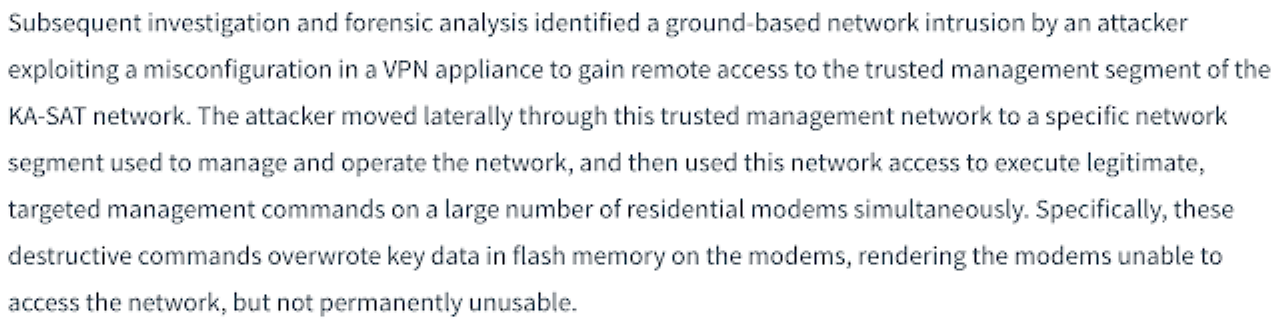
VIASAT incident: from speculation to technical details.

reversemode.com/2022/03/viasat-incident-from-speculation-to.html

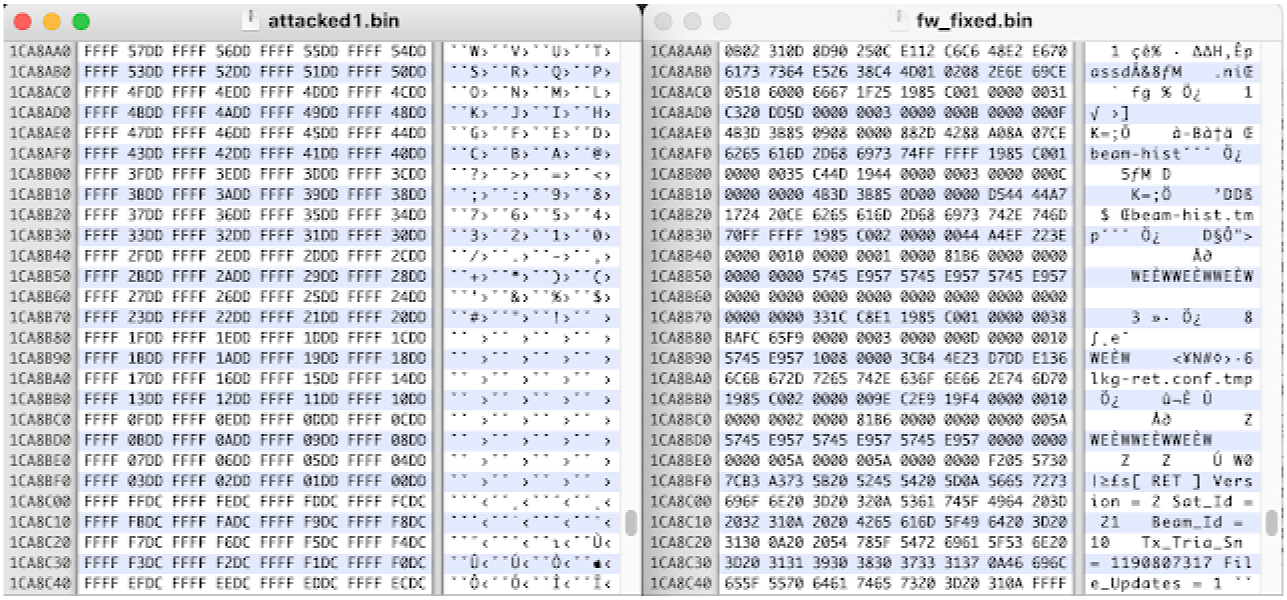
34 days after the incident, yesterday Viasat published a statement providing some technical details about the attack that affected tens of thousands of its SATCOM terminals. Also yesterday, I eventually had access to two Surfbeam2 modems: one was targeted during the attack and the other was in a working condition. Thank you so much to the person who disinterestedly donated the attacked modem.

I've been closely covering this issue since the beginning, providing a plausible theory based on the information that was available at that time, and my experience in this field. Actually, it seems that this theory was pretty close to what really happened.



Subsequent investigation and forensic analysis identified a ground-based network intrusion by an attacker exploiting a misconfiguration in a VPN appliance to gain remote access to the trusted management segment of the KA-SAT network. The attacker moved laterally through this trusted management network to a specific network segment used to manage and operate the network, and then used this network access to execute legitimate, targeted management commands on a large number of residential modems simultaneously. Specifically, these destructive commands overwrote key data in flash memory on the modems, rendering the modems unable to access the network, but not permanently unusable.

Fortunately, now we can move from just pure speculation into something more tangible, so I dumped the flash memory for both modems (Spansion S29GL256P90TFCR2) and the differences were pretty clear. In the following picture you can see 'attacked1.bin', which belongs to the targeted modem and 'fw_fixed.bin', coming from the modem in working conditions.



```
1CA8AA0  FFFF 57DD FFFF 56DD FFFF 55DD FFFF 54DD  ˇˇW›ˇˇV›ˇˇU›ˇˇT›
1CA8AB0  FFFF 53DD FFFF 52DD FFFF 51DD FFFF 50DD  ˇˇS›ˇˇR›ˇˇQ›ˇˇP›
1CA8AC0  FFFF 4FDD FFFF 4EDD FFFF 4DDD FFFF 4CDD  ˇˇO›ˇˇN›ˇˇM›ˇˇL›
1CA8AD0  FFFF 4BDD FFFF 4ADD FFFF 49DD FFFF 48DD  ˇˇK›ˇˇJ›ˇˇI›ˇˇH›
1CA8AE0  FFFF 47DD FFFF 46DD FFFF 45DD FFFF 44DD  ˇˇG›ˇˇF›ˇˇE›ˇˇD›
1CA8AF0  FFFF 43DD FFFF 42DD FFFF 41DD FFFF 40DD  ˇˇC›ˇˇB›ˇˇA›ˇˇ@›
1CA8B00  FFFF 3FDD FFFF 3EDD FFFF 3DDD FFFF 3CDD  ˇˇ?›ˇˇ>›ˇˇ=›ˇˇ<›
1CA8B10  FFFF 3BDD FFFF 3ADD FFFF 39DD FFFF 38DD  ˇˇ;›ˇˇ:›ˇˇ9›ˇˇ8›
1CA8B20  FFFF 37DD FFFF 36DD FFFF 35DD FFFF 34DD  ˇˇ7›ˇˇ6›ˇˇ5›ˇˇ4›
1CA8B30  FFFF 33DD FFFF 32DD FFFF 31DD FFFF 30DD  ˇˇ3›ˇˇ2›ˇˇ1›ˇˇ0›
1CA8B40  FFFF 2FDD FFFF 2EDD FFFF 2DDD FFFF 2CDD  ˇˇ/›ˇˇ.›ˇˇ-›ˇˇ,›
1CA8B50  FFFF 2BDD FFFF 2ADD FFFF 29DD FFFF 28DD  ˇˇ+›ˇˇ*›ˇˇ)›ˇˇ(›
1CA8B60  FFFF 27DD FFFF 26DD FFFF 25DD FFFF 24DD  ˇˇ'›ˇˇ&›ˇˇ%›ˇˇ$›
1CA8B70  FFFF 23DD FFFF 22DD FFFF 21DD FFFF 20DD  ˇˇ#›ˇˇ"›ˇˇ!›ˇˇ ›
1CA8B80  FFFF 1FDD FFFF 1EDD FFFF 1DDD FFFF 1CDD  ˇˇ ›ˇˇ ›ˇˇ ›ˇˇ ›
1CA8B90  FFFF 1BDD FFFF 1ADD FFFF 19DD FFFF 18DD  ˇˇ ›ˇˇ ›ˇˇ ›ˇˇ ›
1CA8BA0  FFFF 17DD FFFF 16DD FFFF 15DD FFFF 14DD  ˇˇ ›ˇˇ ›ˇˇ ›ˇˇ ›
1CA8BB0  FFFF 13DD FFFF 12DD FFFF 11DD FFFF 10DD  ˇˇ ›ˇˇ ›ˇˇ ›ˇˇ ›
1CA8BC0  FFFF 0FDD FFFF 0EDD FFFF 0DDD FFFF 0CDD  ˇˇ ›ˇˇ ›ˇˇ ›ˇˇ ›
1CA8BD0  FFFF 0BDD FFFF 0ADD FFFF 09DD FFFF 08DD  ˇˇ ›ˇˇ ›ˇˇ ›ˇˇ ›
1CA8BE0  FFFF 07DD FFFF 06DD FFFF 05DD FFFF 04DD  ˇˇ ›ˇˇ ›ˇˇ ›ˇˇ ›
1CA8BF0  FFFF 03DD FFFF 02DD FFFF 01DD FFFF 00DD  ˇˇ ›ˇˇ ›ˇˇ ›ˇˇ ›
1CA8C00  FFFF FFDC FFFF FEDC FFFF FDDC FFFF FCDC  ˇˇˇ‹ˇˇ˛‹ˇˇ˝‹ˇˇ¸‹
1CA8C10  FFFF FBDC FFFF FADC FFFF F9DC FFFF F8DC  ˇˇ˚‹ˇˇ˙‹ˇˇ˘‹ˇˇ¯‹
1CA8C20  FFFF F7DC FFFF F6DC FFFF F5DC FFFF F4DC  ˇˇ˜‹ˇˇˆ‹ˇˇı‹ˇˇÙ‹
1CA8C30  FFFF F3DC FFFF F2DC FFFF F1DC FFFF F0DC  ˇˇÛ‹ˇˇÚ‹ˇˇÒ‹ˇˇ‹
1CA8C40  FFFF EFDC FFFF EEDC FFFF EDDC FFFF ECDC  ˇˇÔ‹ˇˇÓ‹ˇˇÌ‹ˇˇÏ‹
```

```
1CA8AA0  0802 310D 8D90 258C E112 C6C6 48E2 E670    1 çê% · ∆∆H‚Êp
1CA8AB0  6173 7364 E526 38C4 4D01 0208 2E6E 69CE  assdÅ&8ƒM   .niŒ
1CA8AC0  0510 6000 6667 1F25 1985 C001 0000 0031    ` fg % Ö¿     1
1CA8AD0  C320 DD5D 0000 0003 0000 000B 0000 000F  √ ›]
1CA8AE0  483D 3B85 0908 0000 882D 4288 A08A 07CE  K=;Ö    à-Bà†ä Œ
1CA8AF0  6265 616D 2D68 6973 74FF FFFF 1985 C001  beam-histˇˇˇ Ö¿
1CA8B00  0000 0035 C44D 1944 0000 0003 0000 000C     5ƒM D
1CA8B10  0000 0000 483D 3B85 0D00 0000 D544 44A7      K=;Ö    'DDß
1CA8B20  1724 20CE 6265 616D 2D68 6973 742E 746D   $ Œbeam-hist.tm
1CA8B30  70FF FFFF 1985 C002 0000 0044 A4EF 223E  pˇˇˇ Ö¿    D§Ô">
1CA8B40  0000 0010 0000 0001 0000 81B6 0000 0000            Å∂
1CA8B50  0000 0000 5745 E957 5745 E957 5745 E957      WEÈWWEÈWWEÈW
1CA8B60  0000 0000 0000 0000 0000 0000 0000 0000
1CA8B70  0000 0000 331C C8E1 1985 C001 0000 0038      3 »· Ö¿    8
1CA8B80  BAFC 65F9 0000 0003 0000 000D 0000 0010  ∫.eˇ
1CA8B90  5745 E957 1008 0000 3CB4 4E23 D7DD E136  WEÈW    <¥N#◊›·6
1CA8BA0  6C6B 672D 7265 742E 636F 6E66 2E74 6D70  lkg-ret.conf.tmp
1CA8BB0  1985 C002 0000 009E C2E9 19F4 0000 0010   Ö¿    û¬É Ù
1CA8BC0  0000 0002 0000 81B6 0000 0000 0000 005A        Å∂       Z
1CA8BD0  5745 E957 5745 E957 5745 E957 0000 0000  WEÈWWEÈWWEÈW
1CA8BE0  0000 005A 0000 005A 0000 0000 F205 5730     Z   Z    Ú W0
1CA8BF0  7CB3 A373 5B20 5245 5420 5D0A 5665 7273  |≥£s[ RET ] Vers
1CA8C00  696F 6E20 3D20 320A 5361 745F 4964 203D  ion = 2 Sat_Id =
1CA8C10  2032 310A 2020 4265 616D 5F49 6420 3D20   21   Beam_Id =
1CA8C20  3130 0A20 2054 785F 5472 6961 5F53 6E20  10   Tx_Tria_Sn
1CA8C30  3D20 3131 3930 3830 3733 3137 0A46 696C  = 1190807317 Fil
1CA8C40  655F 5570 6461 7465 7320 3D20 310A FFFF  e_Updates = 1 ˇˇ
```

A destructive pattern, that corrupted the flash memory rendering the SATCOM modems inoperable, can be observed on the left, confirming what Viasat stated yesterday.

After verifying the destructive attack, I'm now statically analyzing the firmware extracted from the 'clean' modem. Firmware version is 3.7.3.10.9, which seems to date back to late 2017.

Besides talking about a 'management network' and 'legitimate management commands', Viasat did not provide any specific details about this. In my previous blog post I introduced the theory that probably 'TR069' was the involved management protocol.

Obviously, I can't completely confirm this scenario but I'll try to elaborate my reasoning.

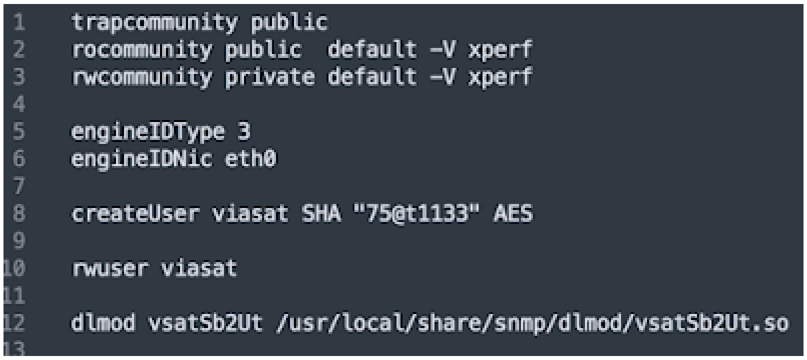
Attacking via a management protocol

I think there are two main options: either the attackers abused a MAC management protocol or an application layer one.

For the MAC case ('ut_mac' binary), in general terms, the attackers would have required an even more privileged access to either the NOC or the Ground Stations, probably in a persistent way via malware. I guess that this kind of privileged access would have been enough to limit the attack to Ukraine, instead of knocking out half Europe. As a result, I'm inclined to think this was not the case.

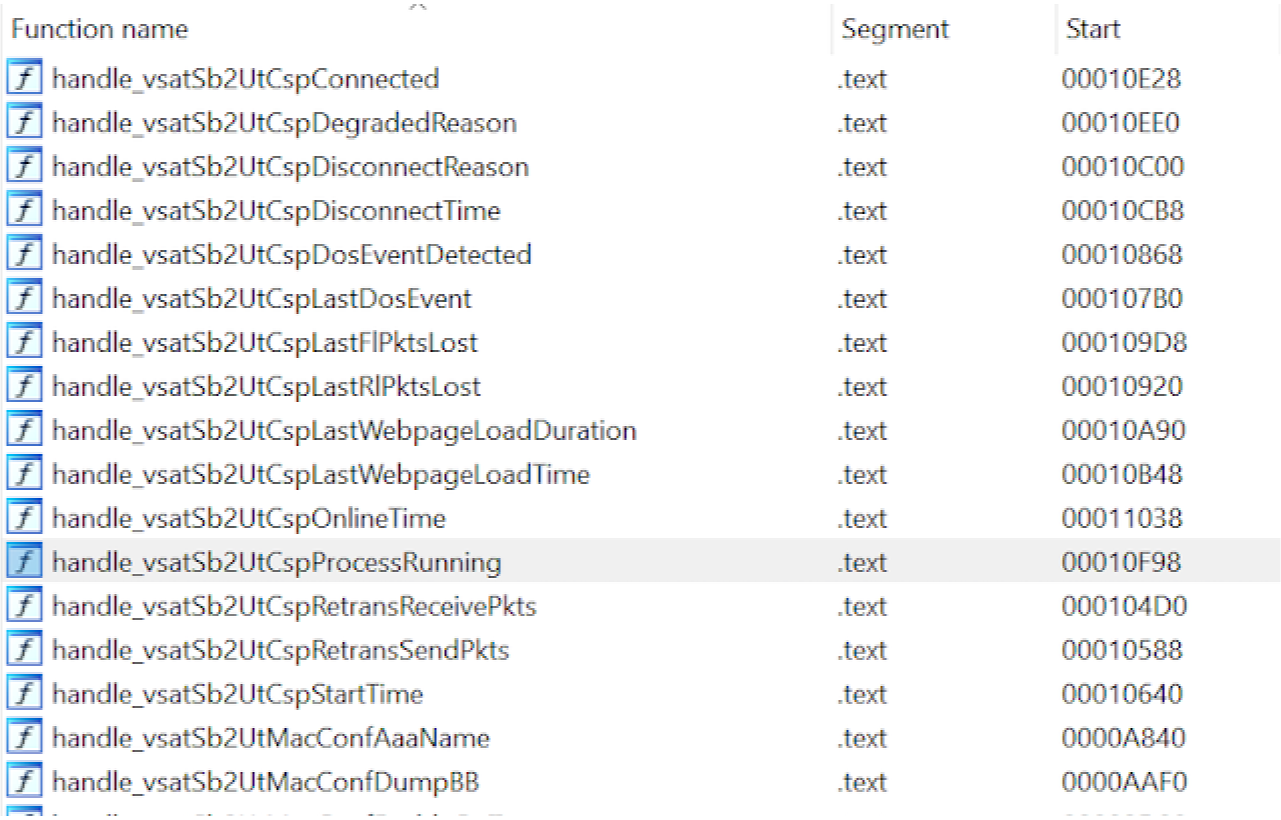
On the other hand, a 'misconfigured VPN' that enabled the attackers to reach the 'management segment' and execute 'commands' seems to be more related to an application layer management protocol: SNMP or TR069.

SNMP



```
trapcommunity public
rocommunity public  default -V xperf
rwcommunity private default -V xperf

engineIDType 3
engineIDNic eth0

createUser viasat SHA "75@t1133" AES

rwuser viasat

dlmod vsatSb2Ut /usr/local/share/snmp/dlmod/vsatSb2Ut.so

```

An initial analysis of 'vsatSb2Ut.so' shows that the implemented MIB does not seem to provide the required functionality to perform this kind of attack.

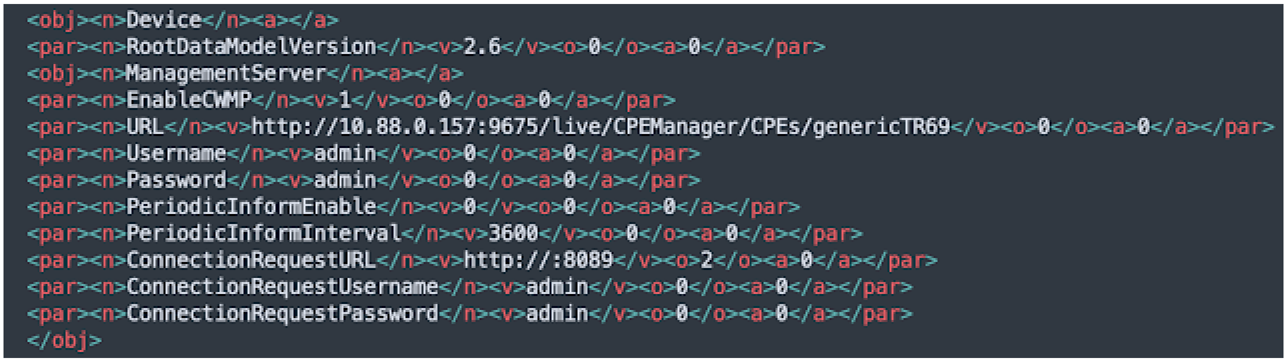


Function name	Segment	Start
handle_vsatSb2UtCspConnected	.text	00010E28
handle_vsatSb2UtCspDegradedReason	.text	00010EE0
handle_vsatSb2UtCspDisconnectReason	.text	00010C00
handle_vsatSb2UtCspDisconnectTime	.text	00010CB8
handle_vsatSb2UtCspDosEventDetected	.text	00010868
handle_vsatSb2UtCspLastDosEvent	.text	000107B0
handle_vsatSb2UtCspLastFlPktsLost	.text	000109D8
handle_vsatSb2UtCspLastRlPktsLost	.text	00010920
handle_vsatSb2UtCspLastWebpageLoadDuration	.text	00010A90
handle_vsatSb2UtCspLastWebpageLoadTime	.text	00010B48
handle_vsatSb2UtCspOnlineTime	.text	00011038
handle_vsatSb2UtCspProcessRunning	.text	00010F98
handle_vsatSb2UtCspRetransReceivePkts	.text	000104D0
handle_vsatSb2UtCspRetransSendPkts	.text	00010588
handle_vsatSb2UtCspStartTime	.text	00010640
handle_vsatSb2UtMacConfAaaName	.text	0000A840
handle_vsatSb2UtMacConfDumpBB	.text	0000AAF0

I would initially discard this option.

TR069

As suggested in the previous blog post, the Surfbeam2 modems are deployed with the Axiros' AXACT client. The nature of the operations performed by TR069 clients makes them very convenient for an attack of this type.



```
  <obj><n>Device</n><a></a>
  <par><n>RootDataModelVersion</n><v>2.6</v><o>0</o><a>0</a></par>
  <obj><n>ManagementServer</n><a></a>
  <par><n>EnableCWMP</n><v>1</v><o>0</o><a>0</a></par>
  <par><n>URL</n><v>http://10.88.0.157:9675/live/CPEManager/CPEs/genericTR69</v><o>0</o><a>0</a></par>
  <par><n>Username</n><v>admin</v><o>0</o><a>0</a></par>
  <par><n>Password</n><v>admin</v><o>0</o><a>0</a></par>
  <par><n>PeriodicInformEnable</n><v>0</v><o>0</o><a>0</a></par>
  <par><n>PeriodicInformInterval</n><v>3600</v><o>0</o><a>0</a></par>
  <par><n>ConnectionRequestURL</n><v>http://:8089</v><o>2</o><a>0</a></par>
  <par><n>ConnectionRequestUsername</n><v>admin</v><o>0</o><a>0</a></par>
  <par><n>ConnectionRequestPassword</n><v>admin</v><o>0</o><a>0</a></par>
  </obj>
```

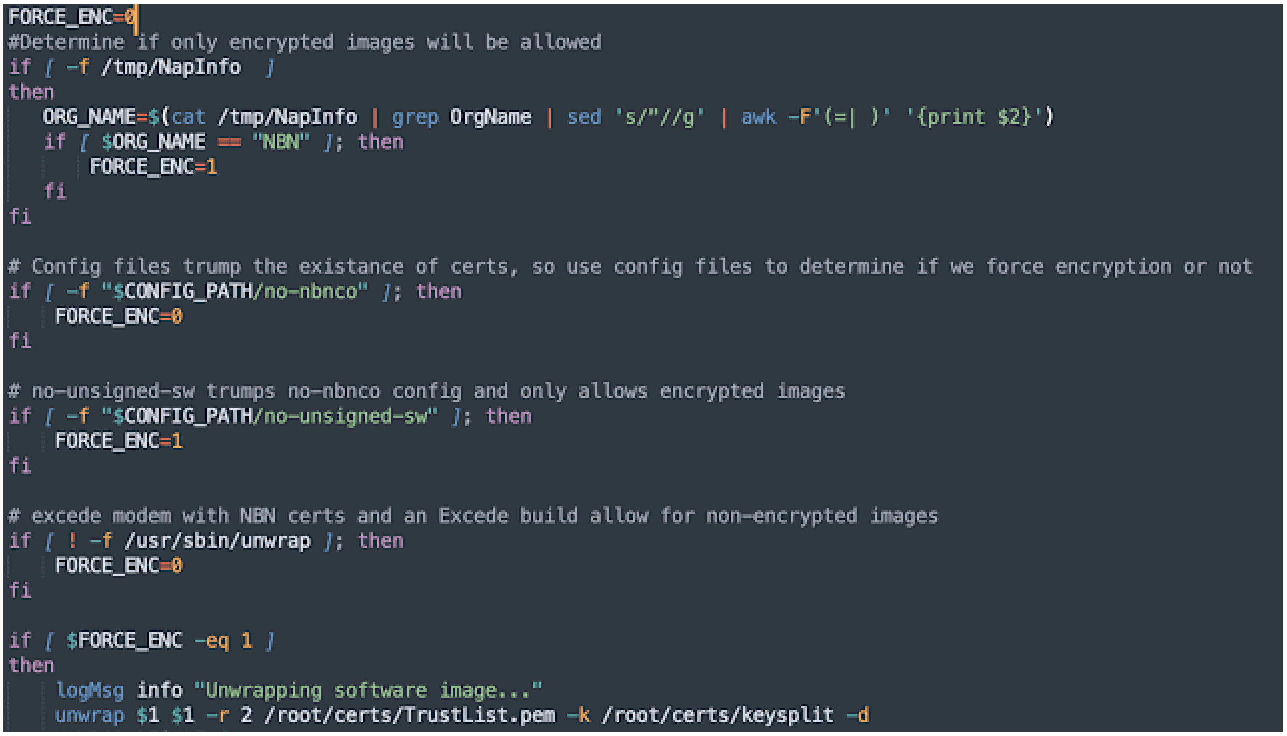
cwmpdefault.xml

By reverse engineering the 'cwmpclient' binary it is possible to recover the Viasat's TR069 data model, analyze how it has been implemented as well as how it communicates with other components to perform the required actions (via IPC queues).

So far, I would highlight the following features/issues:

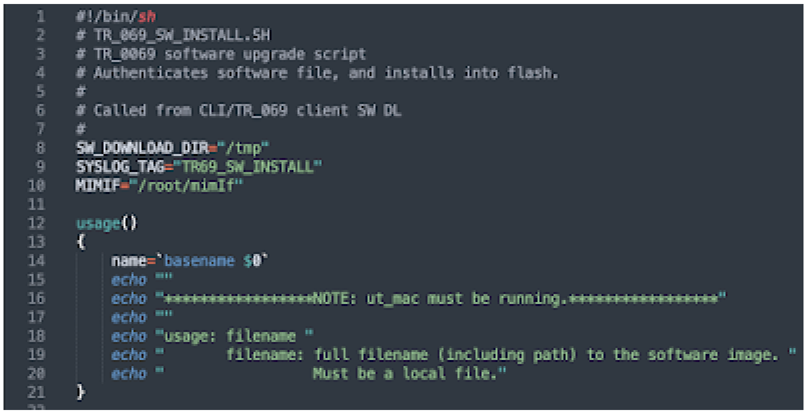
1. * Updated *

As the analysis is ongoing I want to clarify that new firmware may be cryptographically validated, after being downloaded by the TR069 client. It depends on the configuration of the terminal, according to 'sw_unwrap.sh'



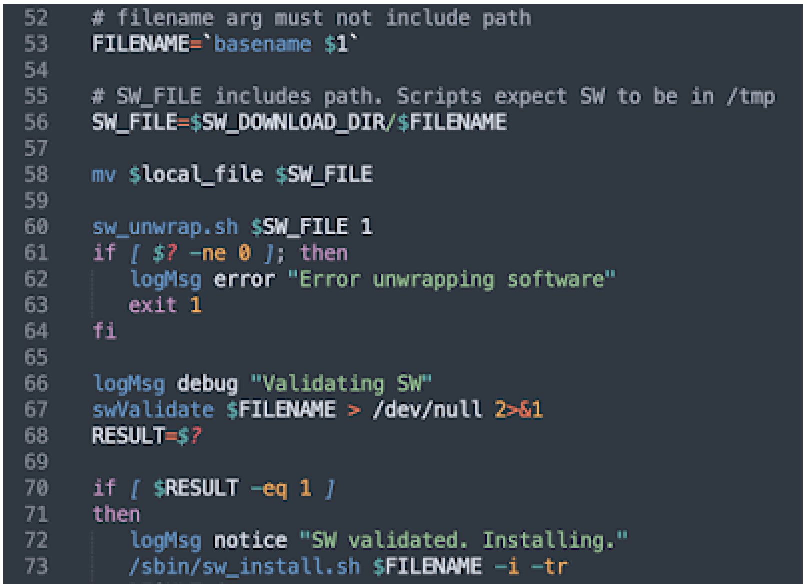
```
FORCE_ENC=0
#Determine if only encrypted images will be allowed
if [ -f /tmp/NapInfo  ]
then
   ORG_NAME=$(cat /tmp/NapInfo | grep OrgName | sed 's/"//g' | awk -F'(=| )' '{print $2}')
   if [ $ORG_NAME == "NBN" ]; then
       FORCE_ENC=1
   fi
fi

# Config files trump the existance of certs, so use config files to determine if we force encryption or not
if [ -f "$CONFIG_PATH/no-nbnco" ]; then
    FORCE_ENC=0
fi

# no-unsigned-sw trumps no-nbnco config and only allows encrypted images
if [ -f "$CONFIG_PATH/no-unsigned-sw" ]; then
    FORCE_ENC=1
fi

# excede modem with NBN certs and an Excede build allow for non-encrypted images
if [ ! -f /usr/sbin/unwrap ]; then
    FORCE_ENC=0
fi

if [ $FORCE_ENC -eq 1 ]
then
    logMsg info "Unwrapping software image..."
    unwrap $1 $1 -r 2 /root/certs/TrustList.pem -k /root/certs/keysplit -d
```

If the signature is not enforced, then the firmware image is just validated against a CRC via 'swValidate'

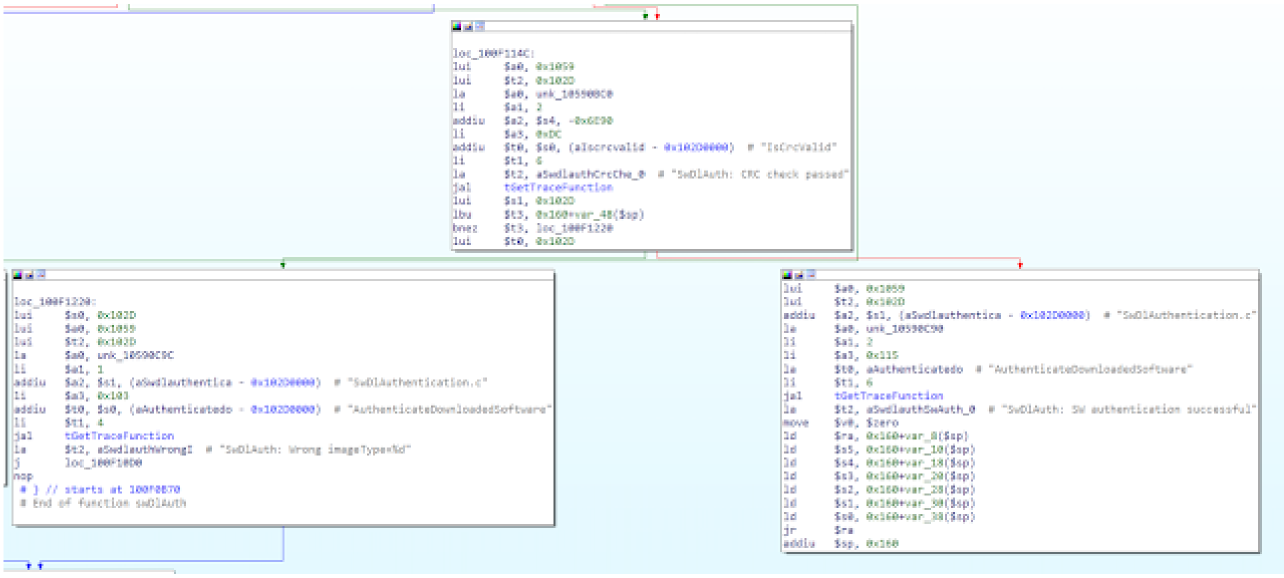


```
#!/bin/sh
# TR_069_SW_INSTALL.SH
# TR_0069 software upgrade script
# Authenticates software file, and installs into flash.
#
# Called from CLI/TR_069 client SW DL
#
SW_DOWNLOAD_DIR="/tmp"
SYSLOG_TAG="TR69_SW_INSTALL"
MIMIF="/root/mimIf"

usage()
{
    name=`basename $0`
    echo ""
    echo "******************NOTE: ut_mac must be running.******************"
    echo ""
    echo "usage: filename "
    echo "        filename: full filename (including path) to the software image. "
    echo "                        Must be a local file."
}
```

...



```
# filename arg must not include path
FILENAME=`basename $1`

# SW_FILE includes path. Scripts expect SW to be in /tmp
SW_FILE=$SW_DOWNLOAD_DIR/$FILENAME

mv $local_file $SW_FILE

sw_unwrap.sh $SW_FILE 1
if [ $? -ne 0 ]; then
    logMsg error "Error unwrapping software"
    exit 1
fi

logMsg debug "Validating SW"
swValidate $FILENAME > /dev/null 2>&1
RESULT=$?

if [ $RESULT -eq 1 ]
then
    logMsg notice "SW validated. Installing."
    /sbin/sw_install.sh $FILENAME -i -tr
```



```
loc_100F114C:
lui     $a0, 0x1059
lui     $t2, 0x102D
la      $a0, unk_105900C0
li      $a1, 3
addiu   $a2, $s4, -0x6E90
li      $a3, 0xDC
addiu   $t0, $s0, (aIscrcvalid - 0x102D0000)  # "IsCrcValid"
li      $t1, 6
la      $t2, aSwdlauthCrcChe_0  # "SwDlAuth: CRC check passed"
jal     tGetTraceFunction
lui     $s1, 0x102D
lbu     $t3, 0x160+var_48($sp)
bnez    $t3, loc_100F1220
lui     $t0, 0x102D
```

```
loc_100F1220:
lui     $s0, 0x102D
lui     $a0, 0x1059
lui     $t2, 0x102D
la      $a0, unk_10590C9C
li      $a1, 1
addiu   $a2, $s1, (aSwdlauthentica - 0x102D0000)  # "SwDlAuthentication.c"
li      $a3, 0x103
addiu   $t0, $s0, (aAuthenticatedo - 0x102D0000)  # "AuthenticateDownloadedSoftware"
li      $t1, 4
jal     tGetTraceFunction
la      $t2, aSwdlauthWrongI  # "SwDlAuth: Wrong imageType=%d"
j       loc_100F1080
nop
 # } // starts at 100F0870
 # End of function swDlAuth
```

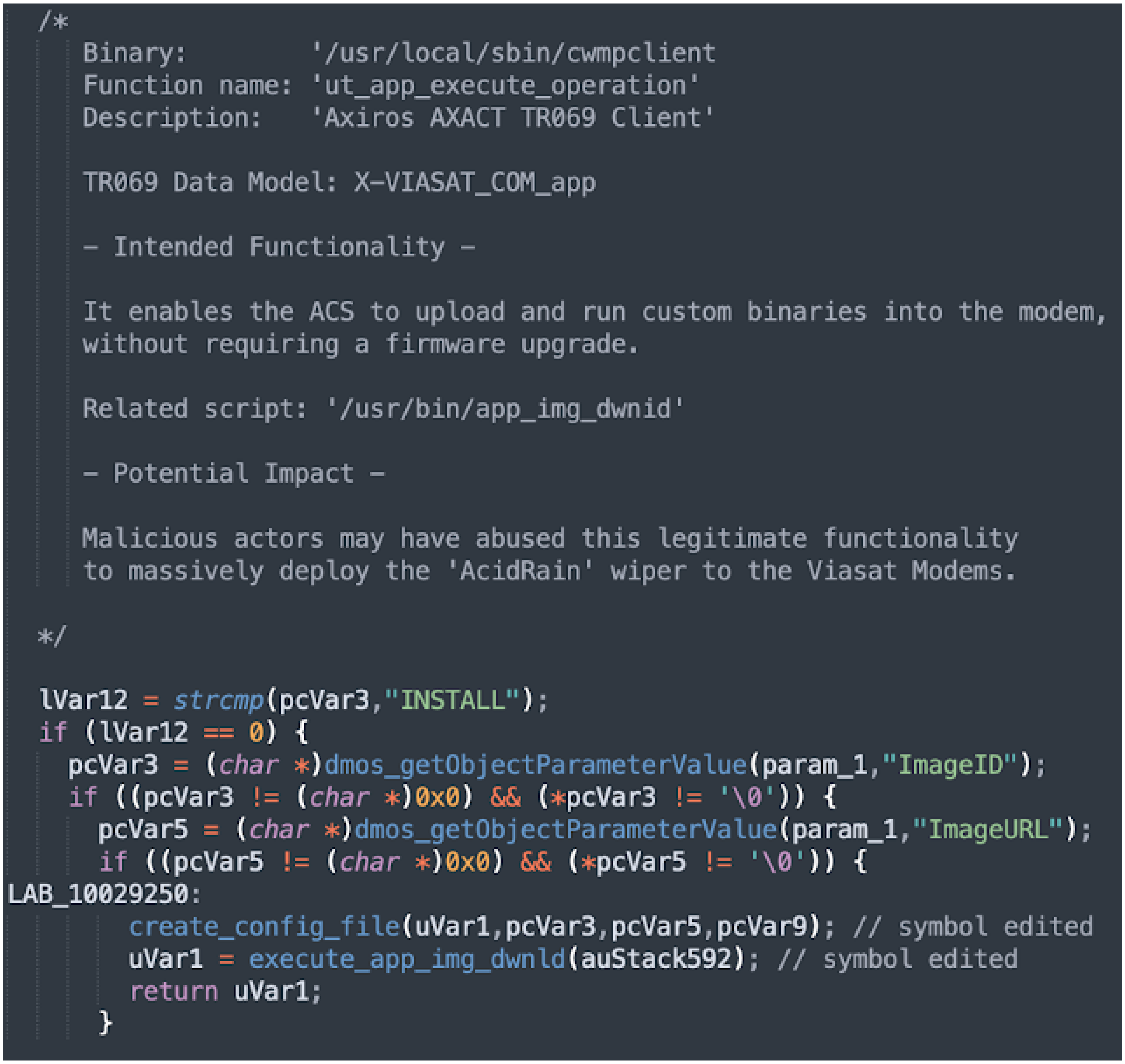
```
lui     $a0, 0x1059
lui     $t2, 0x102D
addiu   $a2, $s1, (aSwdlauthentica - 0x102D0000)  # "SwDlAuthentication.c"
la      $a0, unk_10590C90
li      $a1, 2
li      $a3, 0x115
la      $t0, aAuthenticatedo  # "AuthenticateDownloadedSoftware"
li      $t1, 6
jal     tGetTraceFunction
la      $t2, aSwdlauthSwAuth_0  # "SwDlAuth: SW authentication successful"
move    $v0, $zero
ld      $ra, 0x160+var_8($sp)
ld      $s5, 0x160+var_10($sp)
ld      $s4, 0x160+var_18($sp)
ld      $s3, 0x160+var_20($sp)
ld      $s2, 0x160+var_28($sp)
ld      $s1, 0x160+var_30($sp)
ld      $s0, 0x160+var_38($sp)
jr      $ra
addiu   $sp, 0x160
```

swValidate (implemented in 'ut_mac' binary)

2. * Updated * 'APP INSTALL'

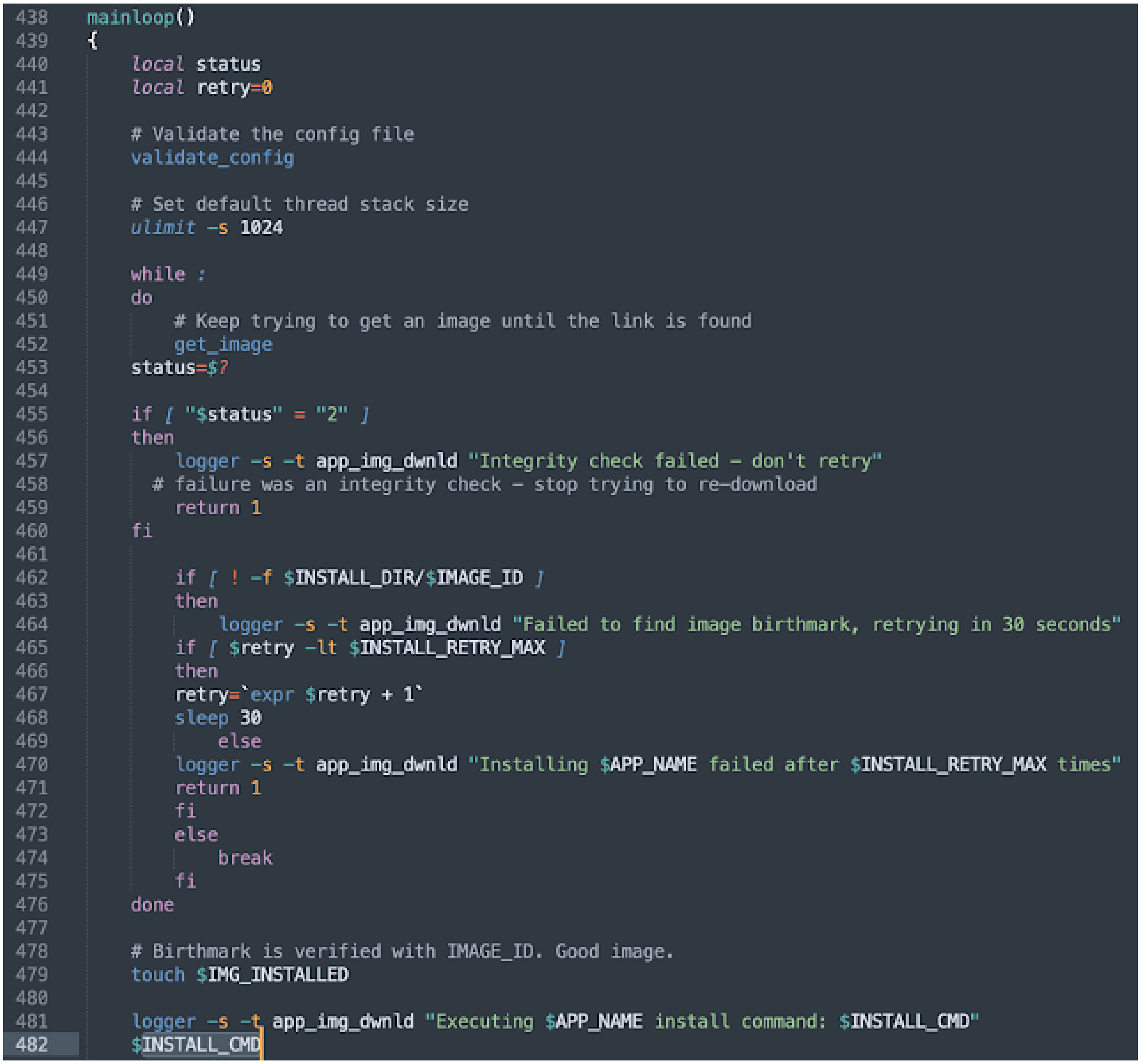
A deeper look at the 'ut_app_execute_operation' function revealed that it is implementing a functionality that enables the ACS to install (upload and run) arbitrary binaries on the modem, without requiring either a signature verification or a complete firmware upgrade.

This functionality seems to match both the Viasat statement as well as the approach to deploy the 'AcidRain' wiper described by SentinelOne.



```
  /*
     Binary:        '/usr/local/sbin/cwmpclient
     Function name: 'ut_app_execute_operation'
     Description:   'Axiros AXACT TR069 Client'

     TR069 Data Model: X-VIASAT_COM_app

     - Intended Functionality -

     It enables the ACS to upload and run custom binaries into the modem,
     without requiring a firmware upgrade.

     Related script: '/usr/bin/app_img_dwnid'

     - Potential Impact -

     Malicious actors may have abused this legitimate functionality
     to massively deploy the 'AcidRain' wiper to the Viasat Modems.

  */

  lVar12 = strcmp(pcVar3,"INSTALL");
  if (lVar12 == 0) {
    pcVar3 = (char *)dmos_getObjectParameterValue(param_1,"ImageID");
    if ((pcVar3 != (char *)0x0) && (*pcVar3 != '\0')) {
      pcVar5 = (char *)dmos_getObjectParameterValue(param_1,"ImageURL");
      if ((pcVar5 != (char *)0x0) && (*pcVar5 != '\0')) {
LAB_10029250:
        create_config_file(uVar1,pcVar3,pcVar5,pcVar9); // symbol edited
        uVar1 = execute_app_img_dwnld(auStack592); // symbol edited
        return uVar1;
      }
```

'/usr/bin/app_img_dwnid'

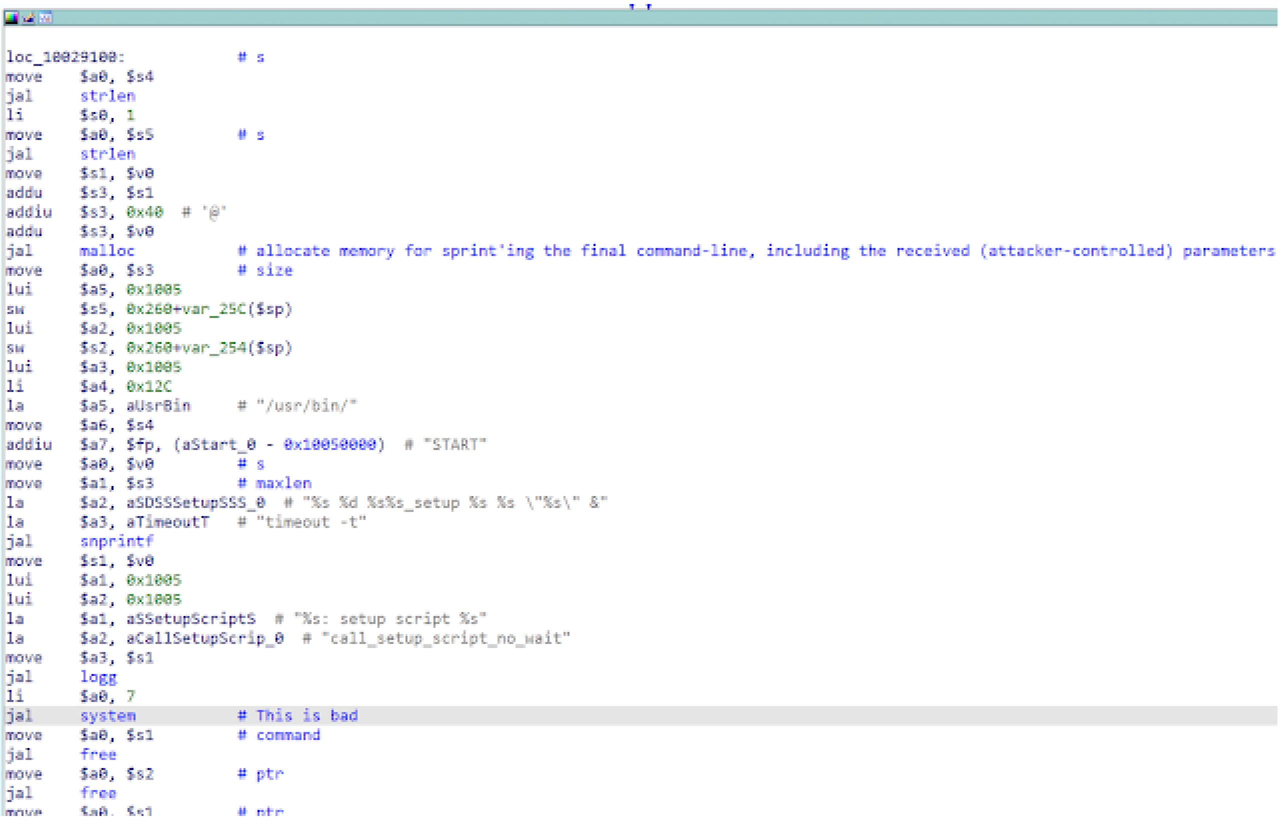


```
mainloop()
{
    local status
    local retry=0

    # Validate the config file
    validate_config

    # Set default thread stack size
    ulimit -s 1024

    while :
    do
        # Keep trying to get an image until the link is found
        get_image
    status=$?

    if [ "$status" = "2" ]
    then
        logger -s -t app_img_dwnld "Integrity check failed - don't retry"
      # failure was an integrity check - stop trying to re-download
        return 1
    fi

        if [ ! -f $INSTALL_DIR/$IMAGE_ID ]
        then
            logger -s -t app_img_dwnld "Failed to find image birthmark, retrying in 30 seconds"
        if [ $retry -lt $INSTALL_RETRY_MAX ]
        then
        retry=`expr $retry + 1`
        sleep 30
            else
        logger -s -t app_img_dwnld "Installing $APP_NAME failed after $INSTALL_RETRY_MAX times"
        return 1
        fi
        else
            break
        fi
    done

    # Birthmark is verified with IMAGE_ID. Good image.
    touch $IMG_INSTALLED

    logger -s -t app_img_dwnld "Executing $APP_NAME install command: $INSTALL_CMD"
    $INSTALL_CMD
```

Command Injections

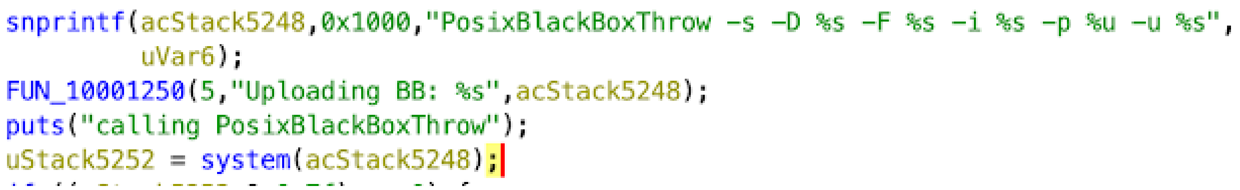
Additionally, there are multiple command injection vulnerabilities that can be trivially exploited from a malicious ACS (or someone with the same privileged position in the network).

i.e 'ut_app_execute_operation' for the custom 'Device.Services.X_VIASAT-COM_app' object ('cwmpclient')



```
loc_10029100:           # s
move    $a0, $s4
jal     strlen
li      $s0, 1
move    $a0, $s5        # s
jal     strlen
move    $s1, $v0
addu    $s3, $s1
addiu   $s3, 0x40  # '@'
addu    $s3, $v0
jal     malloc          # allocate memory for sprint'ing the final command-line, including the received (attacker-controlled) parameters
move    $a0, $s3        # size
lui     $a5, 0x1005
sw      $s5, 0x260+var_25C($sp)
lui     $a2, 0x1005
sw      $s2, 0x260+var_254($sp)
lui     $a3, 0x1005
li      $a4, 0x12C
la      $a5, aUsrBin     # "/usr/bin/"
move    $a6, $s4
addiu   $a7, $fp, (aStart_0 - 0x10050000)  # "START"
move    $a0, $v0        # s
move    $a1, $s3        # maxlen
la      $a2, aSDSSSetupSSS_0  # "%s %d %s%s_setup %s %s \"%s\" &"
la      $a3, aTimeoutT   # "timeout -t"
jal     snprintf
move    $s1, $v0
lui     $a1, 0x1005
lui     $a2, 0x1005
la      $a1, aSSetupScriptS  # "%s: setup script %s"
la      $a2, aCallSetupScrip_0  # "call_setup_script_no_wait"
move    $a3, $s1
jal     logg
li      $a0, 7
jal     system          # This is bad
move    $a0, $s1        # command
jal     free
move    $a0, $s2        # ptr
jal     free
move    $a0, $s1        # ptr
```

Also in '/usr/bin/bbagent' (listening on *:8700/TCP, when activated)



```
snprintf(acStack5248,0x1000,"PosixBlackBoxThrow -s -D %s -F %s -i %s -p %u -u %s",
               uVar6);
FUN_10001250(5,"Uploading BB: %s",acStack5248);
puts("calling PosixBlackBoxThrow");
uStack5252 = system(acStack5248);
```

'Lifeline' - Firmware update over multicast

This is an interesting 'emergency' feature intended to perform a firmware upgrade over a specific Multicast group, when everything else fails. It's implemented across different binaries: 'ut_mac', 'mim', 'mimIf' and 'lifelineClient'

Conclusion

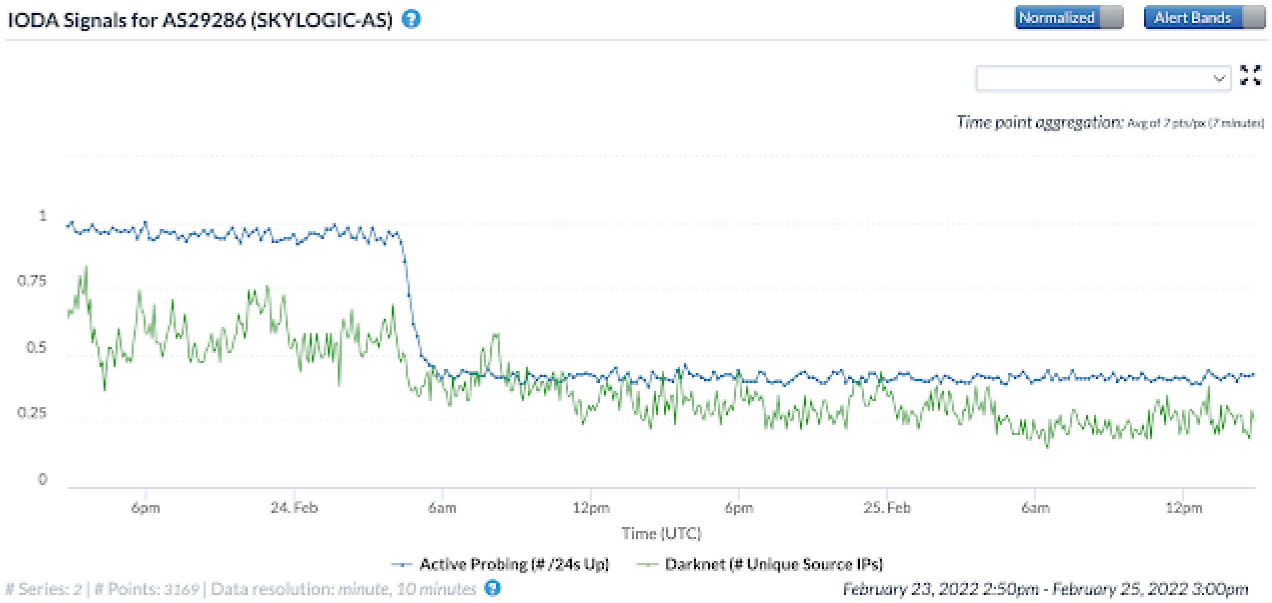
There are similarities between these issues and the approach followed by the attackers in the Viasat incident, especially the TR069 'APP INSTALL' feature, but I am not implying that any of these techniques were actually abused by the attackers. However, overall the security posture of the Surfbeam2 firmware does not look good.

Hopefully these vulnerabilities are no longer present in the newest Viasat firmware, otherwise that may pose a security risk.

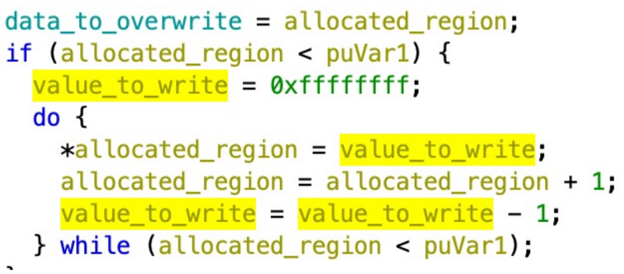
There are several unknowns yet to be resolved.

1. How the initial compromise of the VPN appliance worked. Did the attackers have valid credentials (maybe stolen from either Skylogic or its partners) or they exploited a known vulnerability (assuming an 0day doesn't match a 'misconfigured VPN appliance' explanation)?

2. How exactly the attack propagated to other countries, lasting for several hours. One of the affected persons I talked to got his modem knocked out around 9:00 am (GMT+1), several hours after the initial attack.



3. Before the destructive payload was executed, there was any other kind of malicious code running in the modems for a short period of time? Sentinelone published a very interesting research on 'AcidRain', a wiper that is able to generate the same destructive pattern observed in the modem's flash memory.


```
data_to_overwrite = allocated_region;
if (allocated_region < puVar1) {
  value_to_write = 0xffffffff;
  do {
    *allocated_region = value_to_write;
    allocated_region = allocated_region + 1;
    value_to_write = value_to_write - 1;
  } while (allocated_region < puVar1);
```

Coincidentally, this wiper also has similarities with 'VPNfilter' malware.

4. Did the compromise of the management segment involve additional attacks besides the VPN issue?

Unfortunately these technical questions can only be answered by people with an insider knowledge. Let's see if Viasat is willing to provide further details on this case.

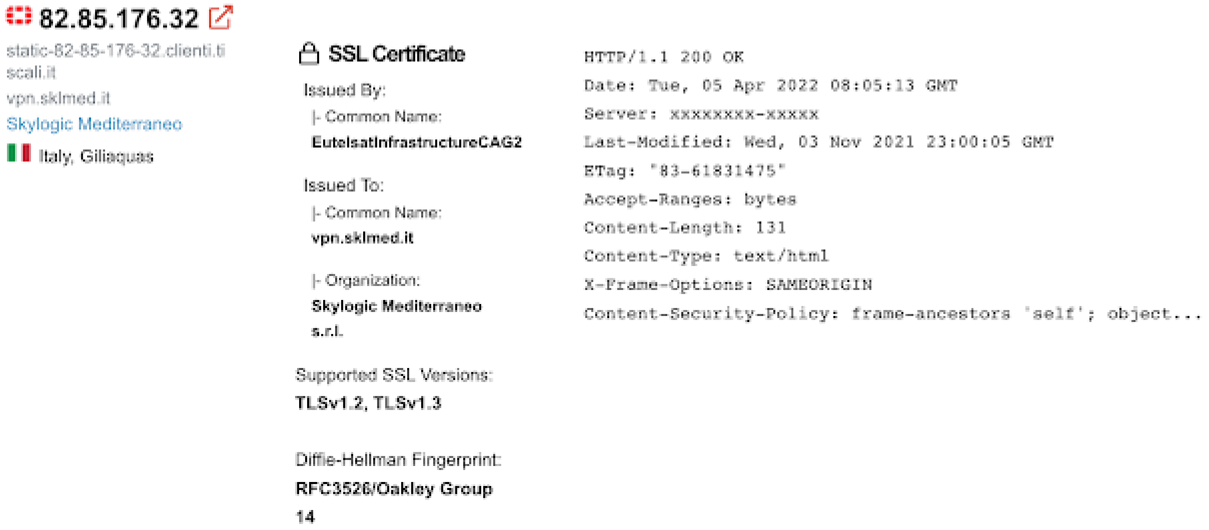
*** Updated - The VPN Attack vector***

Viasat has not elaborated the VPN attack vector yet, but they acknowledged to journalists that the attack originated from the Internet. Viasat is also distancing itself from the fiasco by directly pointing to Skylogic and its ground infrastructure.

Although we're entering again the land of speculation, there are some factual bits that should be considered.

A simple recon of Skylogic's ground network (AS201935) reveals a couple of interesting things:

1. Skylogic relies on Fortigate appliances



82.85.176.32

static-82-85-176-32.clienti.tiscali.it
vpn.sklmed.it
Skylogic Mediterraneo
Italy, Giliaquas

SSL Certificate

Issued By:
|- Common Name:
EutelsatInfrastructureCAG2

Issued To:
|- Common Name:
vpn.sklmed.it

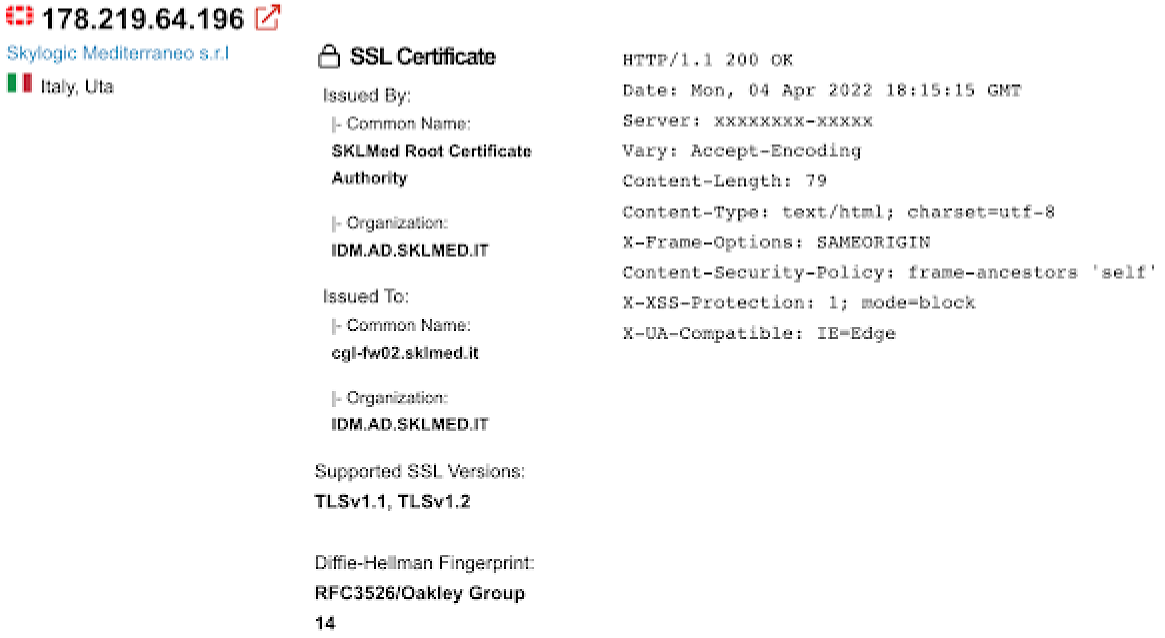
|- Organization:
Skylogic Mediterraneo s.r.l.

Supported SSL Versions:
TLSv1.2, TLSv1.3

Diffie-Hellman Fingerprint:
RFC3526/Oakley Group 14

```
HTTP/1.1 200 OK
Date: Tue, 05 Apr 2022 08:05:13 GMT
Server: xxxxxxxx-xxxxx
Last-Modified: Wed, 03 Nov 2021 23:00:05 GMT
ETag: "83-61831475"
Accept-Ranges: bytes
Content-Length: 131
Content-Type: text/html
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'; object...
```

'cgl-fw02' may be indicating the Skylogic's Cagliari teleport.



178.219.64.196

Skylogic Mediterraneo s.r.l

Italy, Uta

SSL Certificate

Issued By:

|- Common Name:

SKLMed Root Certificate Authority

|- Organization:

IDM.AD.SKLMED.IT

Issued To:

|- Common Name:

cgl-fw02.sklmed.it

|- Organization:

IDM.AD.SKLMED.IT

Supported SSL Versions:

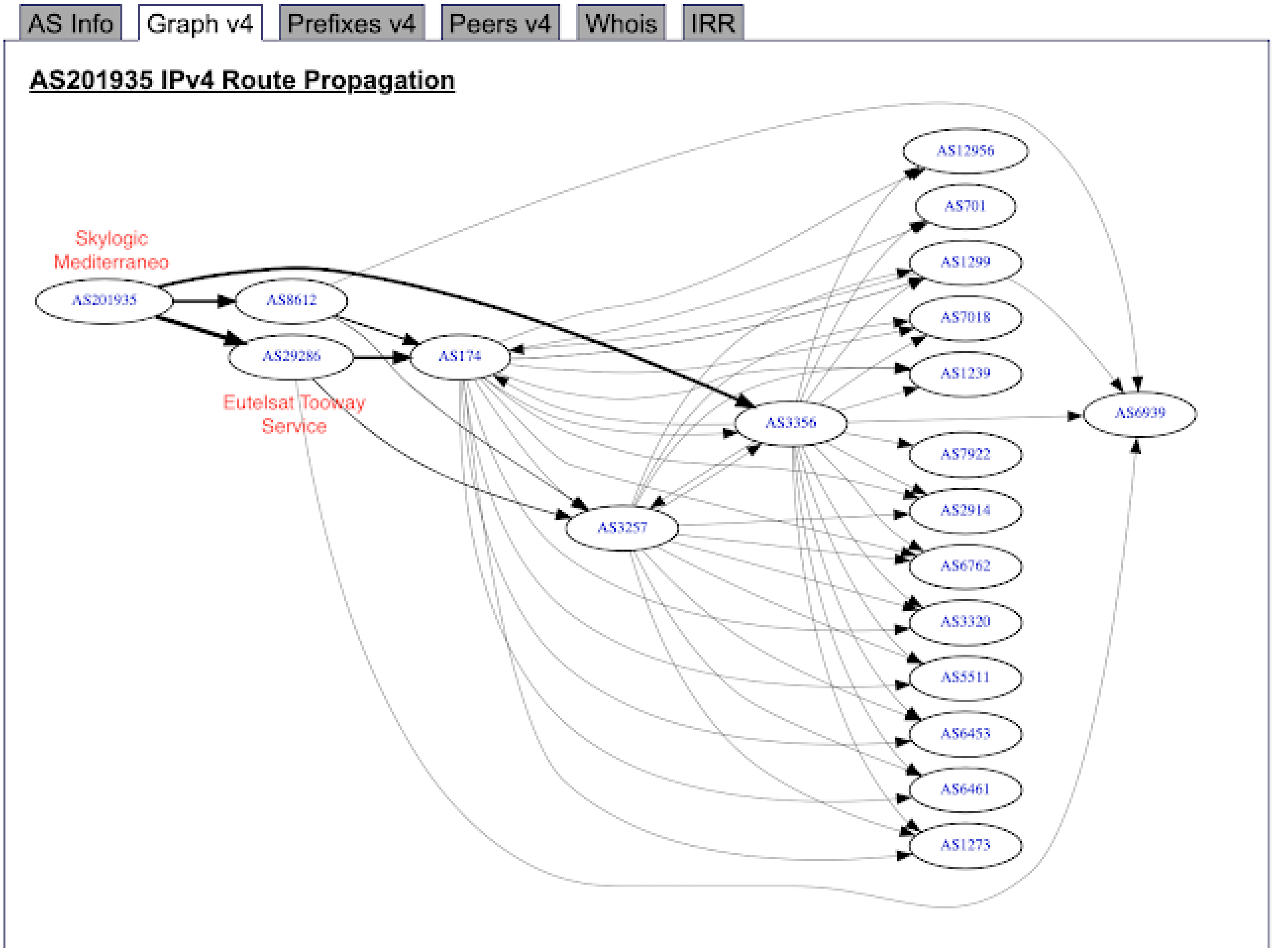
TLSv1.1, TLSv1.2

Diffie-Hellman Fingerprint:

RFC3526/Oakley Group 14

```
HTTP/1.1 200 OK
Date: Mon, 04 Apr 2022 18:15:15 GMT
Server: xxxxxxxx-xxxxx
Vary: Accept-Encoding
Content-Length: 79
Content-Type: text/html; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'
X-XSS-Protection: 1; mode=block
X-UA-Compatible: IE=Edge
```

2. The route propagation matches the attack. Viasat's statement explicitly mentions that the attacker moved laterally until reaching the management network.



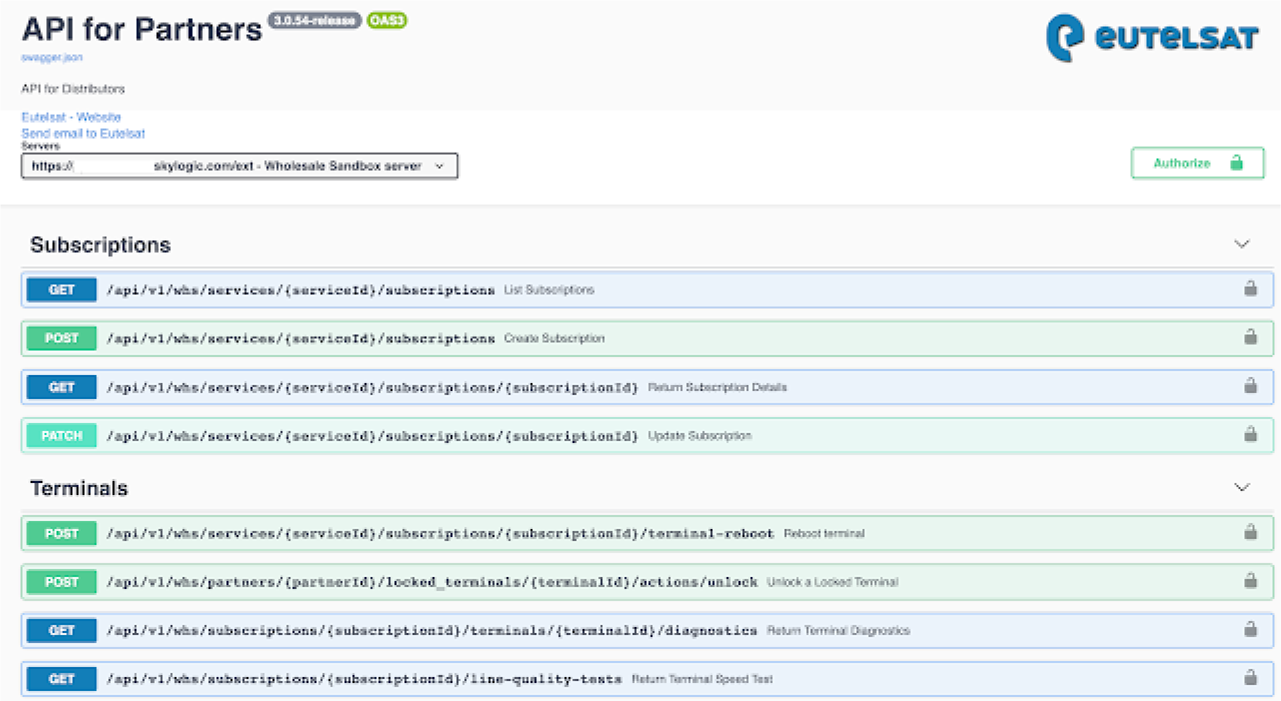
AS201935 IPv4 Route Propagation

It is also worth mentioning that, in 2021, there were different attack campaigns and leaks targeting Fortinet VPN appliances. These attacks were carried out by groups of malicious actors that exploited multiple vulnerabilities that were discovered in these products.

Viasat's statement mentions a 'misconfigured VPN appliance', so if we consider that this definition may be a euphemism for an 'unpatched VPN appliance', then we may have a plausible attack vector. It is also possible that malicious actors may have previously collected valid VPN credentials as a result of these attacks.

Another interesting aspect that Viasat implicitly introduces in its statement is the potential security weaknesses that may be derived from the complexity of wholesale operations for a Satellite infrastructure. Down in this chain we find ground station operators, satellite service providers, distributors , resellers...

At some point they all need certain kind of access to provide their services, so this integration also may pose a challenge in terms of security. For instance, a publicly exposed server provides a glimpse of the Eutelsat's partners API capabilities.



API for Partners

swagger.json

API for Distributors

Eutelsat - Website
Send email to Eutelsat

https:// skylogic.com/ext - Wholesale Sandbox server

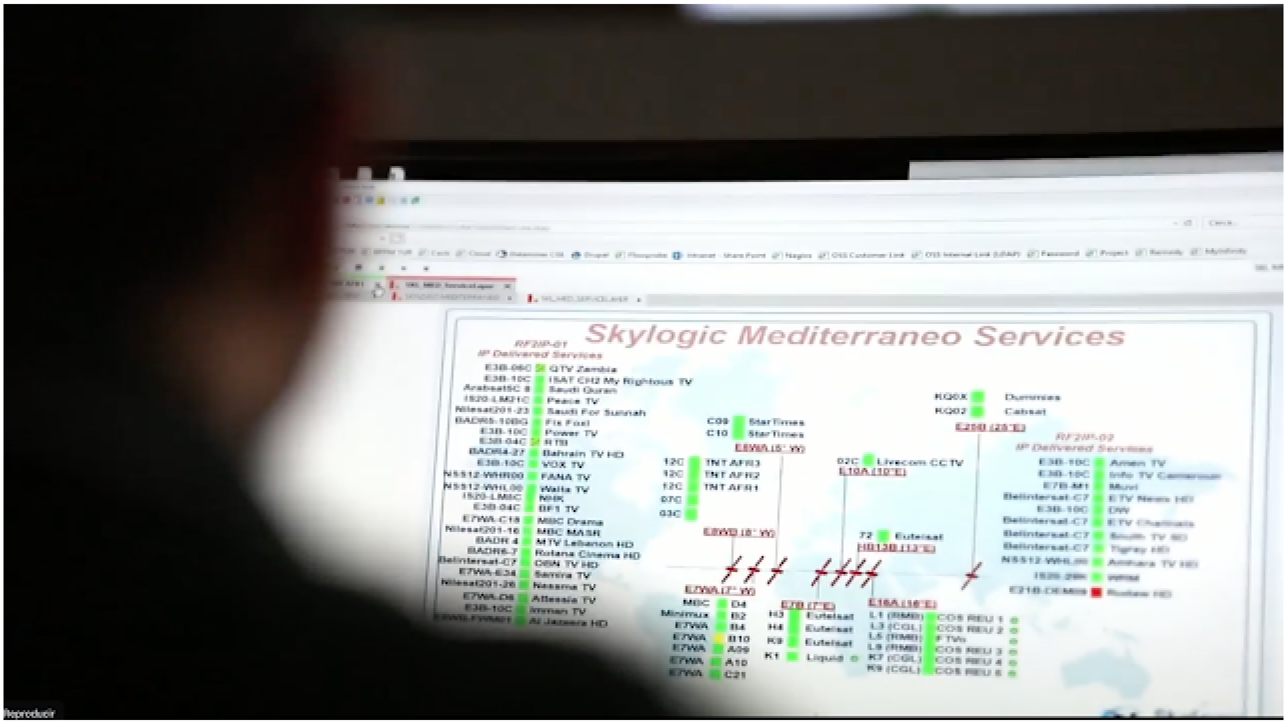
Subscriptions

```
GET    /api/v1/whs/services/{serviceId}/subscriptions List Subscriptions
POST   /api/v1/whs/services/{serviceId}/subscriptions Create Subscription
GET    /api/v1/whs/services/{serviceId}/subscriptions/{subscriptionId} Return Subscription Details
PATCH  /api/v1/whs/services/{serviceId}/subscriptions/{subscriptionId} Update Subscription
```

Terminals

```
POST   /api/v1/whs/services/{serviceId}/subscriptions/{subscriptionId}/terminal-reboot Reboot terminal
POST   /api/v1/whs/partners/{partnerId}/locked_terminals/{terminalId}/actions/unlock Unlock a Locked Terminal
GET    /api/v1/whs/subscriptions/{subscriptionId}/terminals/{terminalId}/diagnostics Return Terminal Diagnostics
GET    /api/v1/whs/subscriptions/{subscriptionId}/line-quality-tests Return Terminal Speed Test
```

In general terms, it is also recommended to not expose an operator's desktop in corporate videos. It usually leaks information that may facilitate different kinds of attacks.



SATCOM terminals under attack in Europe: a plausible analysis.

------ Update 03/12/2022 Reuters has published new information on this incident, which initially matches the proposed scenario. You can find the update at the bottom of this post. ------ February 24th: at the same time Russia initiated a full-scale attack on Ukraine, tens of thousands of KA-SAT SATCOM terminals suddenly stopped working in several european countries: Germany, Ukraine, Greece, Hungary, Poland...Germany's Enercon moved forward and acknowledged that approximately 5800 of its wind turbines, presumably those remotely operated via a SATCOM link in central Europe, had lost contact with their SCADA server . In the affected countries, a significant part of the customers of Eutelsat's domestic broadband service were also unable to access Internet. From the very beginning Eutelsat and its parent company Viasat, stated that the issue was being investigated as a cyberattack. Since then, details have been scarcely provided but few days ago I came across a really inter

Finding vulnerabilities in Swiss Post’s future e-voting system - Part 1

In September '21, I came across this story "Swiss Post Offers up to €230,000 for Critical Vulnerabilities in e-Voting System" while catching up with the security news. The headline certainly caught my attention as it looked like an outlier from the regular bug bounty programs or well-known exploit contests, not only for the announced rewards but mainly because of the target. So essentially Swiss Post , the national postal service of Switzerland, was opening to the general public a bug bounty program, using the YesWeHack platform, intended to uncover vulnerabilities in its future e-voting system. The first part of this blog post series will detail the approach used to analyze the Swiss Post e-voting system, as well as the first round of vulnerabilities that I reported during September/October '21. Index Introduction Approach Attack Surface Vulnerabilities 1. Insecure USB file handling during 'importOperation' 2. Insecure 'ReturnCodeGenerationI