

VIASAT incident: from speculation to technical details.

Published: 2022-03-31 · Archived: 2026-04-05 15:06:30 UTC

VIASAT incident: from speculation to technical details.

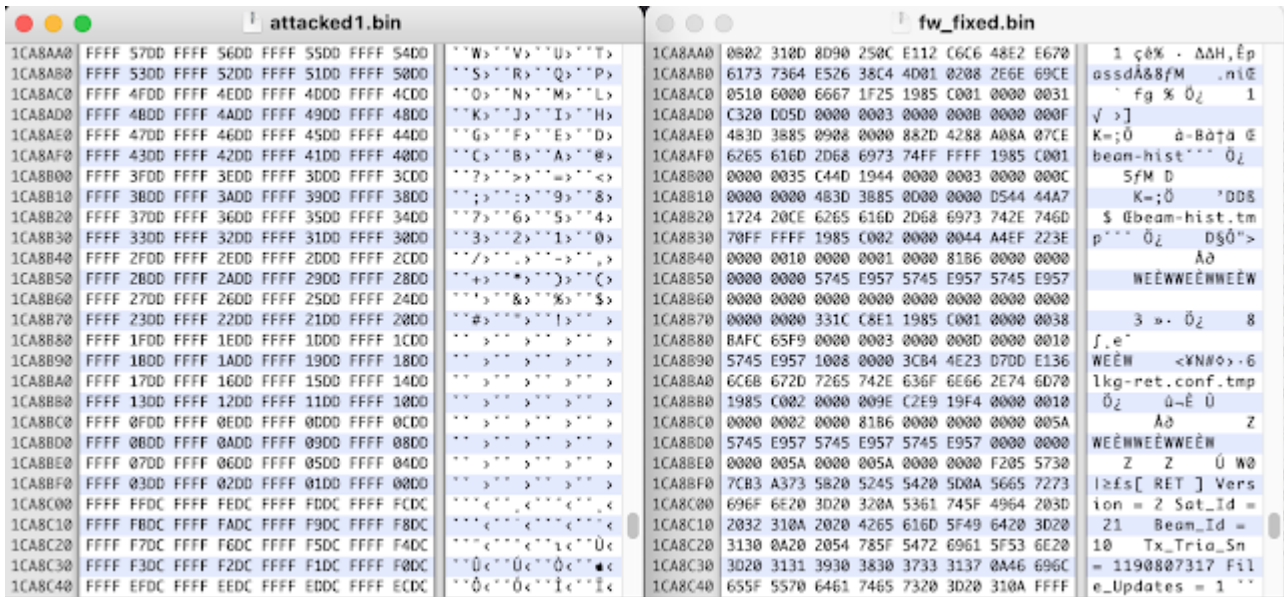
34 days after the incident, yesterday Viasat [published](#) a statement providing some technical details about the attack that affected tens of thousands of its SATCOM terminals. Also yesterday, I eventually had access to two Surfbeam2 modems: one was targeted during the attack and the other was in a 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.



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

```
1 trapcommunity public
2 rocommunity public default -V xperf
3 rwcommunity private default -V xperf
4
5 engineIDType 3
6 engineIDNic eth0
7
8 createUser viasat SHA "75@t1133" AES
9
10 rwuser viasat
11
12 dlmod vsatSb2Ut /usr/local/share/snmp/dlmod/vsatSb2Ut.so
13
```

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_vsatsb2utCspLastFIPktsLost	.text	000109D8
 handle_vsatsb2utCspLastRIPktsLost	.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/CPManager/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 existence 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
```

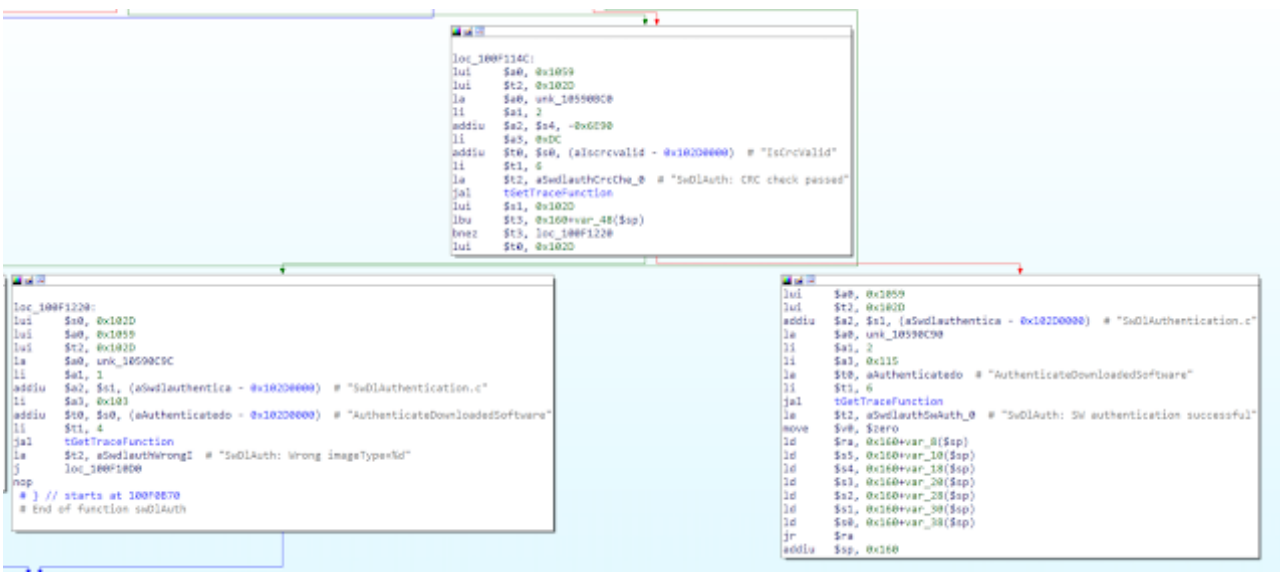
If the signature is not enforced, then the firmware image is just validated against a CRC via 'swValidate'

```
1  #!/bin/sh
2  # TR_069_SW_INSTALL.SH
3  # TR_069 software upgrade script
4  # Authenticates software file, and installs into flash.
5  #
6  # Called from CLI/TR_069 client SW DL
7  #
8  SW_DOWNLOAD_DIR="/tmp"
9  SYSLOG_TAG="TR069_SW_INSTALL"
10 MIMIF="/root/mimif"
11
12 usage()
13 {
14     name=`basename $0`
15     echo ""
16     echo "*****NOTE: ut_mac must be running.*****"
17     echo ""
18     echo "usage: filename "
19     echo "      filename: full filename (including path) to the software image. "
20     echo "      Must be a local file."
21 }
22
```

...

```
52 # filename arg must not include path
53 FILENAME=`basename $1`
54
55 # SW_FILE includes path. Scripts expect SW to be in /tmp
56 SW_FILE=$SW_DOWNLOAD_DIR/$FILENAME
57
58 mv $local_file $SW_FILE
59
60 sw_unwrap.sh $SW_FILE 1
61 if [ $? -ne 0 ]; then
62     logMsg error "Error unwrapping software"
63     exit 1
64 fi
65
66 logMsg debug "Validating SW"
67 swValidate $FILENAME > /dev/null 2>&1
68 RESULT=$?
69
70 if [ $RESULT -eq 1 ]
71 then
72     logMsg notice "SW validated. Installing."
73     /sbin/sw_install.sh $FILENAME -i -tr

```



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'

```
438 mainloop()
439 {
440     local status
441     local retry=0
442
443     # Validate the config file
444     validate_config
445
446     # Set default thread stack size
447     ulimit -s 1024
448
449     while :
450     do
451         # Keep trying to get an image until the link is found
452         get_image
453         status=$?
454
455         if [ "$status" = "2" ]
456         then
457             logger -s -t app_img_dwnld "Integrity check failed - don't retry"
458             # failure was an integrity check - stop trying to re-download
459             return 1
460         fi
461
462         if [ ! -f $INSTALL_DIR/$IMAGE_ID ]
463         then
464             logger -s -t app_img_dwnld "Failed to find image birthmark, retrying in 30 seconds"
465             if [ $retry -lt $INSTALL_RETRY_MAX ]
466             then
467                 retry=`expr $retry + 1`
468                 sleep 30
469             else
470                 logger -s -t app_img_dwnld "Installing $APP_NAME failed after $INSTALL_RETRY_MAX times"
471                 return 1
472             fi
473         else
474             break
475         fi
476     done
477
478     # Birthmark is verified with IMAGE_ID. Good image.
479     touch $IMG_INSTALLED
480
481     logger -s -t app_img_dwnld "Executing $APP_NAME install command: $INSTALL_CMD"
482     $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 ('cwmplclient')

```

loc_10029100:          # s
move    $a0, $s4
jal     strlen
li      $s0, 1
move    $a0, $s5          # s
jal     strlen
move    $s1, $v0
addiu   $s3, $s1
addiu   $s3, 0x40 # '@'
addiu   $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      $s5, 0x268+var_25C($sp)
lui     $a2, 0x1005
sw      $s2, 0x268+var_254($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, aSOSSSSetupSSS_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          # ntr

```

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);

```

'Lifecycle' - 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

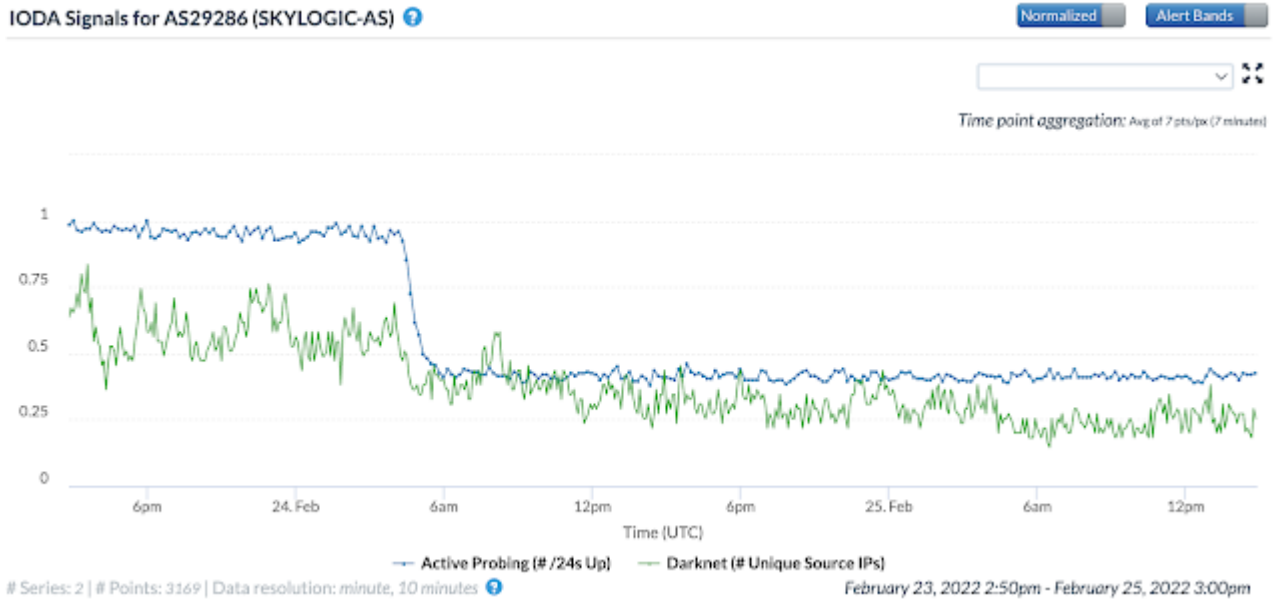
There are similarities between these issues and the approach followed by the attackers in the Viasat incident, especially the TR069 'APP INSTALL' feature, but I am not implying that any of these techniques were actually abused by the attackers. However, overall the security posture of the Surfbeam2 firmware 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.ti
scali.it
vpn.skimed.it
Skylogic Mediterraneo
 Italy, Giliaguas

SSL Certificate

Issued By:
|- Common Name:
EutelsatInfrastructureCAG2

Issued To:
|- Common Name:
vpn.skimed.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-xxxxxx
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.skimed.it

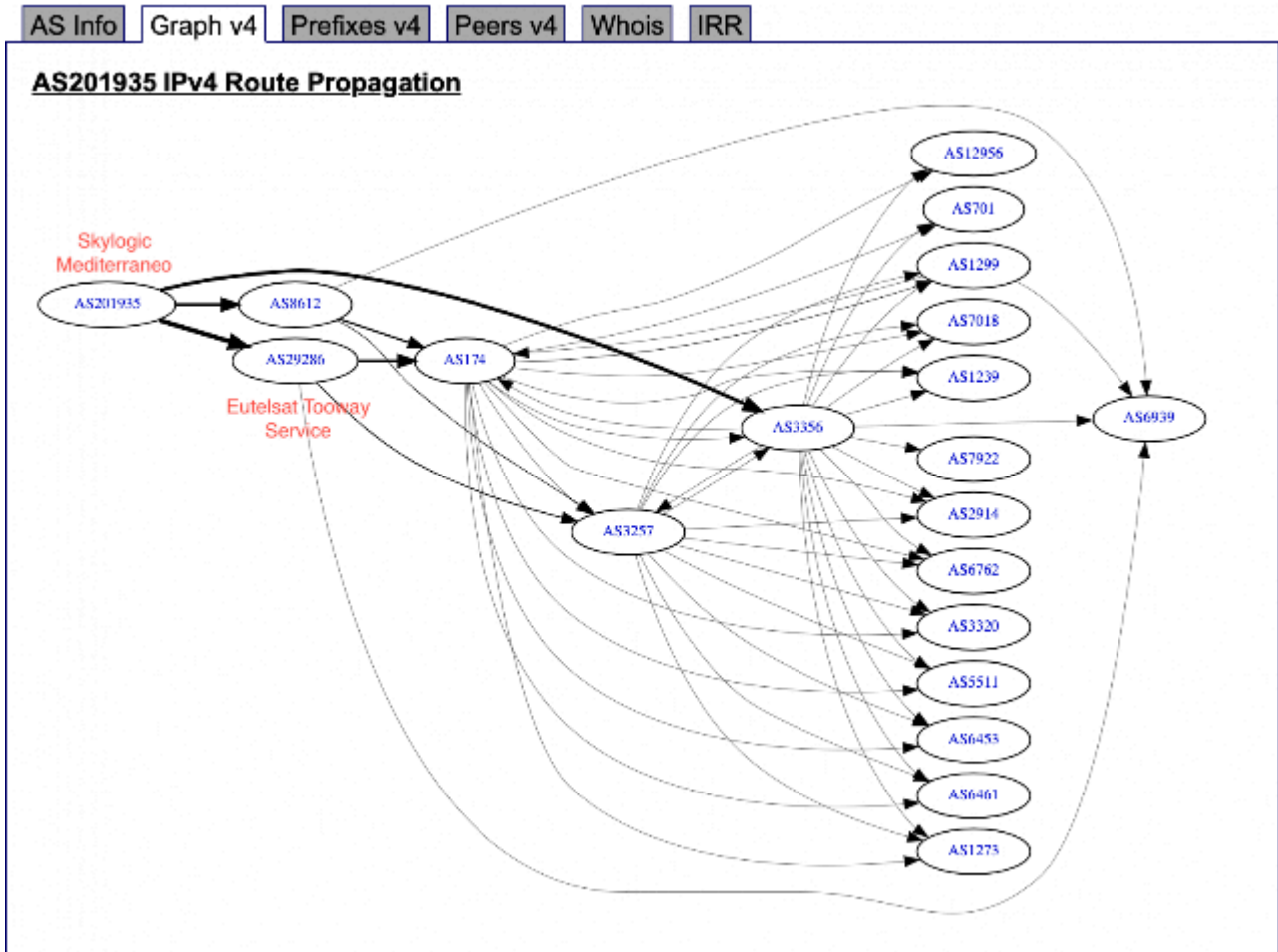
|- Organization:
IDM.AD.SKLMED.IT

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-xxxxxx
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.




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 3.0.54-release OAS3 

swagger.json
API for Distributors
Eutelsat - Website
Send email to Eutelsat Servers

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

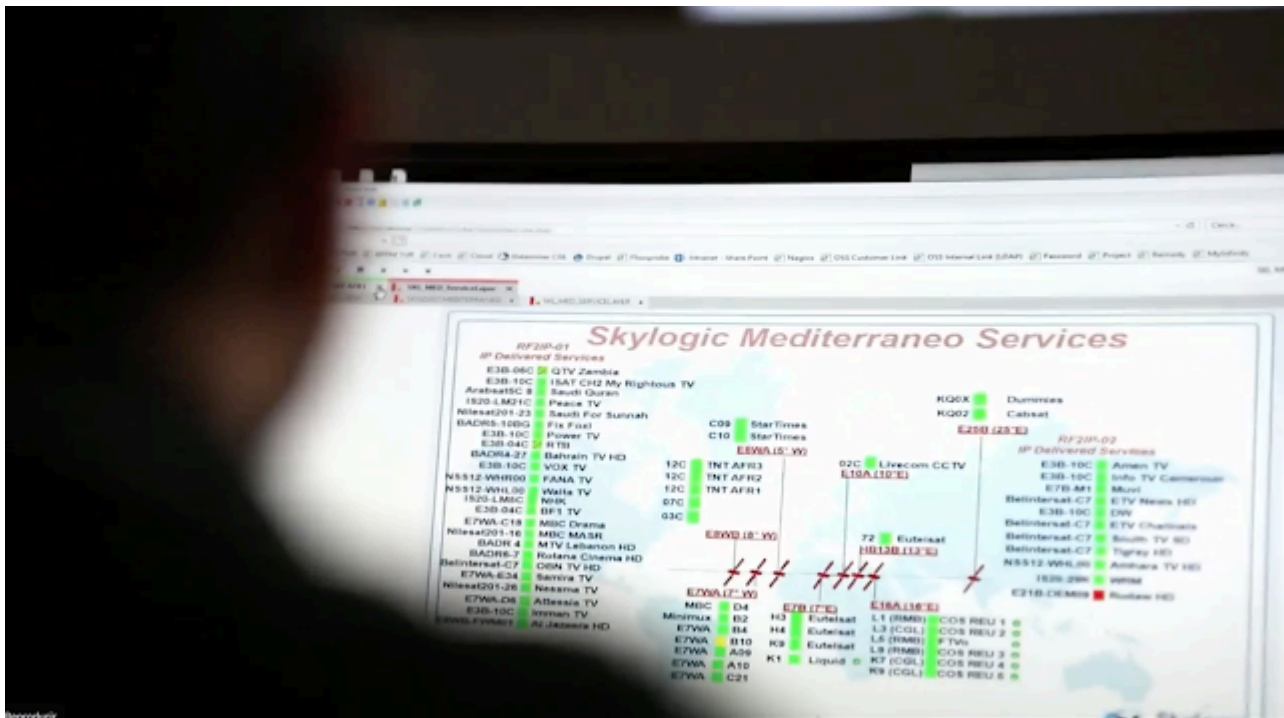
Subscriptions

- 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.



Source: <https://www.reversemode.com/2022/03/viasat-incident-from-speculation-to.html>