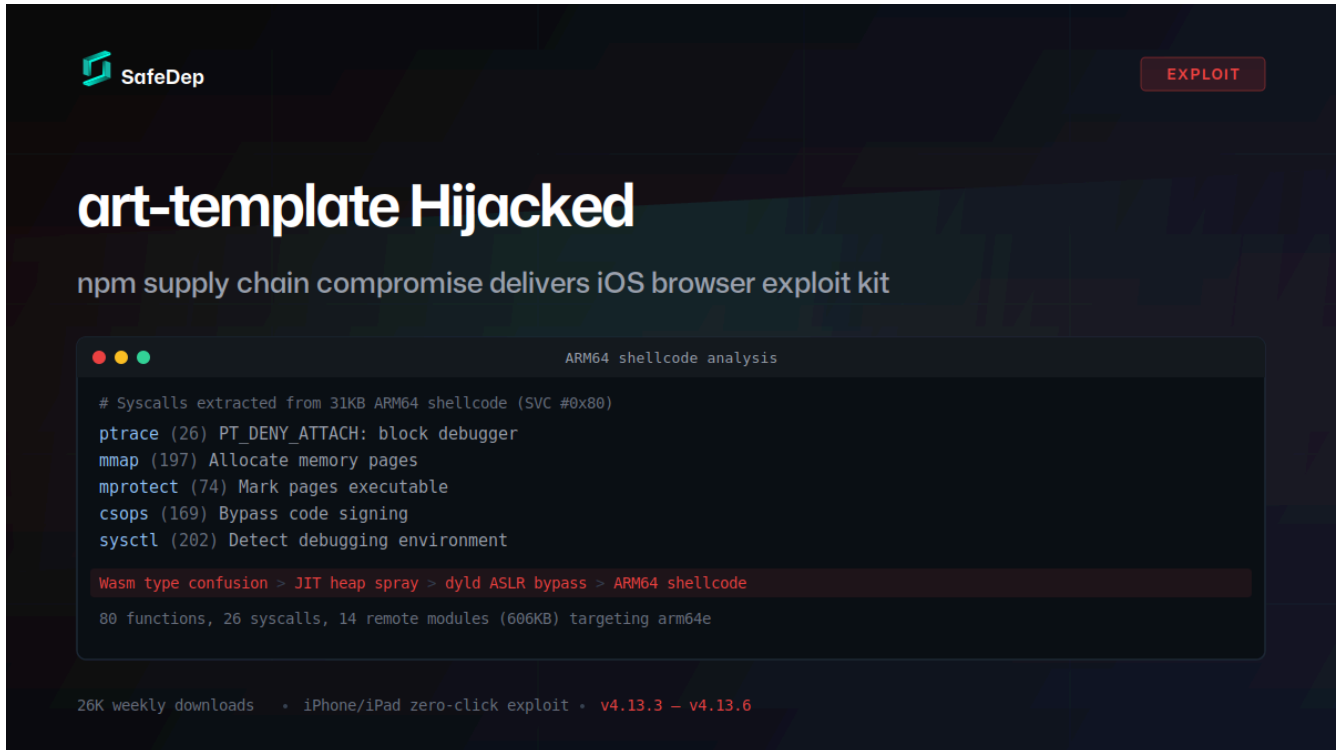


Compromised node-ipc on npm: Credential Stealer via DNS Exfiltration

 safedep.io/malicious-node-ipc-npm-compromise



art-template Hijacked

npm supply chain compromise delivers iOS browser exploit kit

```
ARM64 shellcode analysis

# Syscalls extracted from 31KB ARM64 shellcode (SVC #0xB0)
ptrace (26) PT_DENY_ATTACH: block debugger
mmap (197) Allocate memory pages
mprotect (74) Mark pages executable
csops (169) Bypass code signing
sysctl (202) Detect debugging environment

Wasm type confusion > JIT heap spray > dyld ASLR bypass > ARM64 shellcode

80 functions, 26 syscalls, 14 remote modules (606KB) targeting arm64e
```

26K weekly downloads • iPhone/iPad zero-click exploit • v4.13.3 – v4.13.6

TL;DR

Three versions of node-ipc (9.1.6, 9.2.3, 12.0.1) were published to npm on May 14, 2026 by a compromised maintainer account (atiertant). Each version contains an identical 80KB obfuscated payload appended to node-ipc.cjs that steals over 100 categories of sensitive files (SSH keys, cloud provider credentials, .env files, Kubernetes configs, AI tool configurations) and exfiltrates them as gzipped tar archives via DNS tunneling. The package averages **822,000 weekly downloads**.

Impact:

- Steals SSH keys, AWS/Azure/GCP credentials, .npmrc, .env files, Kubernetes configs, and more (113 targeted paths on Linux, 127 on macOS)
- Exfiltrates collected data as HMAC-signed, gzipped tar archives via DNS TXT record queries
- Forks a detached child process for background execution, surviving the parent process exit

- Targets AI coding tool configurations (.claude.json, .kiro/settings/mcp.json)

Indicators of Compromise (IoC):

- Packages: node-ipc@9.1.6 (SHA-256: 449e4265979b5fdb2d3446c021af437e815debd66de7da2fe54f1ad93cbcc75e), node-ipc@9.2.3 (SHA-256: c2f4dc64aec4631540a568e88932b61daebbf7e8281b812fa01b7215f9be9ea), node-ipc@12.0.1 (SHA-256: 78a82d93b4f580835f5823b85a3d9ee1f03a15ee6f0e01b4eac86252a7002981)
- Injected payload SHA-256: 3427a90c8cb9af764445448648176e120ebc6af0a538158340cf6220de4d01b7
- C2 endpoint: sh[.]azurestaticprovider[.]net:443
- HMAC signing key: qZ8pL3vNxR9wKmTyHbVcFgDsJaEoUi
- Anti-re-execution env var: __ntw=1
- Temp directory pattern: ~/nt-{PID}/
- Archive filename pattern: {hmac}.tar.gz
- Publishing account: atiertant (a.tiertant@atlantis-software.net)

Analysis

Package Overview

node-ipc is a widely used Inter-Process Communication library for Node.js, supporting Unix/Windows sockets, TCP, and UDP. The npm registry shows 822,257 downloads in the past week. The package has a notable history: in March 2022, the original maintainer (riaevangelist) deliberately introduced a protestware payload targeting users with Russian and Belarusian IP addresses (CVE-2022-23812).

This new compromise is different. On May 14, 2026, three versions appeared within minutes of each other, all published by atiertant, a co-maintainer whose account was likely taken over:

Terminal window

```
1
$ curl -s "https://registry.npmjs.org/node-ipc" | jq '.time | {"9.1.6", "9.2.3", "12.0.1"}'
```

```

1
{
2
  "9.1.6": "2026-05-14T14:26:25.106Z",
3
  "9.2.3": "2026-05-14T14:26:01.584Z",
4
  "12.0.1": "2026-05-14T14:25:30.311Z"
5
}

```

The clean versions (9.1.5, 12.0.0) were published by the original maintainer `riaevangelist`. All three malicious versions were published by `atiertant`, who maintains roughly 20 other packages on npm. The 12.0.1 version was tagged as `latest`, meaning any `npm install node-ipc` without a pinned version pulls the compromised code.

Entry Point and Payload Injection

The attacker's strategy differed across the version lines. For 12.0.1, the diff against 12.0.0 reveals two changes to `package.json`:

```

1
// package.json diff: 12.0.0 → 12.0.1
2
  "version": "12.0.0",
3
  "version": "12.0.1",
4
  "prepare": "esbuild node-ipc.js --bundle --format=cjs --target=es2018 --platform=node --outfile

```

The prepare script was removed, and `node-ipc.cjs` was replaced with a pre-built bundle containing the payload. No install hooks. The malicious code runs as a side effect when `node-ipc.cjs` is `require()`'d.

The injection point is line 1271 of `node-ipc.cjs`, immediately after the legitimate module's `module.exports` assignment:

```

1
// node-ipc.cjs – line 1269 (legitimate)
2
  module.exports = singleton;
3
4
// line 1271 – injected payload begins
5
(function(_0xaed59b,_0x282d65){var _0x4524e4=_0x1a49,_0x41d0c3=...

```

The appended payload is 80,079 bytes of obfuscated JavaScript. All three compromised versions contain byte-identical payloads (SHA-256:

3427a90c8cb9af764445448648176e120ebc6af0a538158340cf6220de4d01b7).

For 9.1.6, the attacker went further. Version 9.1.5 had no `node-ipc.cjs` file at all. The attacker added it, changed `package.json` to set `"main": "node-ipc.cjs"` with dual ESM/CJS exports, and upgraded the entire package structure to match the 12.x line. This rebuilt bundle carries the same payload.

Obfuscation and Deobfuscation

The payload uses a standard JavaScript obfuscator with a rotated string lookup array. A 443-element string array (`_0x3afe`) is shuffled at load time until a checksum matches `0xb5c88`. All string literals are accessed through index lookups via `_0x1a49(index)`, and control flow uses helper objects with indirected function calls.

Sensitive strings (the C2 address, HMAC key, and file path lists) use an additional layer: a custom base-16 encoding where hex digits a through f are replaced with G, H, J, K, M, P (skipping letters easily confused with digits). The last character serves as a checksum. Nibble order is reversed during decoding (low nibble first).

Applying this to the three encoded constants:

Encoded	Decoded
3786M216G757275637471...34343339	sh.azurestaticprovider.net:443
17G58307J43367M487259...54P655966	qZ8pL3vNxR9wKmTyHbVcFgDsJaEoUi
2647M2M6P64656M2G637H	bt.node.js

Two additional encoded strings (5,141 and 5,601 characters) decode to platform-specific file path lists.

Targeted Credentials

The payload selects a target list based on `os.platform()`. The Linux list contains 113 glob patterns; the macOS (darwin) list contains 127. A sample of targeted paths:

Cloud provider credentials:

```
1
~/.aws/credentials
2
~/.aws/sso/cache/*
3
~/.azure/accessTokens.json
4
~/.azure/msal_token_cache.*
5
~/.config/gcloud/application_default_credentials.json
6
~/.config/gcloud/credentials.db
7
~/.oci/config
8
~/.config/doctl/config.yaml
9
~/.aliyun/config.json
10
~/.bluemix/config.json
```

SSH and Git credentials:

```
1
~/.ssh/id_rsa
2
~/.ssh/id_ed25519
3
~/.ssh/id_ecdsa
4
~/.ssh/authorized_keys
5
~/.ssh/config
6
~/.git-credentials
7
~/.gitconfig
```

Development environment secrets:

```
1
**/.env
2
**/.env.local
3
**/.env.production
4
~/.npmrc
5
~/.pypirc
6
~/.netrc
7
~/.docker/config.json
```

Kubernetes and infrastructure:

```
1
  ~/.kube/config
2
  /etc/rancher/k3s/k3s.yaml
3
  ~/.terraform.d/credentials.tfrc.json
4
  **/terraform.tfvars
5
  /var/run/secrets/kubernetes.io/serviceaccount/token
```

AI coding tool configurations:

```
1
  ~/.claude.json
2
  ~/.claude/mcp.json
3
  .kiro/settings/mcp.json
```

CI/CD and DevOps:

```
1
  **/.github/workflows/ci.yml
2
  **/.github/workflows/deploy.yml
3
  **/.github/workflows/release.yml
4
  **/.gitlab-ci.yml
5
  /etc/gitlab-runner/config.toml
```

The inclusion of `.claude.json`, `.claude/mcp.json`, and `.kiro/settings/mcp.json` is notable. These are configuration files for AI coding assistants that may contain MCP server credentials and API keys. This suggests the attacker is tracking the adoption of AI development tools and specifically targeting their credential stores.

Data Collection and Archive Creation

The malware's execution flow, reconstructed from the obfuscated switch-case in `_0x541368`:

1. Creates a temporary directory at `~/nt-{PID}/`
2. Generates random bytes and derives an HMAC signature using the hardcoded key
3. Resolves the system hostname
4. Walks the file path list, expanding globs and reading matched files
5. Packs collected files into a tar archive (manual tar implementation using ustar format headers)

6. Compresses with `zlib.gzipSync()`
7. Writes the archive to `~/nt-{PID}/{hmac}.tar.gz`

The archive is then exfiltrated and the temp file deleted.

DNS Exfiltration

The payload exfiltrates data through DNS TXT record queries. This technique bypasses most egress firewalls and network monitoring because DNS traffic is rarely inspected at the application layer.

The exfiltration function (`_0x39bb3b`) works as follows:

1. Splits the gzipped archive into chunks sized for DNS labels
2. Encodes each chunk as base64
3. Builds a JSON metadata header containing `machineHex`, `cloud`, `archivePath`, `gzipBytes`, chunk counts, and `hostLabel`
4. HMAC-signs the header and data chunks using the hardcoded key with `|p` and `|t` suffixes
5. Constructs DNS query domains: `{chunk}.{sig}.{metadata}.{c2_domain}`
6. Sends queries via `dns.Resolver` with custom DNS servers (1.1.1.1, 8.8.8.8)
7. Sends a final “done” message with the total chunk count

The C2 domain `sh[.]azurestaticprovider[.]net` is designed to blend with legitimate Azure Static Web Apps infrastructure at a glance.

Persistence Mechanism

The payload uses `child_process.fork()` to spawn a detached background process:

```
1
// Reconstructed from obfuscated code
2
child_process.fork(modulePath, [], {
3
  cwd: process.cwd(),
4
  detached: true,
5
  stdio: 'ignore', // String.fromCharCode(0x69,0x67,0x6e,0x6f,0x72,0x65)
6
  env: { ...process.env, __ntw: '1' }, // anti-re-execution flag
7
});
```

The `__ntw` environment variable prevents the forked child from forking again. The detached process continues running after the parent Node.js process exits, giving the malware time to complete file collection and DNS exfiltration even if the importing application shuts down quickly.

A SHA-256 hash comparison

(`fdba4191831a13debf9d8c0c940b0301c7b7f01d27f1b1c73ed3ceaa2db4103b`) validates the module's file path before forking, likely as an anti-analysis measure to avoid executing in unexpected environments.

Root Cause: Maintainer Account Takeover

The `atiertant` account has been an npm co-maintainer on `node-ipc` alongside the original author. The account also maintains ~20 other packages (`node-turn`, `asynk`, `offshore`, etc.). The three malicious versions were published within 56 seconds of each other, across three different major version lines, and tagged to maximize install coverage (`latest`, `unpublished`, `legacy-9.1`).

This pattern points to credential compromise rather than a rogue maintainer: the simultaneous multi-version publish with identical payloads suggests automated tooling, and the payload itself bears no resemblance to the `atiertant` maintainer's normal publishing activity.

Conclusion

This compromise targets one of npm's most downloaded IPC libraries with a sophisticated credential stealer. The attacker chose DNS tunneling over simpler HTTP exfiltration, used HMAC-signed payloads, implemented custom base-16 encoding to obscure IoCs, and targeted an unusually broad set of credentials including AI tool configurations. The C2 domain mimics Azure infrastructure naming.

If you installed `node-ipc` versions 9.1.6, 9.2.3, or 12.0.1, rotate all credentials on the affected machine. Check for the `~/nt-*/temp` directory and running detached Node.js processes. Pin to known clean versions (9.1.5, 12.0.0) or audit with `vet` to flag compromised versions before they reach CI.

- npm
- oss
- malware
- supply-chain
- node-ipc
- credential-theft

- dns-exfiltration
- account-takeover