based on the described functionality, we can say that the malware's primary focus is on collecting data from the database. Unlike other backdoors, Linux/CDRThief does not have support for shell command execution or exfiltrating specific files from the compromised softswitch's disk. However, these functions could be introduced in an updated version.

The malware can be deployed to any location on the disk under any file name. It's unknown what type of persistence is used for starting the malicious binary at each boot. However, it should be noted that once the

malware is started, it attempts to launch a legitimate binary present on the Linknat VOS2009/VOS3000 platform using the following command:

exec -a '/home/kunshi/callservice/bin/callservice -r /home/kunshi/.run/callservice.pid'

This suggests that the malicious binary might somehow be inserted into a regular boot chain of the platform in order to achieve persistence and possibly masquerading as a component of the Linknat softswitch software.

At the time of writing we do not know how the malware is deployed onto compromised devices. We speculate that attackers might obtain access to the device using a brute-force attack or by exploiting a vulnerability. Such vulnerabilities in VOS2009/VOS3000 have been reported publicly in the past.

Conclusion

We analyzed Linux/CDRThief malware, which has a unique purpose to target specific VoIP softswitches. We rarely see VoIP softswitches targeted by threat actors; this makes the Linux/CDRThief malware interesting.

It's hard to know the ultimate goal of attackers who use this malware. However, since this malware exfiltrates sensitive information, including call metadata, it seems reasonable to assume that the malware is used for cyberespionage. Another possible goal for attackers using this malware is VoIP fraud. Since the attackers obtain information about activity of VoIP softswitches and their gateways, this information could be used to perform International Revenue Share Fraud (IRSF).

For any inquiries, or to make sample submissions related to the subject, contact us at threatintel@eset.com.

Indicators of Compromise

ESET detection name

Linux/CDRThief.A

File based mutexes

/dev/shm/.bin
/dev/shm/.linux

Files created during malware update

/dev/shm/callservice
/dev/shm/sys.png

Hashes

CC373D633A16817F7D21372C56955923C9DDA825
8E2624DA4D209ABD3364D90F7BC08230F84510DB (UPX packed)
FC7CCABB239AD6FD22472E5B7BB6A5773B7A3DAC

8532E858EB24AE38632091D2D790A1299B7BBC87 (Corrupted)
82F51F098B85995C966135E9E7F63D1D8DC97589 (UPX packed)

C&C

http://119.29.173[.]65
http://129.211.157[.]244
http://129.226.134[.]180
http://150.109.79[.]136
http://34.94.199[.]142
http://35.236.173[.]187
http://update[.]callercore[.]com

Exfiltration encryption key (RSA)

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCQ3k3GgS3FX4pI7s9x0krBYqbMcSaw4BPY91Ln
tt5/X8s9l0BC6PUTbQcUzs6PPXhKKTx8ph5CYQqdWynxOLJah0FMMRYxS8d0HX+Qx9eWUeKRHm2E
AtZQjdHxqTJ9EBpHYWV4RrWmeoOsWAOisvedlb23O0E55e8rrGGrZLhPbwIDAQAB
-----END PUBLIC KEY-----

MITRE ATT&CK techniques

Note: This table was built using version 7 of the MITRE ATT&CK framework.

Tactic	ID	Name	Description
Defense Evasion	T1027	Obfuscated Files or Information	Linux/CDRThief contains obfuscates strings in the payload.
	T1027.002	Obfuscated Files or Information: Software Packing	Some Linux/CDRThief samples are packed with UPX.
Credential Access	T1552.001	Unsecured Credentials: Credentials In Files	Linux/CDRThief reads credentials for MySQL database from a configuration file.
Discovery	T1082	System Information Discovery	Linux/CDRThief obtains detailed information about the compromised computer.
Collection	T1560.003	Archive Collected Data: Archive via Custom Method	Linux/CDRThief compresses stolen data with gzip before exfiltration.
Command and Control	T1071.001	Application Layer Protocol: Web Protocols	Linux/CDRThief uses HTTP for communication with C&C server.

Tactic	ID	Name	Description
Exfiltration	T1041	Exfiltration Over C2 Channel	Linux/CDRThief exfiltrates data to the C&C server.

Source: https://www.welivesecurity.com/2020/09/10/who-callin-cdrthief-linux-voip-softswitches/