Are the Bad Guys Already in Your Software Supply Chain?

(Spoiler Alert: Yes)

THREATCON1
2025-09-22

Paul Novarese <pvn@huntedlabs.io>

$ whoami

Paul Novarese
Hunted Labs
pvn@huntedlabs.io
Fediverse: @pvn@mas.to
Signal: pvn.99

Agenda

1. Building software is different now
2. Supply chain attacks are ascendant
3. Our defenses are weak
4. Maybe we can take some actions to make things slightly better?

My Biases

- This talk is mainly about application security
  - (as opposed to regulatory compliance, OS hardening, etc)
- My background is more Ops than Dev
- I empathize more with blue teams
- I mostly see through an ASPM lens (particularly SCA)
- My day job is soaked in cloud native woo woo
- I have spent most of my career working in open source

Graham Smith 3:19 PM

Yesterday

Ben at [redacted] just asked: "As an aside, what are your thoughts about the recent NPM attack?

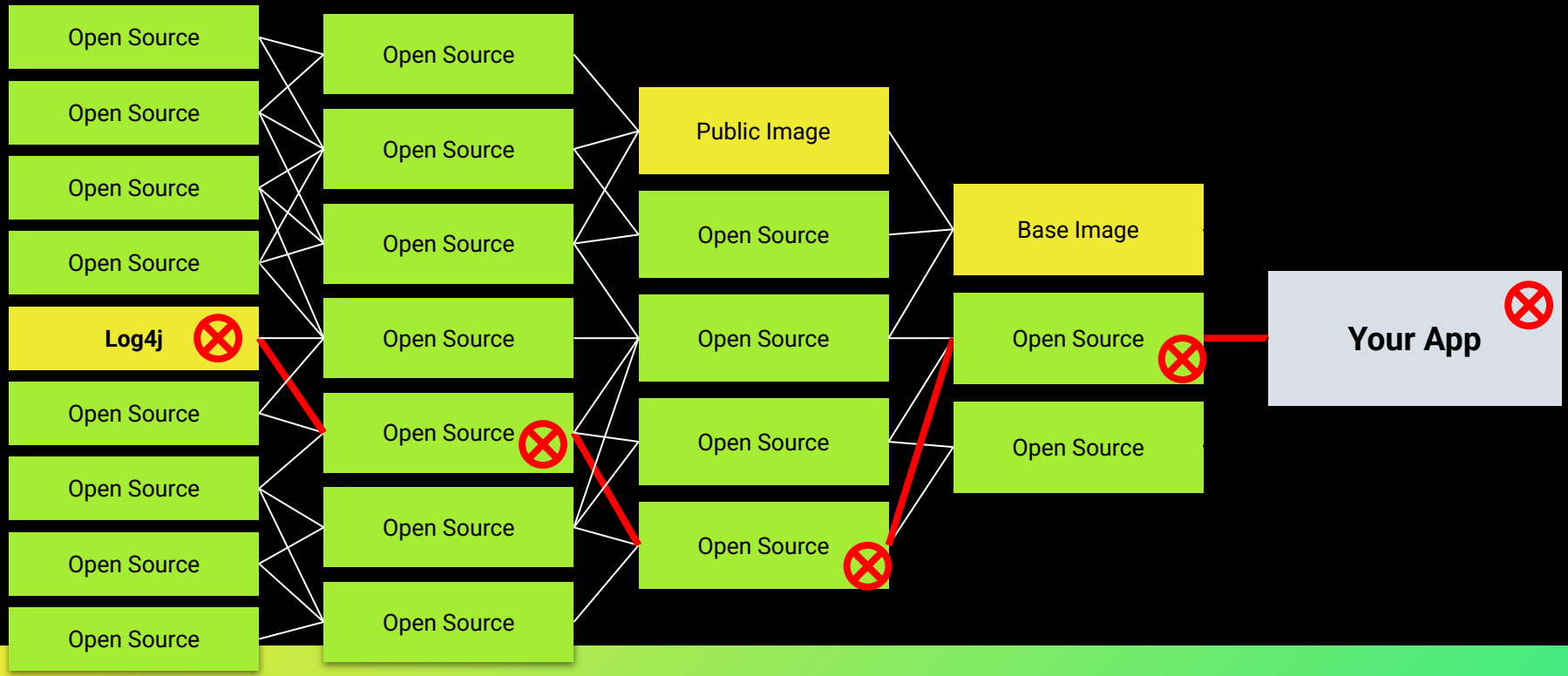
pvn 3:19 PM

hahah which one

Software is Different Now

How we build is foundational to the problem

The Software Supply Chain



Notes for The Software Supply Chain

- log4shell is a *vulnerability* in the supply chain (not a “supply chain attack”)
  - worst case scenario for the “old threat” (traditional vulnerabilities)
- This made “software supply chain” as a concept real for many for the first time
- First time a lot of people heard of SBOMs
- Not much changed, CVE advisory went out, people remediated and then went right back to consuming open source the way they were before
- This diagram also shows us something very important
  - *software is built differently now than 20 years ago*
- Software is built on software that is built on software

Open Source is Bigger Than You Can Imagine

Josh Bressers https://www.youtube.com/watch?v=SEnB9Riiazs

Notes for Open Source is Bigger Than You Can Imagine

- NPM introduced 2010 (and many other language package managers around there)
- 50 million releases just in npm
- 5 million packages
- 8000 new npm releases PER DAY
- Proposal to do security reviews on 10,000 open source projects per year
- Even if you exclude “junk” with 1 version it’s still tens of millions
- Clearly this is doomed

The Jia Tan Reverse Funnel Plan

XZ

Fedora

Suse

Debian

Docker Images

RHEL

SLES

Ubuntu

etc

Base Image

Open Source

Open Source

Open Source

Open Source

Open Source

Open Source

Your Organization

Enterprise

Government Agency

Nonprofit

Enterprise

Enterprise

NGO

Government Agency

Enterprise

Notes for The Jia Tan Reverse Funnel Plan

- XZ very different than log4shell
  - the particulars here aren't important
  - the basic idea is that liblzma/systemd/sshd confluence was exploited
  - this is an actual ATTACK unlike log4shell
- More importantly, it's almost certainly state-sponsored

Software Supply Chain Attacks

Why are the SO HOT right now?

Paul Novarese · You

Software Supply Chain Security/InfoSec/OPSEC

3mo ·

I'm going to start actually collecting data but it feels like most enterprise software development projects have way more risk from deliberate malware software supply chain attacks than from "traditional" vulnerabilities. Even if this isn't currently the case, it will be very soon since the velocities aren't even in the same neighborhood.

10

8 comments · 1 repost

Notes for Previous Slide

- Yes, CVEs are increasing
- *But, malware supply chain attacks are increasing faster*
- CVEs are becoming more noisy, less actionable
- Most of the scores are bogus
- *KEV is one of the few strong signals remaining*
- Sonatype just published the data, 230% increase in Q1 YoY
  https://www.sonatype.com/blog/open-source-malware-index-q1-2025

Notes for Previous Slide

- This is the before times
- Those switches are for literally toggling in boot code bit by bit
- Not a lot of dependencies in this universe (direct OR transitive)
- There is (basically) no (software) supply chain here

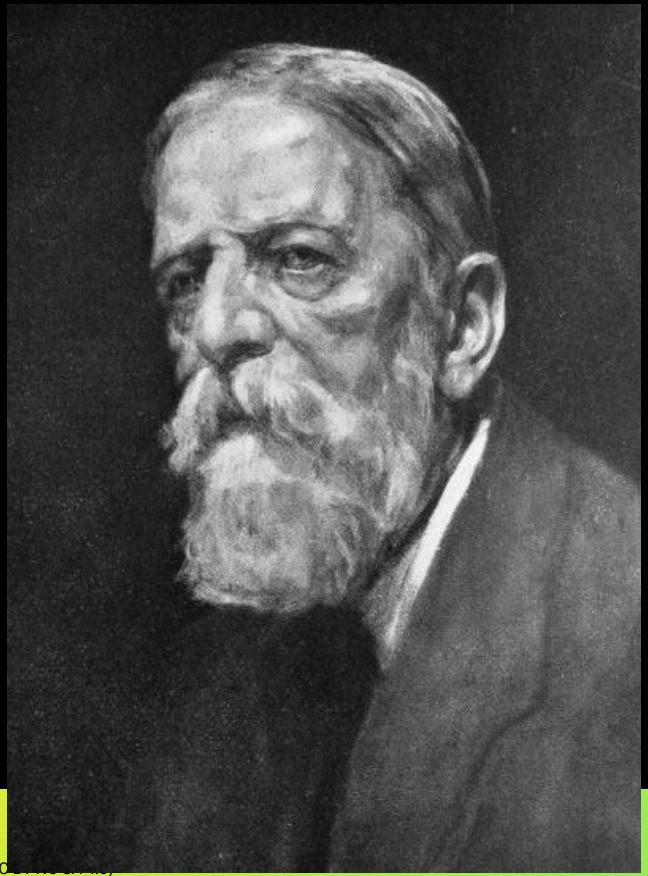
Open Source is (Still) Bigger Than You Can Imagine

Josh Bressers https://www.youtube.com/watch?v=SEnB9Riiazs

Notes for Open Source is (Still) Bigger Than You Can Imagine

- This graph is NPM specifically
- But it looks the same for any subset of open source you can look at
- It started exponential growth when automatic package managers arrived
- *This explosion has given attackers a huge opportunity*
- But this alone doesn't explain why they're moving to supply chain attacks instead of vulnerability exploitation

Economic Considerations



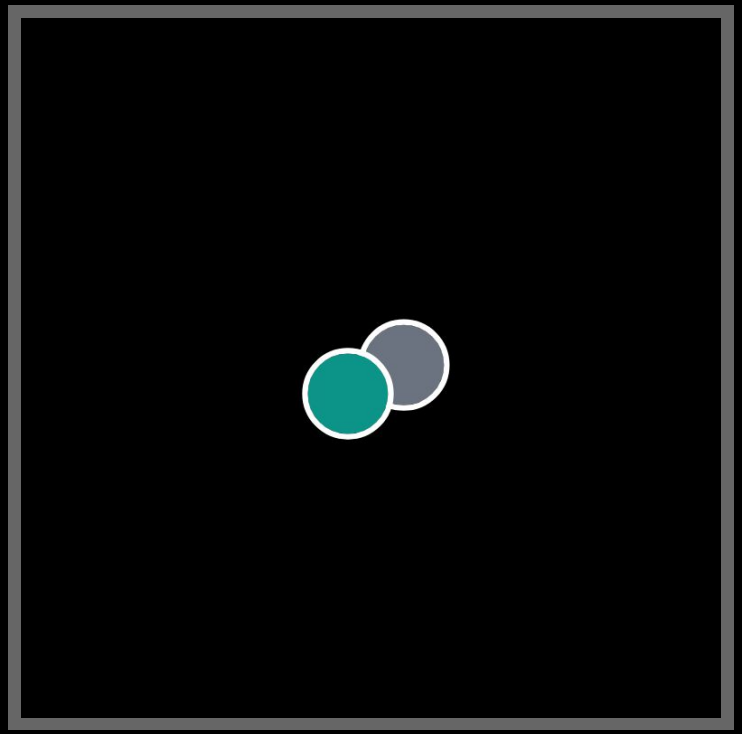
Notes for Economic Considerations

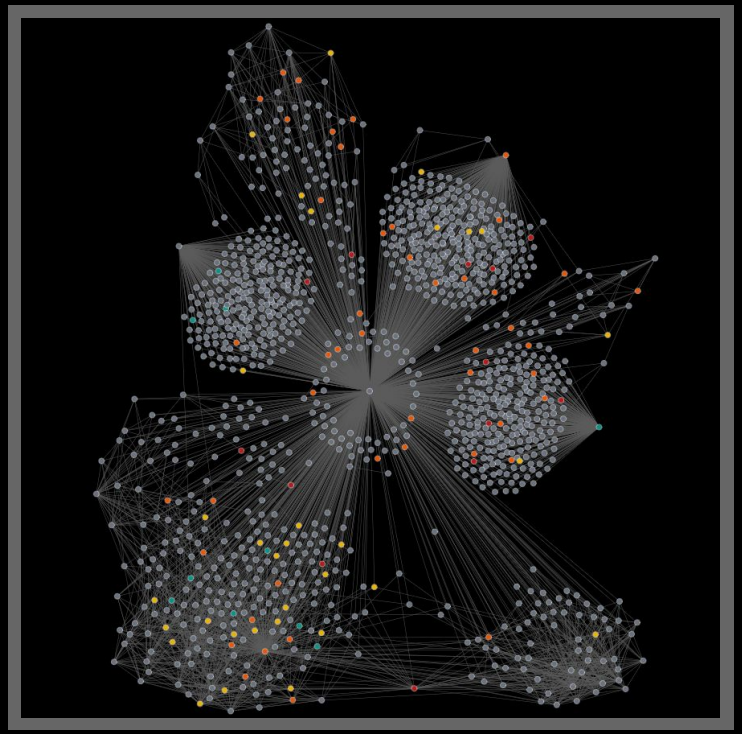
- (Left) Friedrich von Wieser - coined “opportunity cost”
- (Right) Jules Dupuit - invented cost/benefit analysis

- Zero days are hard
- Expensive to develop (code quality is increasing)
- The payoff has been shrinking
- You’re limited in what you can accomplish based on the nature of the vulnerability

- Malware injection is relatively easy
- It can scale, you can do basically anything
- So why are we just now seeing these take off???

Economic Considerations (cont.)





Notes for Economic Considerations (cont.)

- In an old app you dont have a lot of options
- In a modern app you have millions
- The door is open

So this combination:

1) Economics of vulnerability exploitation are getting worse, fast
2) Bang/Buck for supply chain attacks is through the roof
3) There’s a huge new frontier of open source packages that are just waiting for someone to hijack

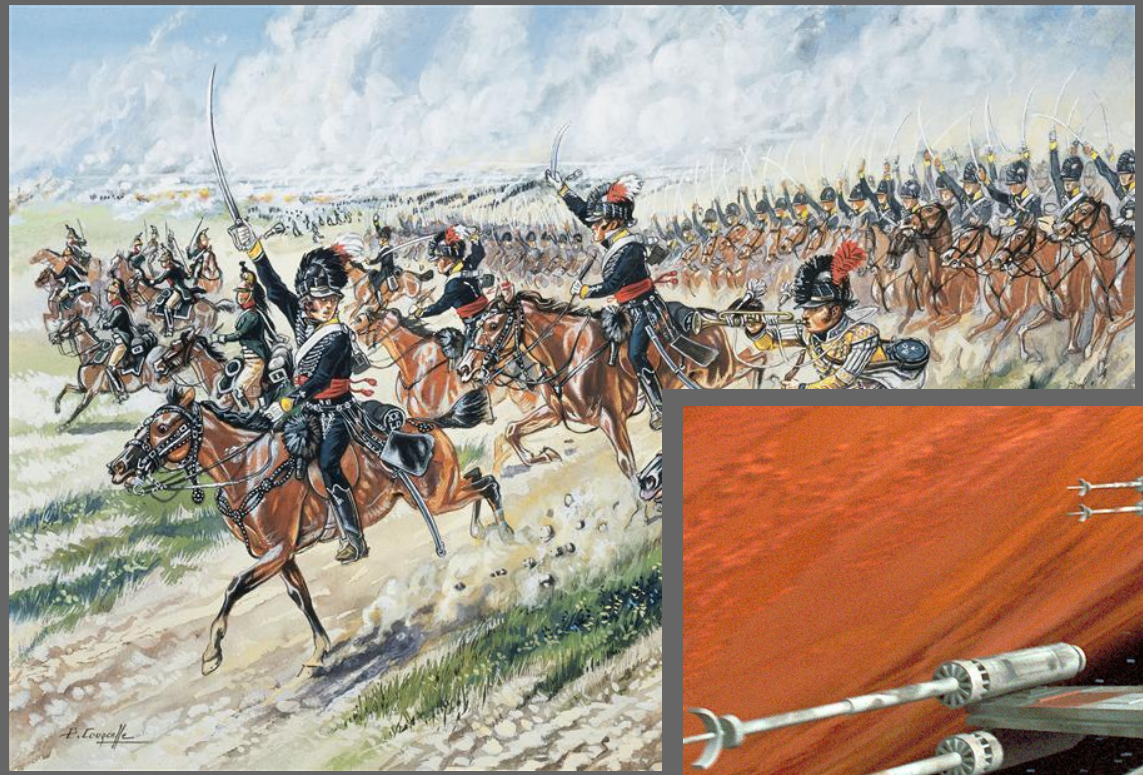
Suddenly we’re living in a new nightmare

The Defenses are Weak

We are not Equipped to Deal with This

Notes for Tactics 1

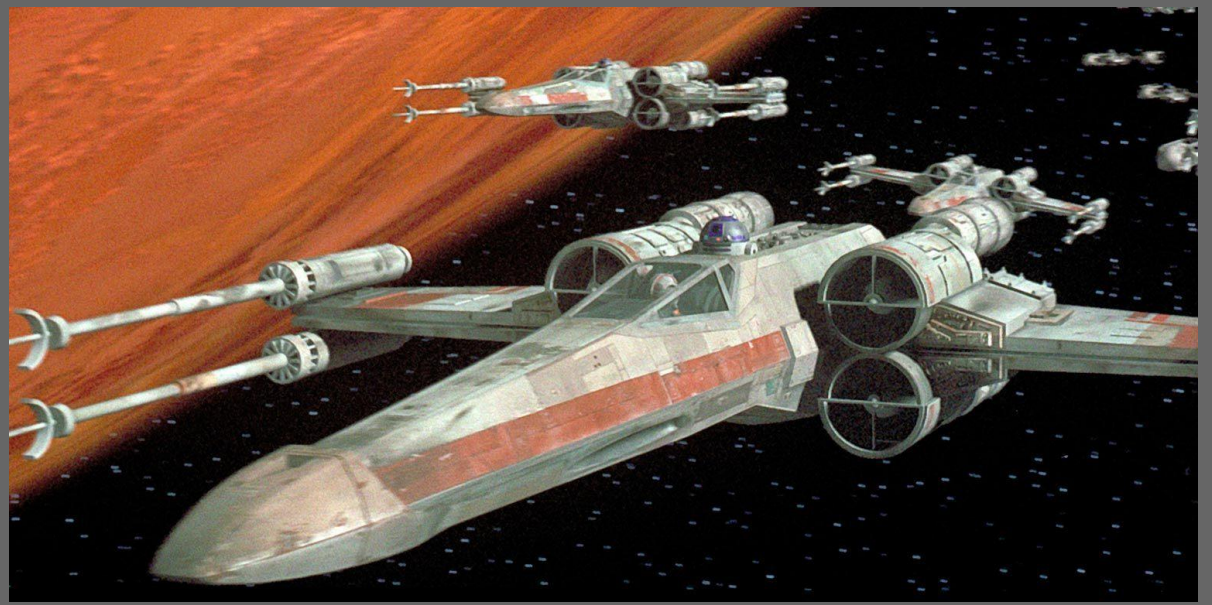
- A bit about tactics
- Backdoors are a thing, but they’re not like this anymore
- This movie has baked certain (incorrect) expectations into people’s brains



What defenders are equipped for

vs.

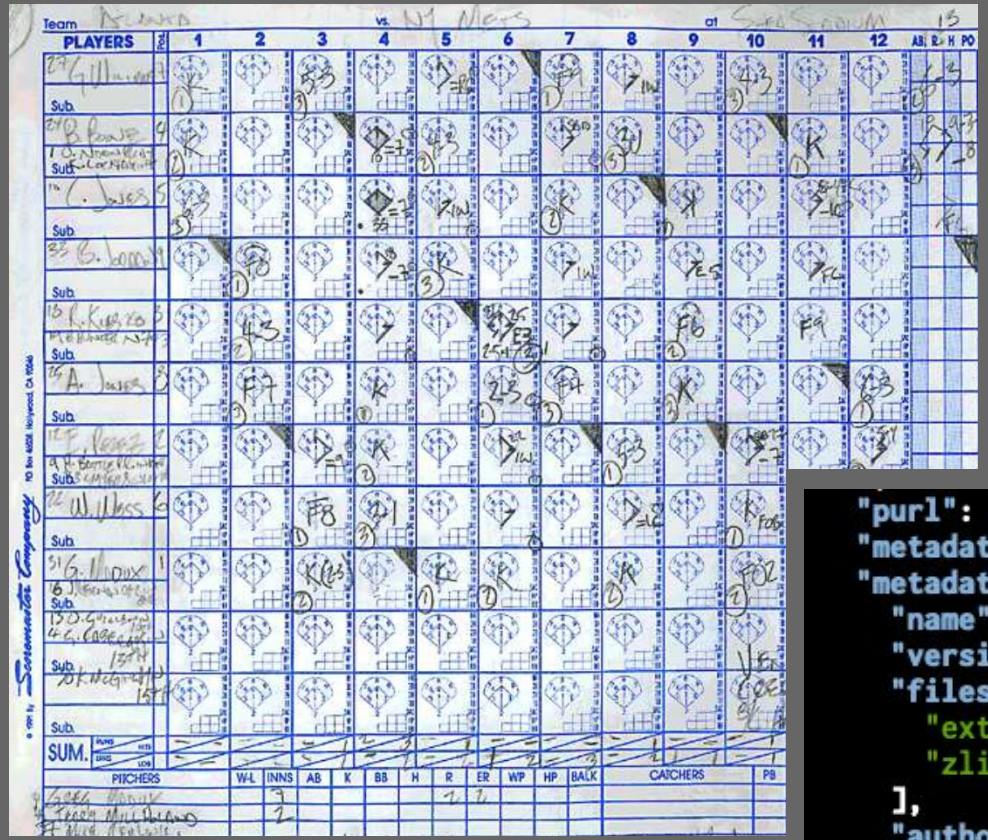
What they’re actually up against



Notes for Tactics 2

- Speed = success
- The scenario in your head of planting backdoors and then waiting is wrong
- Most criminal/opportunistic attacks are over in ~2 hours
- XZ attack unfolded slower in absolute terms
  - But still involved a strong push to speed up acceptance testing
  - on the timescale of the ecosystem it was occurring in, still moved at a massively accelerated scale compared to normal operations
- Defenders' tools and techniques are stuck in the last century
- Any defense relying on advisories (or threat databases in general) is not going to help against these sorts of attacks
  - These are inherently slow processes involving collection, analysis, judgement, dissemination
  - all of which takes time

Tools



Nutrition Facts

Serving Size 6 rolls (85g)
Servings Per Container 2.5

Amount Per Serving	
Calories 210	Calories from Fat 80
	% Daily Value*
Total Fat 9g	**14%**
Saturated Fat 2g	**11%**
Trans Fat 1.5g	
Cholesterol 10mg	**3%**
Sodium 390mg	**16%**
Total Carbohydrate 25g	**8%**

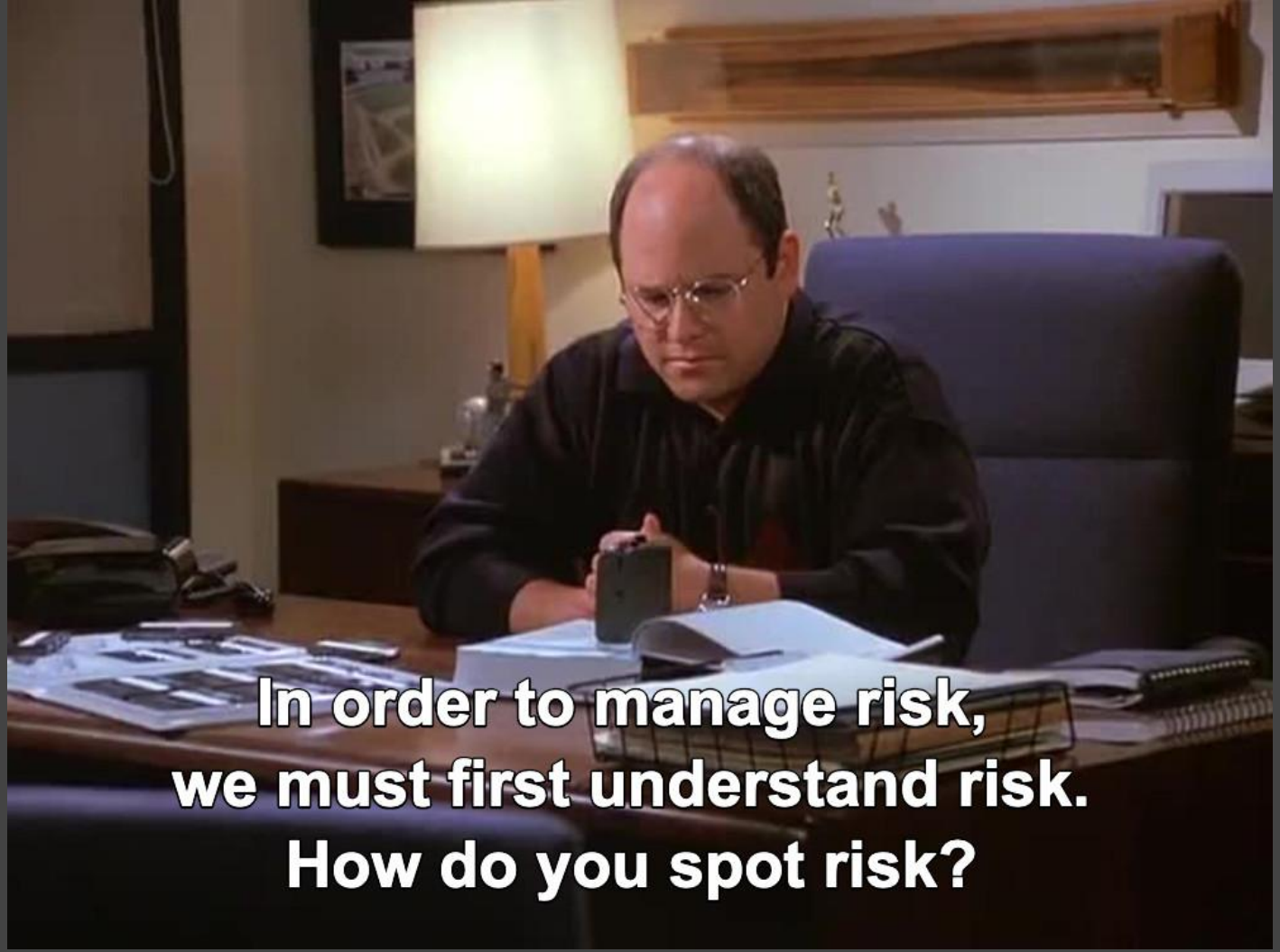
```
"purl": "pkg:gem/zlib@2.0.0",
"metadataType": "GemMetadata",
"metadata": {
  "name": "zlib",
  "version": "2.0.0",
  "files": [
    "ext/zlib/extconf.rb",
    "zlib.so"
  ],
  "authors": [
    "Yukihiro Matsumoto",
    "UENO Katsuhiro"
  ],
  "homepage": "https://github.com/ruby/zlib"
```

Notes for Tools

- OpenSSF scorecard
- Github Insights, project health, is the project vibrant,
- who are the contributors (are they even real people)
- SBOMs
  - more than just CPEs/vuln matching
  - *the map to open source insights*
- My dog will eat anything off the sidewalk
  - *We don’t have to eat out of dumpsters*

How do we Find the Bad Guys?

Where to look?



Notes for Risk Management

- We gotta figure out what's going on before we can look for the warning signs
- We need to know *who they are* before we can figure out *who they are*

typical bitcoin enjoyer

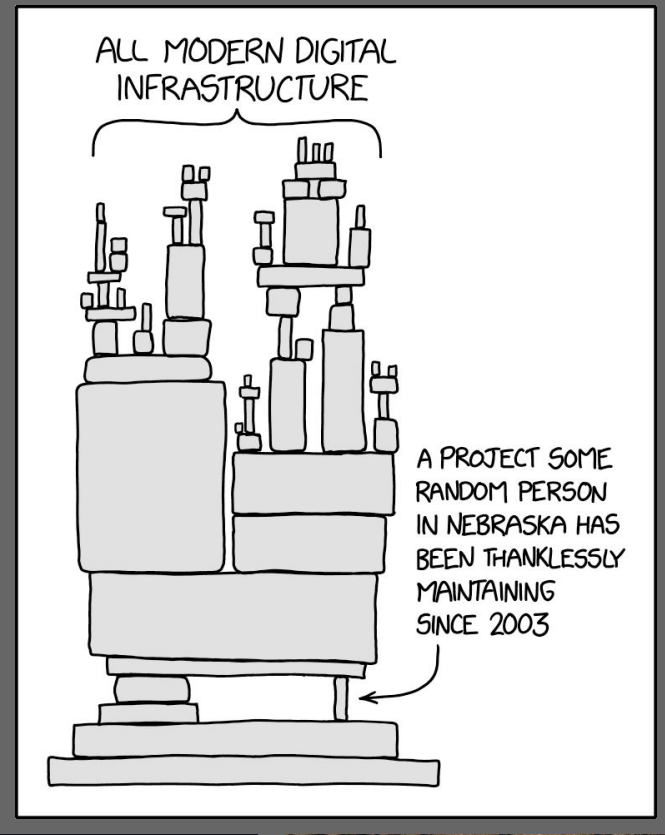
AVERAGE

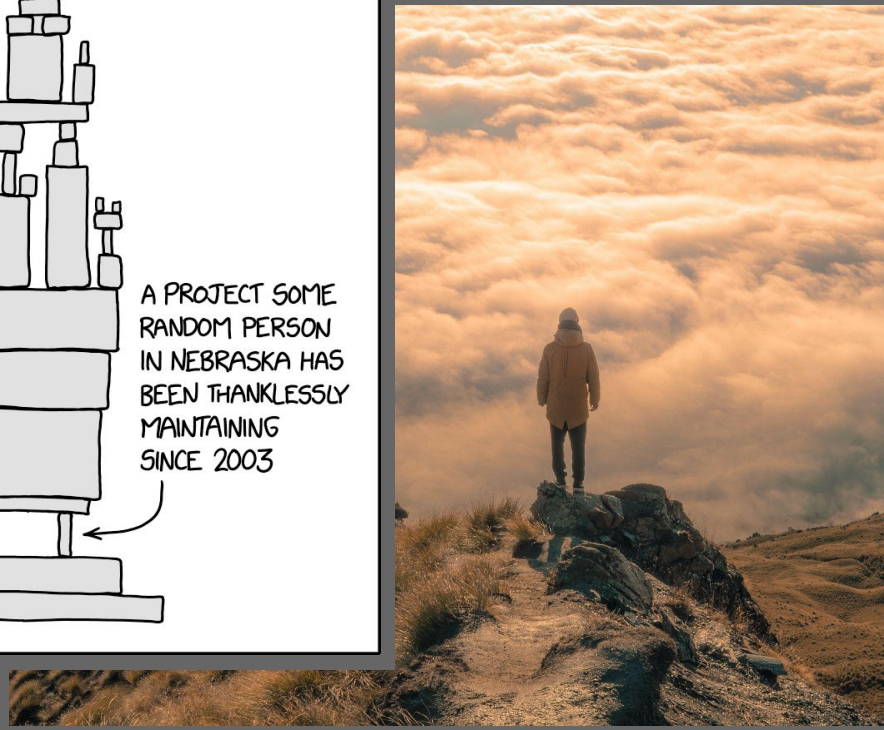
stuxnet enthusiast

Notes for Weirdos

- Basically two types of supply chain attacks in the wild
  - Steal bitcoin (ransomware with less steps) (criminals + DPRK)
  - State actors doing extremely sppok stuff

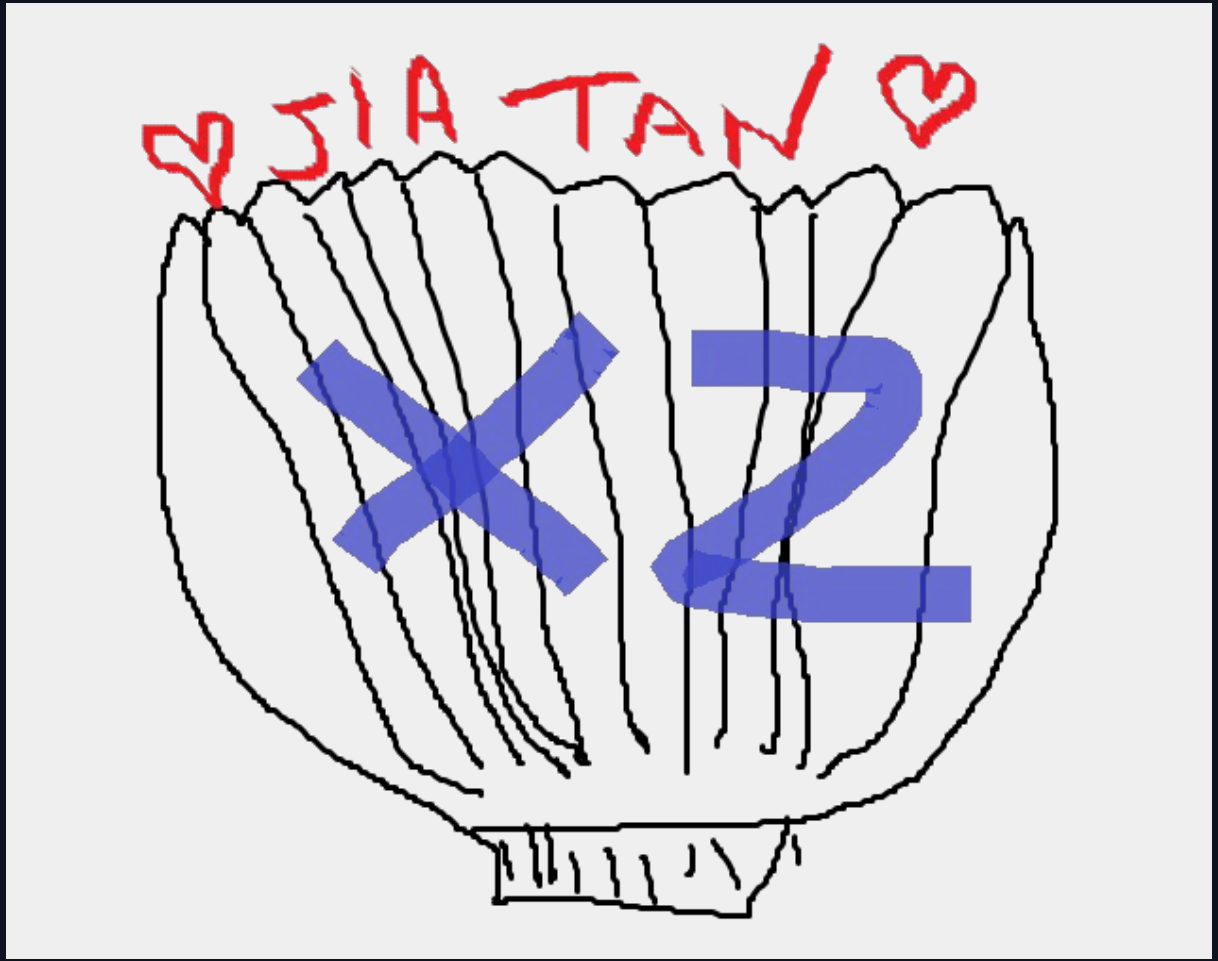
Targets





Notes for Targets

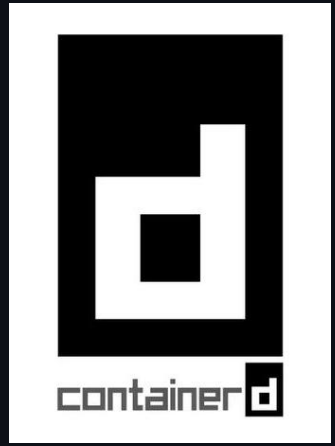
- Criminals are looking easy hits
  - projects with poor hygiene
  - solo absent maintainers
  - bad openssf scorecards
- Long-con state actors are looking for critical projects
  - overworked maintainers are a bonus
  - But projects that are TOO critical are often bad targets
- There are a LOT of projects in both buckets
- These are generalities and not hard-and-fast rules
- And let’s be very clear about the difference between contributors and maintainers



Notes for XZ

- Xz: extremely critical, ubiquitous
  - Turned out to be even more critical than anyone realized
- Extremely low profile
  - basically nobody directly working on it was paying attention
- Solo, overworked, burned out maintainer

```
# containerd project maintainers
#
# See GOVERNANCE.md for committer versus reviewer roles
#
# COMMITTERS
# GitHub ID, Name, Email address,GPG fingerprint
"AkihiroSuda","Akihiro Suda","akihiro.suda.cz@hco.ntt.co.jp","C020 EA87 6CE4 E06C 7AB9 5AEF 4952 4C6F 9F63 8F1A"
"dmcgowan","Derek McGowan","derek@mcgstyle.net","8C7A 111C 2110 5794 B0E8 A27B F58C 5D0A 4405 ACDB"
"estesp","Phil Estes","estesp@gmail.com","71AE 6DCD DFF8 C2D1 DDCA EEC9 FE25 9812 6B19 6A38"
"mikebrow","Mike Brown","brownwm@us.ibm.com",""
"fuweid","Fu Wei","fuweid89@gmail.com",""
"mxpv","Maksym Pavlenko","pavlenko.maksym@gmail.com",""
"dims","Davanum Srinivas","davanum@gmail.com","A67E 5FD8 80EB 089F 2317 7967 80D8 3A79 6103 BF59"
"kzys","Kazuyoshi Kato","kaz@fly.io",""
"samuelkarp","Samuel Karp","me@samuelkarp.com","0A4B DF41 8E8D ECB8 7F3E 9E14 AAA3 FE8A 831F C087,910C 2860 8D33 DDE6 89C0 3290 997C 5A3C D316 7CB5"
"kiashok","Kirtana Ashok","kirtana.ashok@gmail.com",""
#
# REVIEWERS
# GitHub ID, Name, Email address, GPG fingerprint
"jterry75","Justin Terry","jlterry@amazon.com",""
"cpuguy83","Brian Goff","cpuguy83@gmail.com",""
"thajeztah","Sebastiaan van Stijn","github@gone.nl","DE9A E9A2 F211 75DC B4F0 E564 7669 8F39 D527 CE8C"
"ktock","Kohei Tokunaga","ktokunaga.mail@gmail.com",""
"dcantah","Daniel Canter","danny@dcantah.dev",""
"MikeZappa87","Michael Zappa","Michael.Zappa@gmail.com",""
"klihub","Krisztian Litkey","krisztian.litkey@intel.com",""
"Iceber","Cai Wei","wei.cai-nat@daocloud.io",""
"laurazard","Laura Brehm","laurabrehm@hey.com",""
"henry118","Henry Wang","henwang@amazon.com",""
"ruiwen-zhao","Ruiwen Zhao","zhaorw93@gmail.com",""
"akhilerm","Akhil Mohan","akhilerm@gmail.com",""
"corhere","Cory Snider","csnider@mirantis.com",""
"austinvazquez","Austin Vazquez","austin.vazquez.dev@gmail.com",""
"djdongjin","Jin Dong","djdongjin95@gmail.com",""
"chrishenzie","Chris Henzie","chrishenzie@gmail.com",""
```



Notes for containerd

- Containerd is a critical project
- But this project has a whole org chart of maintainers
- Trying to slip something in here is going to be extremely difficult
- Bureaucracy sometimes is advantageous!

anchore / syft

Type / to search

Code | Issues 401 | Pull requests 40 | Actions | Projects | Wiki | Security | Insights

- Pulse
- Contributors
- Community Standards
- Commits
- Code frequency
- Dependency graph
- Network
- Forks
- Actions Usage Metrics
- Actions Performance Metrics

April 10, 2025 – April 17, 2025

Period: 1 week

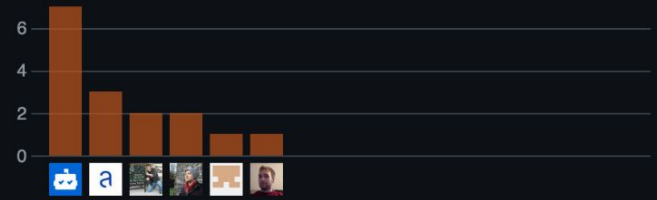
Overview

17 Active pull requests

7 Active issues

13	4	2	5
Merged pull requests	Open pull requests	Closed issues	New issues

Excluding merges, **6 authors** have pushed **13 commits** to main and **13 commits** to all branches. On main, **13 files** have changed and there have been **188 additions** and **43 deletions**.



13 Pull requests merged by **5** people

Notes for Insights

This is PROACTIVE (the better advisory data, scoring etc is about reactive improvements)

This is (currently) largely a manual process (it's getting a lot easier)

But Evaluating project health isn't directly about safety, it's about tracking all of those deps in the iceberg,

What happens when it hits the fan?

Are the projects you're depending on healthy, will you be able to work with them?

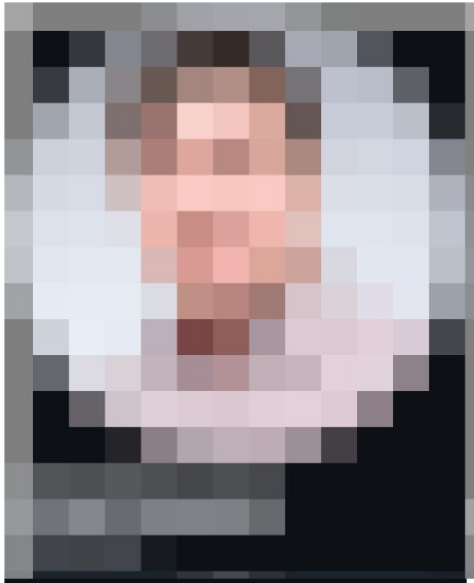
Also, things like openssf scorecard are just snapshots, it's also important to watch this stuff over time, that's how you really find the deviations from "normal"

This is extremely hard!

You vet your employees and your physical suppliers

Open source software runs in the exact same environment as your in-house code

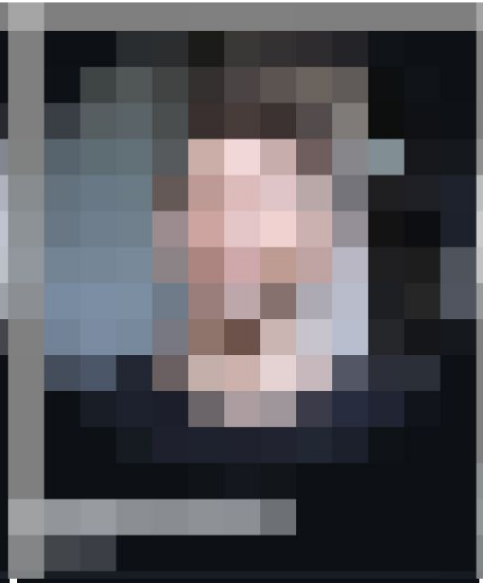
Why wouldn't you vet open source developers/maintainers/contributors



Follow

12 followers · **2** following

Moscow



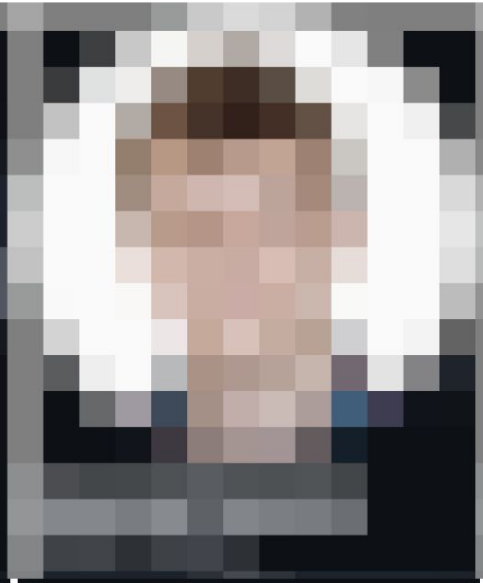
Follow

Technical manager at [redacted] Services

206 followers · **0** following

[redacted]

Moscow



Follow

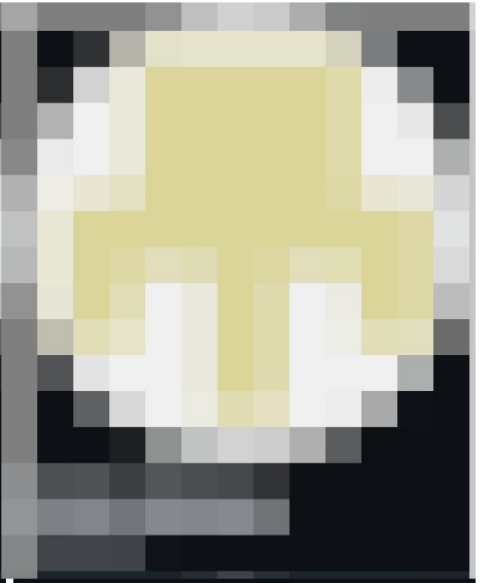
Head of Development Service at [redacted]

46 followers · **25** following

[redacted]

Moscow, Russia

[redacted]



Follow

10 followers · **12** following

Moscow

Notes for What We Found

- This is a project we're investigating, not calling out these particular guys but it was a noteworthy example
  - These guys control a pretty critical project used by tons of stuff, they all work for the same firm that is sanctioned by the US government, and there appear to be no outside maintainers
  - Their code contributions in this project and others have been fantastic and useful
  - Never had a CVE reported
  - Maybe these guys will never do anything nefarious
  - The project itself is in the corporate organization of a sanctioned company, and these maintainers could be swapped out
- A LOT of these guys are out in the open
- It's almost like they're "farming" credibility by being good open source participants (cf. Jia Tan)
- What should we be looking for? Contributors? Maintainers?
- FSB, CCP/PLA, DPRK, Iran all have different methods
- Solo maintainers, poor OpenSSF scorecard/hygine, projects that suddenly add a new very active maintainer
- **Changes in activity level**

Threat Reports:

Easyjson: https://huntedlabs.com/the-russian-open-source-project-that-we-cant-live-without/
Fast-Glob: https://huntedlabs.com/popping-fast-globs-hood/

chalk TS

5.6.2 • Public • Published 10 days ago

Readme | Code Beta | 0 Dependencies | 129883 Dependents | 43 Versions

Install

```
> npm i chalk
```

Repository

github.com/chalk/chalk

lk/chalk#readme

d this package

Releases 32

v5.6.2 Latest

last week

+ 31 releases

Notes for A Simple Example

- Microsoft owns both npm and github
- As we mentioned, a lot of attacks involve compromising a maintainer's npm account
- It's extremely rare for BOTH npm and github accounts to be compromised together
- So we see this sort of version mismatch pretty frequently
- Should be easy to sanity check this
- This obviously wouldn't catch everything but it would catch a lot of stuff

Open Source is not a Vendor

This

Not this

Notes for Open Source is not a Vendor

- *Open Source is (largely) a RESOURCE, not a VENDOR*
  - (this is different at the OS level (mostly))
- A supplier means you transfer responsibility to someone in exchange for money
- Getting libraries from Red Hat = supplier
- Getting stuff from npm - not a supplier
- A lot of critical plumbing is maintained by unpaid guys
  - They have day jobs, take vacations, etc.

Yes, the planet got destroyed by malware. But for a beautiful moment in time we created a magical developer experience.

Notes for What Can We Do?

- Github makes things worse because they want things to be extremely frictionless for devs
  - Friction is a big deal BUT there has to be SOME weight on securityPin dependencies whenever possible
  - Pinning isn't always pinning! (e.g. github tags are NOT immutable)
  - I used to think pinning was really bad, but it's basically imperative now
- You're much more susceptible if you're a devops practitioner or cloud native
- **Use the language and package manager security features**
- Fixing transitive dependencies is extremely difficult
- **You can't patch faster, you can't keep up, nobody can**
- Malware is increasing faster than vulnerabilities
- **Scan your builds and produce SBOMs**
- Don't eat out of dumpsters
- Take advantage of open source insights (expert mode)
- It's very difficult to stop long-con state actors
-

HUNTED LABS

Notes for Verified Open Source Contributor

- A lot of people think forcing registration or verification of open source contributor identities would prevent a lot of this
- On balance it would probably make things worse
- A lot of state actors are operating on GitHub under their real names already
- Any attempt to force registration would have huge backlash
- Lots of projects would abandon GitHub and become harder to track/monitor

HUNTED LABS