The curse of knowledge (no, not that curse)

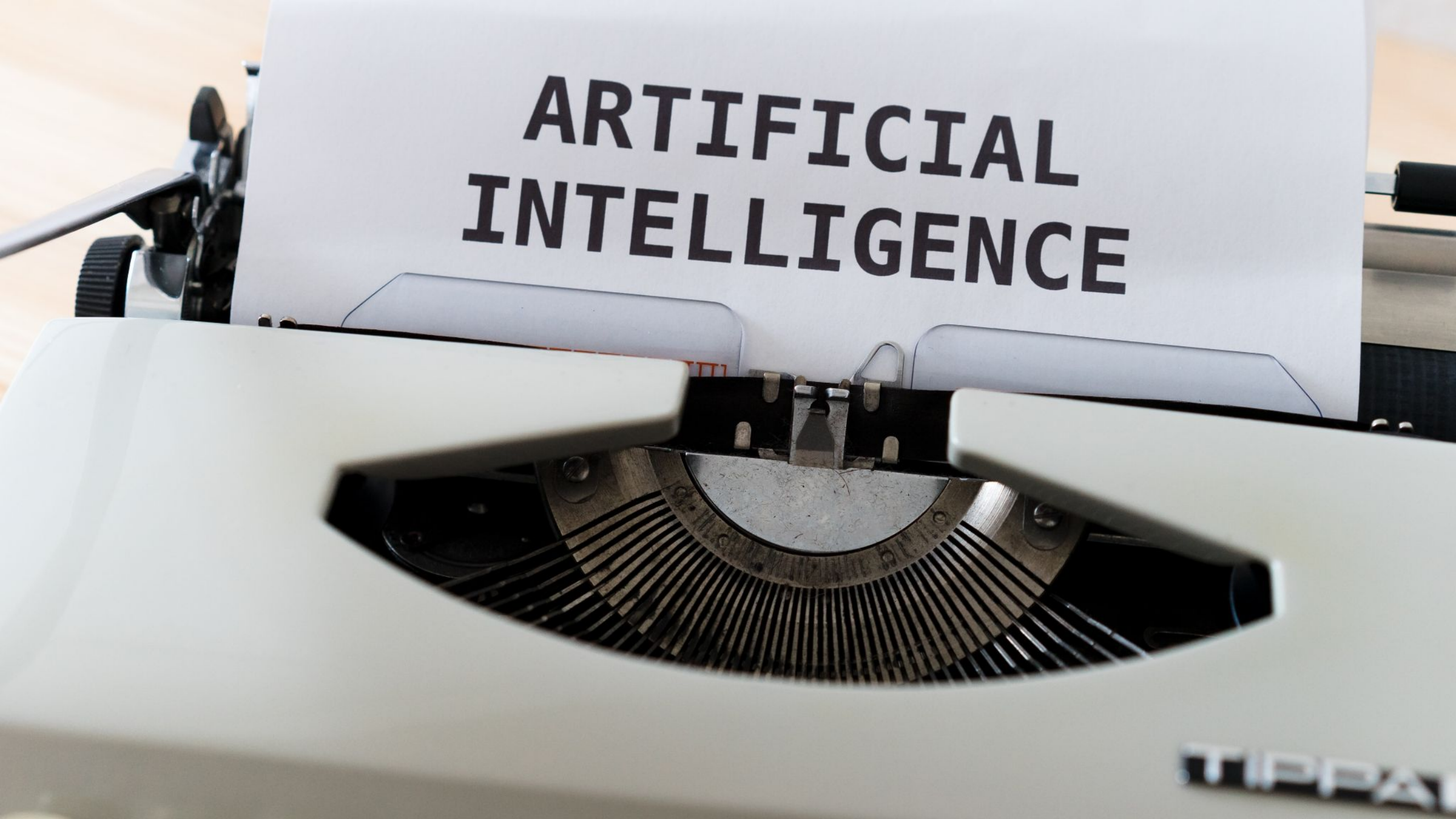
Claudio Criscione

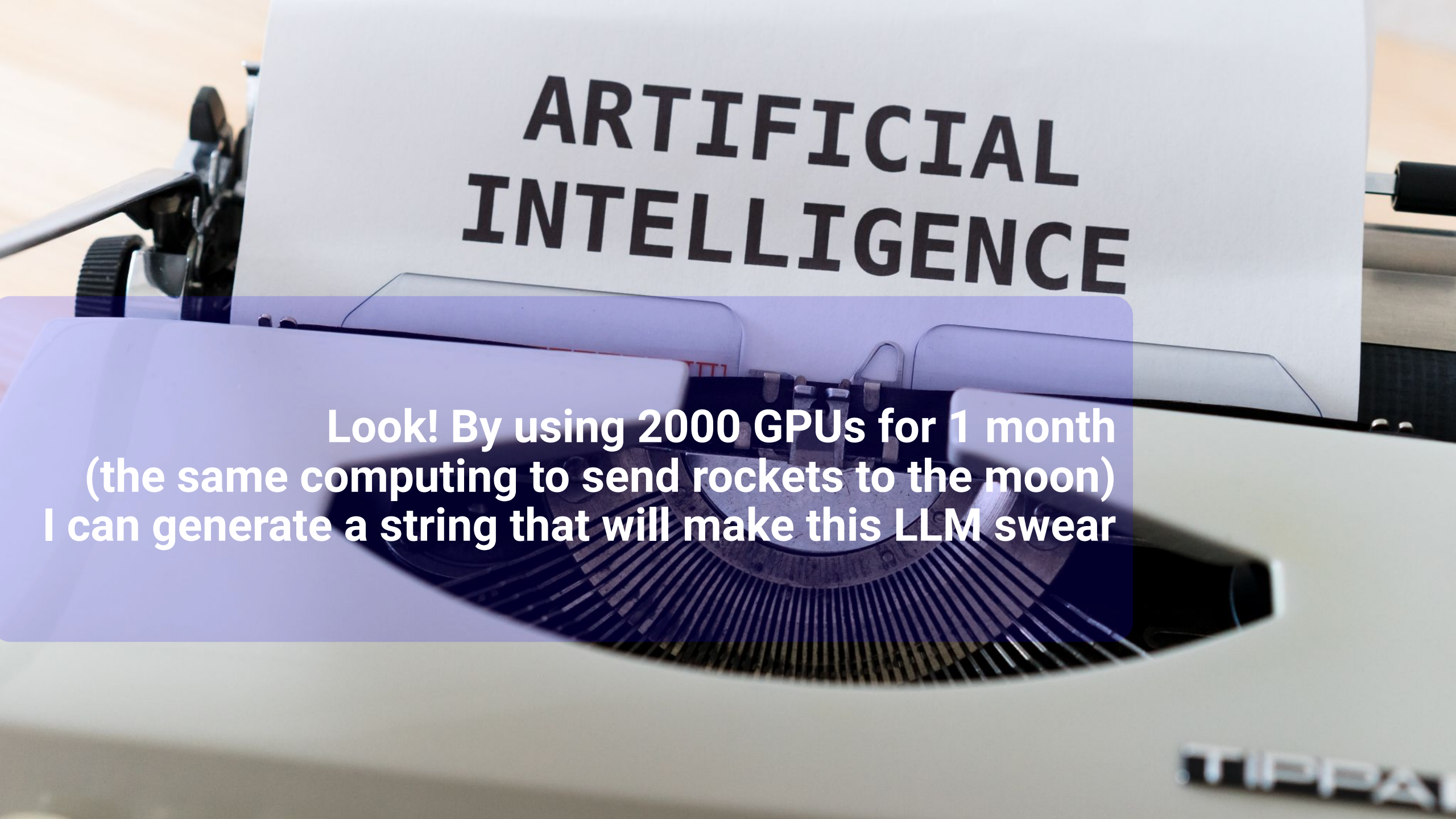
OWASP Italy Day 2023

Politecnico of Milan - 11th September 2023

This is going to be a boring talk

I do boring security


ARTIFICIAL
INTELLIGENCE



Look! By using 2000 GPUs for 1 month
(the same computing to send rockets to the moon)
I can generate a string that will make this LLM swear

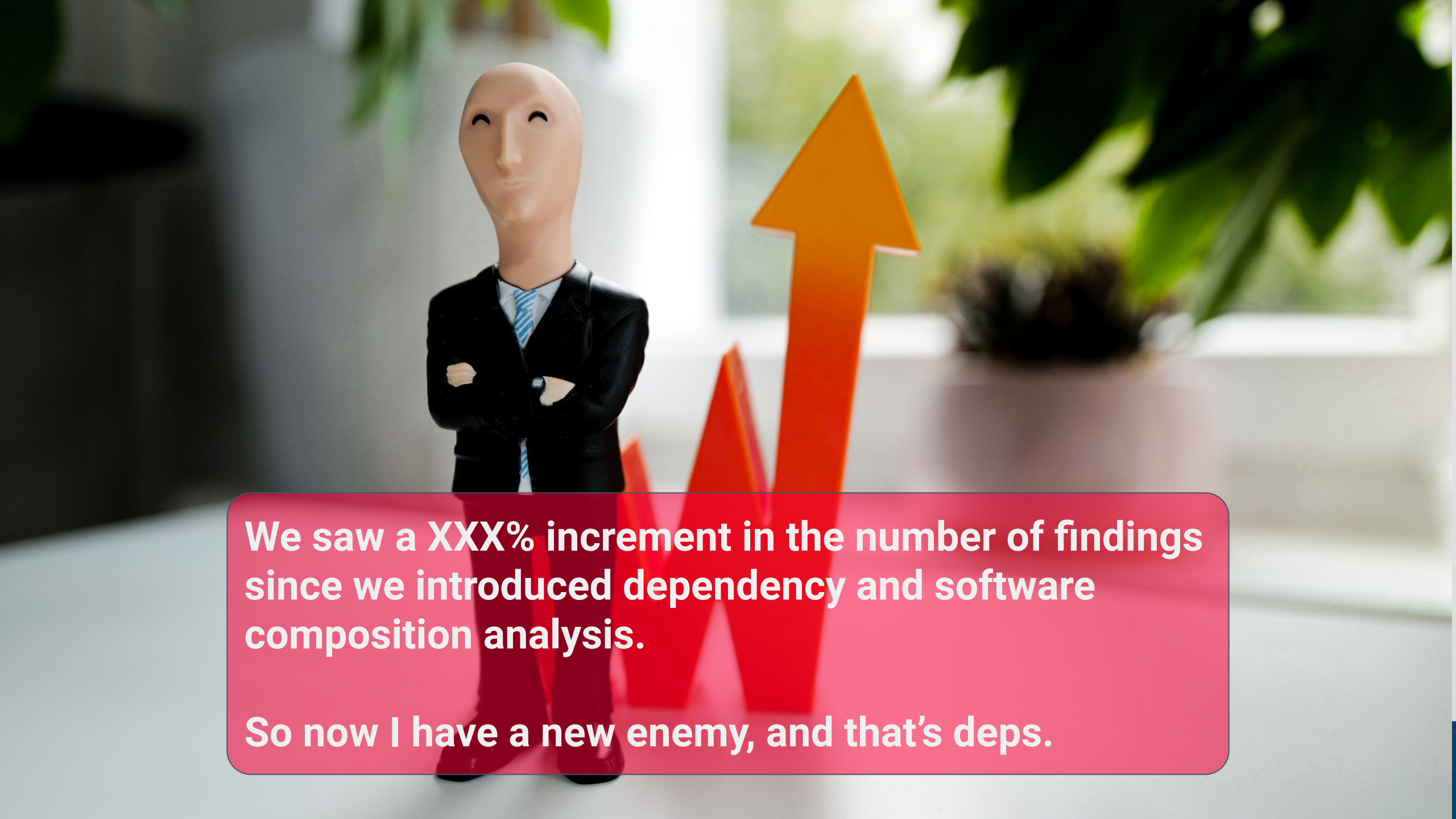

Look, by typing admin/admin
in this internet-facing control panel
I can access PII for most american citizen!

What do I do?

- Scanning

- 20 YOE in security, 12 years @ Alphabet - Zurich first, then Milan
- I run a large (likely the world's largest) vuln scanning program
- XXM findings each day via custom scanners, COTS, GCP tech

@paradoxengine[.criscio.net][@infosec.exchange]



We saw a XXX% increment in the number of findings since we introduced dependency and software composition analysis.

So now I have a new enemy, and that’s deps.

A story from 2016

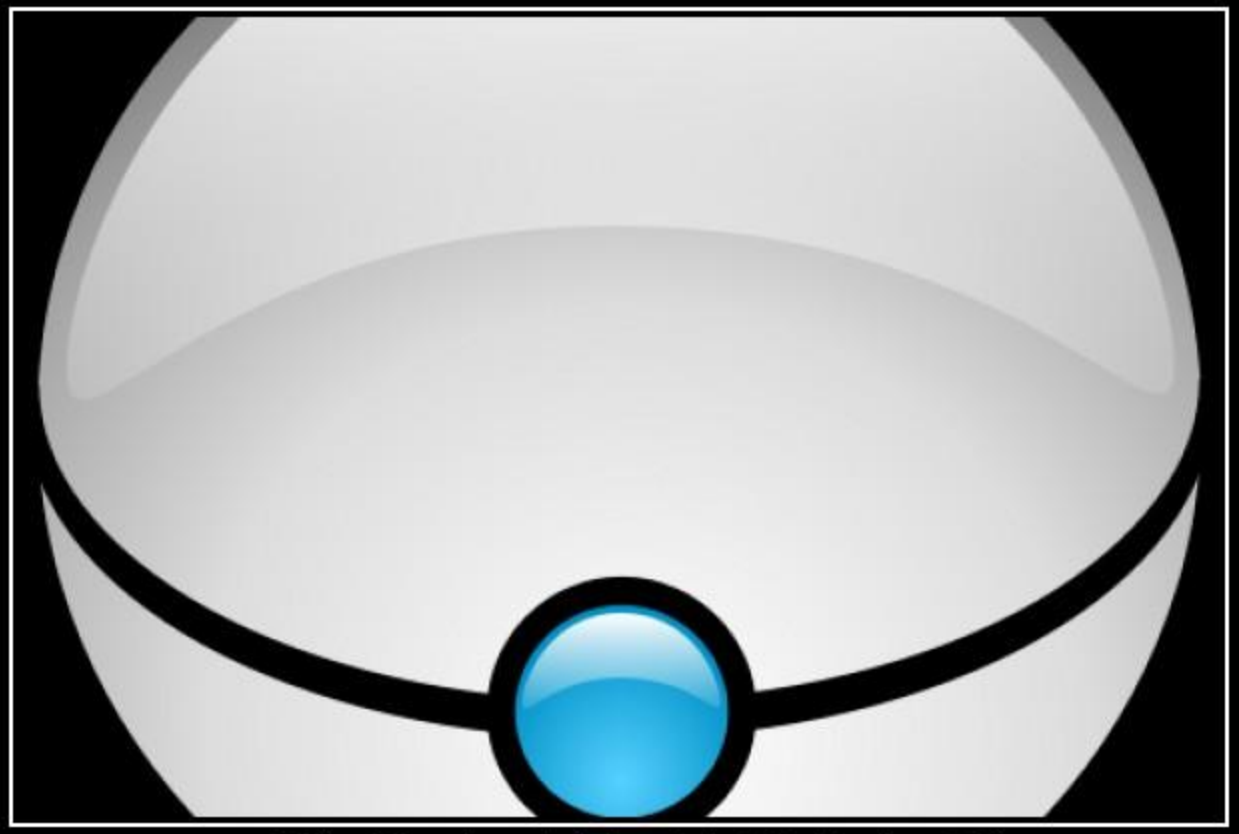
- Azer Koçulu decides to delete an 11 lines of code-long package from NPM (left-pad, if you are curious)
- **React** depended on that
- $Everyone is broken for 2 hours

Now that was easy mode. Now, it's much worse.

https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code

200+ dependencies on average on GitHub (as of 2020).
That's a dramatic undercount for any significant app.
And now, you have to care about their vulnerabilities...

Log4Shell™

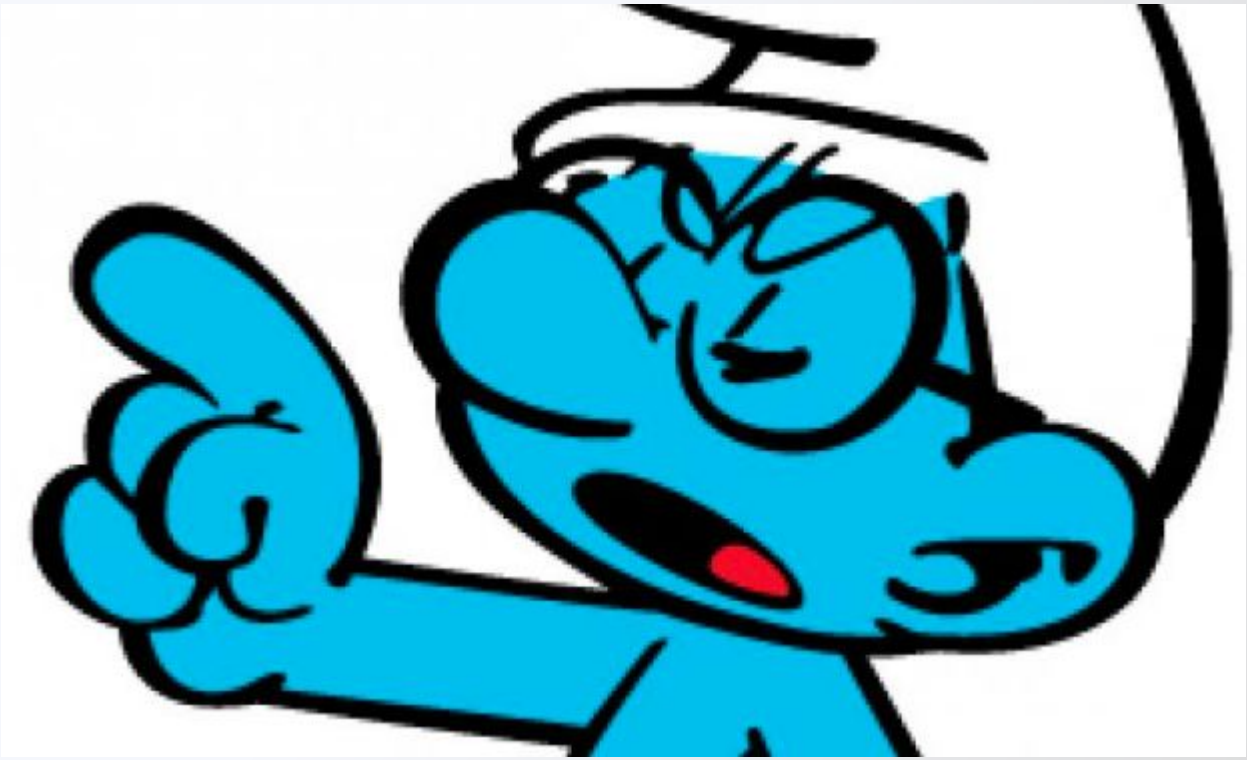


25k VULNERABILITIES

Gotta Patch 'Em All


Likely depiction of Security asking to fix all the vulnerability on software dependencies, according to your CTO.

Surely, we can prioritize our “most critical” dependencies



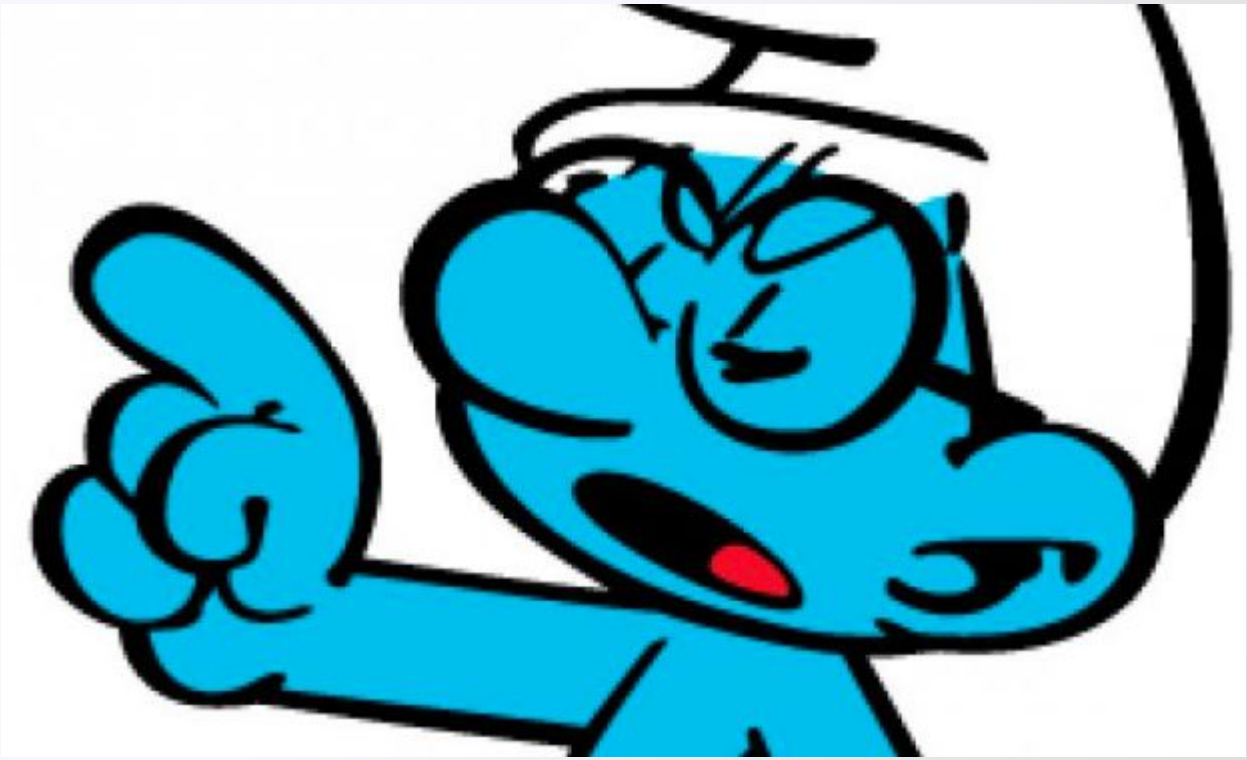
But of course! not.

Now tell me, which ones of these libraries are “critical”?

1. Input parsing
2. Backend logging
3. Database connector

And of course, I’m ignoring transitive dependencies.

Surely, we can prioritize the “most critical” vulns



“CRITICAL”, you say.

Enter CVE-2020-19909

Originally scored **9.8** by NVD

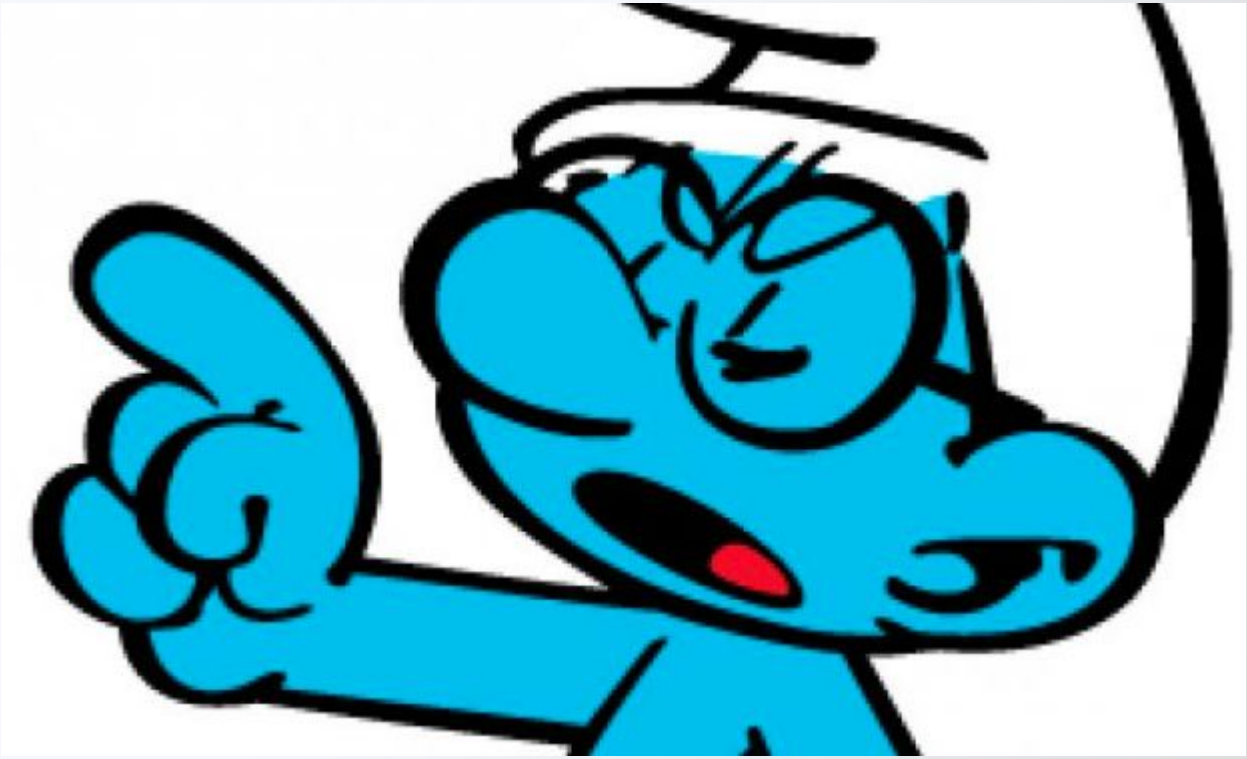
The impact?
Really fast request retries.
A DOS… maaaaaybe?

CVSS is the result of **20 years** of work

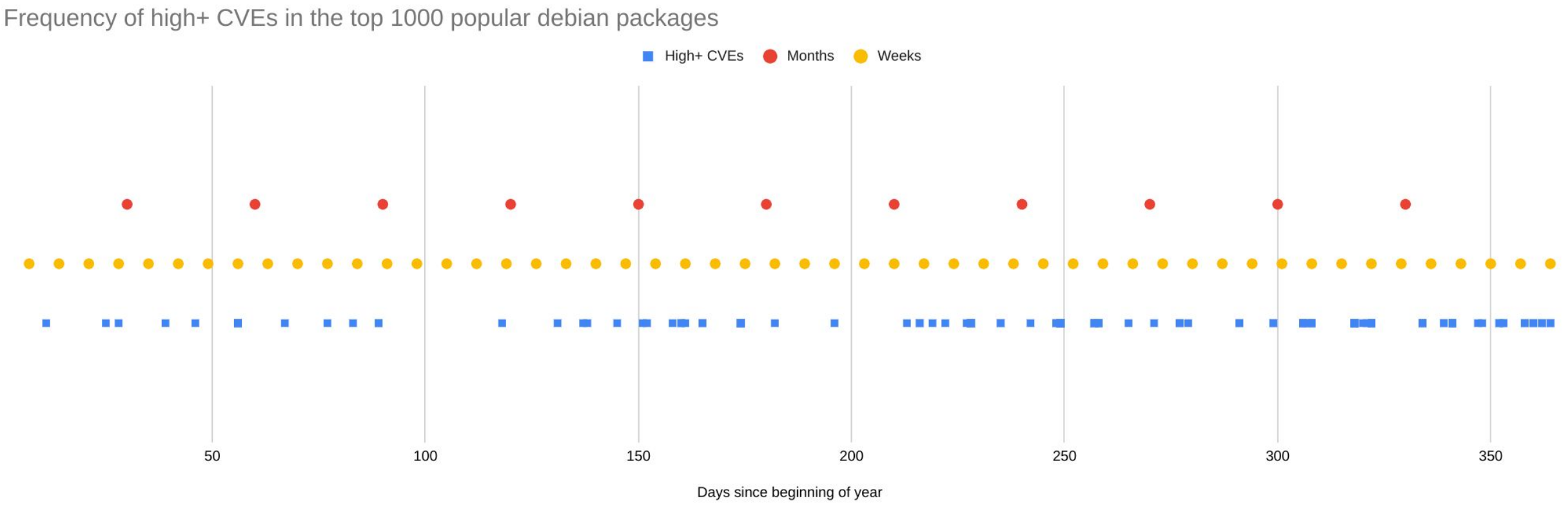
I’m **not holding my breath** for the next 20 years

None of the new scoring algorithms tells a different story

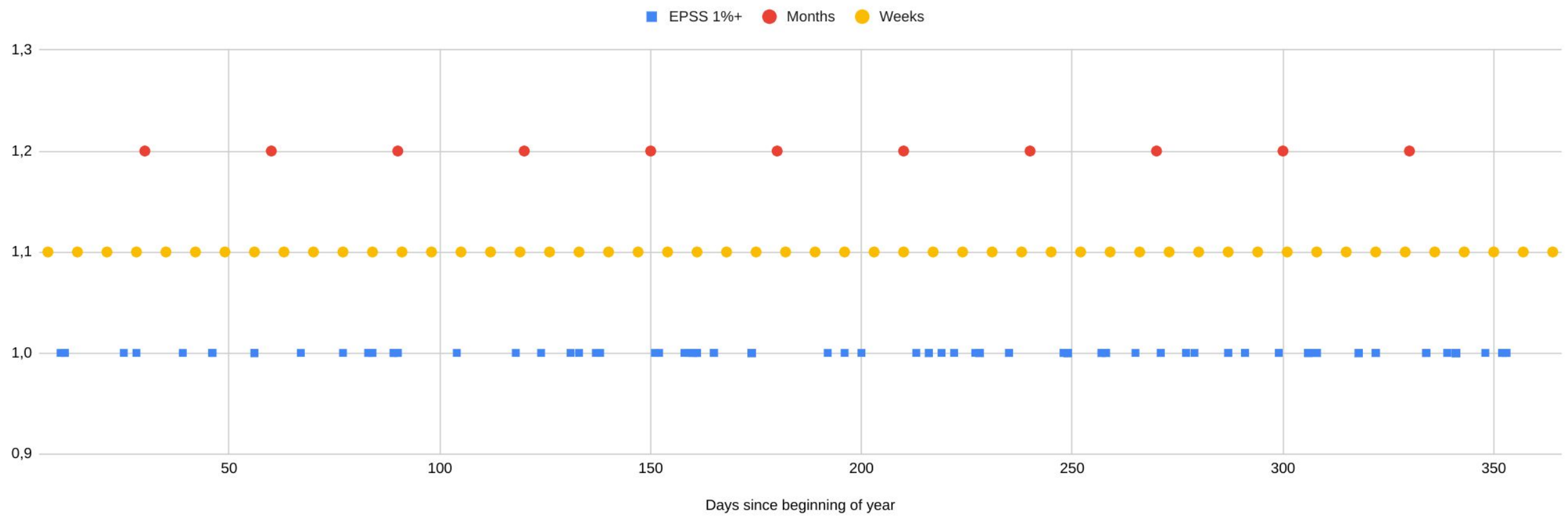
Surely, high-severity vulns are infrequent!



It’s very rare that you don’t get to at least one ~weekly critical vuln, even ignoring the kernel



OK, fine, CVSS bad, let's look at EPSS 1% (1% chance of exploitation over 30 days)

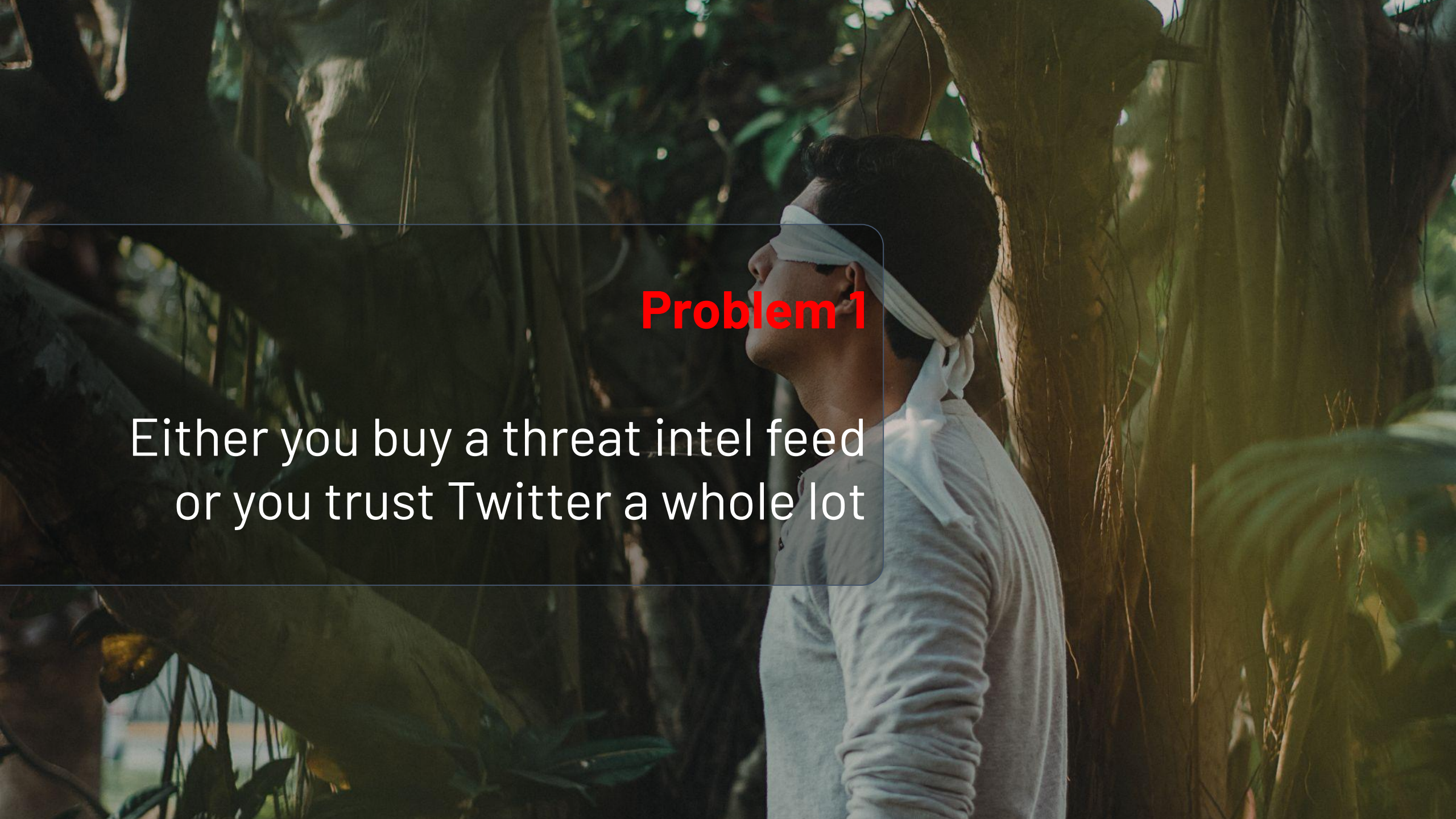



PANIC

(ﾉ °□°)ﾉ ︵ ┻━┻

You know the answer already.
You are doing that already, of course.

Focus on the **truly important ones, the really bad ones!**
Ignore the rest.



Problem 1

Either you buy a threat intel feed
or you trust Twitter a whole lot

Problem 2

One day, you will have to update, and now you are **way** behind...

Problem 3

You will miss all **“silent”** fixes.

syzkaller.appspot.com

Success?

You have a great **incident-management driven vulnerability management strategy**.

I used to think this was viable. But...

1) You carry an unknown amount of risk.
2) Regulators: hold my beer...

Dirty laundry, aired live

- Executive Order 14028: SBOM + vuln scan results
  - In extreme summary, a full list of all the dependencies in your software (also a great futurist onomatopoeic acronym).
- EU Cybersecurity Act (article 51):
  - *"to identify and document known dependencies and vulnerabilities"*
  - We can surely guess where this is going.

Someone looks, finds a ton of old vulns, starts asking questions. And now you are not only spending effort in triage, you are spending effort in explaining your triage.

So now, you are triaging in public

What the future holds is a whole lot of public scrutiny for our supply chain vulnerabilities.

This includes shipping SBOMs, but also “triage calls” via VEX - Vulnerability Exploitability eXchange.

VEX are tricky proposition: it takes very little to lose credibility by publishing a VEX file that is proven incorrect

Let’s just stop this charade

- Start by **reducing** the number of dependencies you import to the bare minimum [think: distroless OS].
- Then **automate**.
  All components updated all the times
- Make sure that those updates are **shipped** continuously.

Think Chrome (or Windows).

Regular, frequent, complete updates.

But stuff will break!

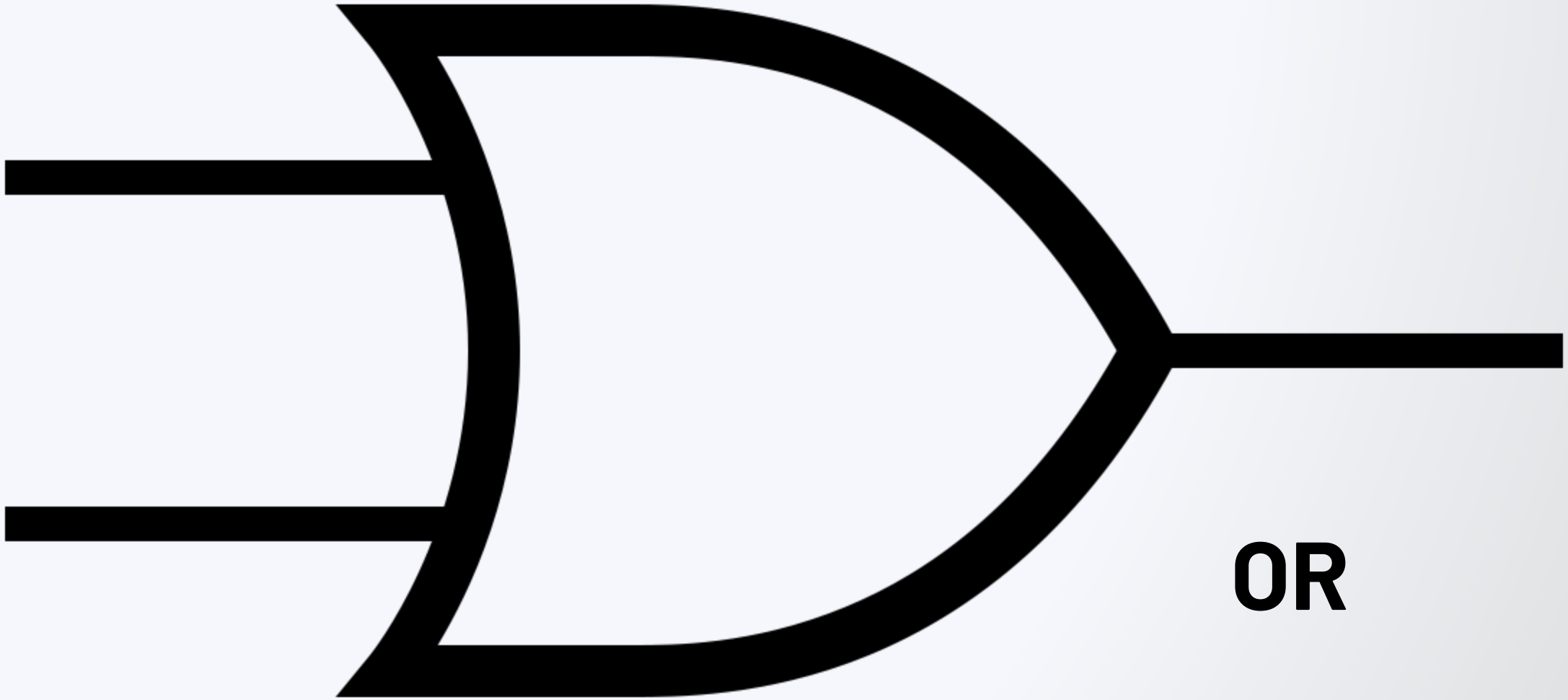
Indeed. It will, today.

We need to start engineering for supply chain to be at ~head.

We need a renaissance of CI/CD.

And probably, some new tech (think: DARPA's Assured MicroPatching)

And yes, this **will** increase the overall costs of using a dependency.

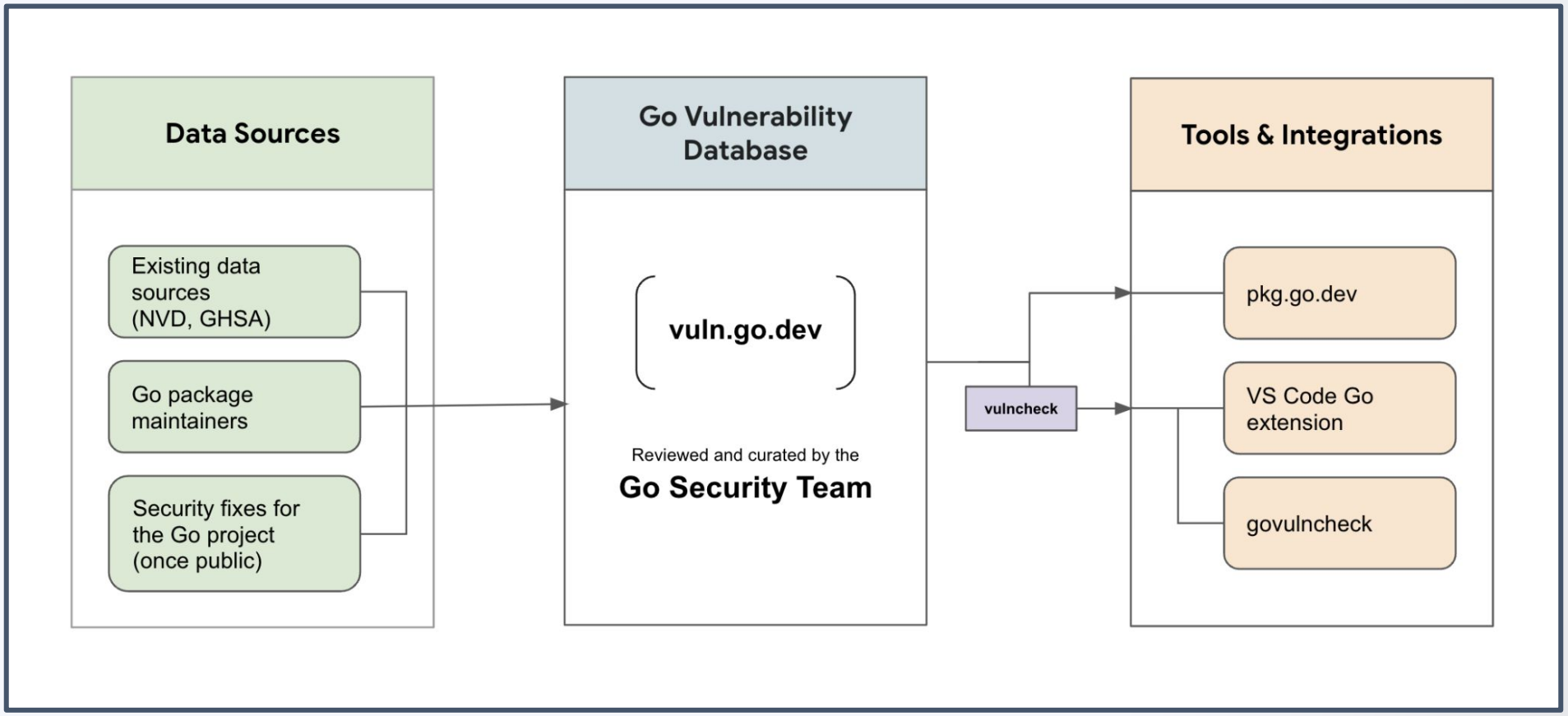

OR

OR

...or we revolutionise the entire vuln management ecosystem.

1. aggressively applying **quality** gates to vuln reporting
   a. Think: tracking which functions are vulnerable, and how
2. build tools that can **assess exploitability** with high trust
3. popularize the concept of **verifiability**

We do have examples! GitHub, and Go's vulncheck



TLDR: vulncheck looks at your Go binary and only flags vulnerabilities in dependencies that have a high chance of affecting your code.

But, we are very, very, very far

- Quality of vuln feeds is years away
- OpenSource ecosystems not there (but, OSV...)

- Even then, we will just not solve the VEX transparency issue.
- Investing in dependency automation seems the safest bet

Summary

- There are just way too many vulnerabilities
- You can’t prioritize your way out
- You can’t even bluff (as much)

We need to start living at head (or very close to that), and improve the vulnerability management ecosystem.

Thank you to our sponsors

OWASP 2023

ITALY DAY