How a single NPM package can ruin your business

Rui Quelhas

Principal Software Developer, Oracle

whoami

https://www.ruiquelhas.xyz

- UMinho EI alumnus
- Software engineer at Oracle
- MySQL Middleware and Clients Team
- Lead Developer of the MySQL X DevAPI Connector for Node.js
- Security enthusiast, not expert
- Almost a decade of open source

Let’s chat!

Safe harbor

My views only.

What is at risk?

- emerging threat endangering software developers and suppliers
- targets include mostly
  - vulnerable or insecure source code
  - systems building and compiling that source code
  - workflows to update resulting packages

Goal: infect legitimate components to distribute malware

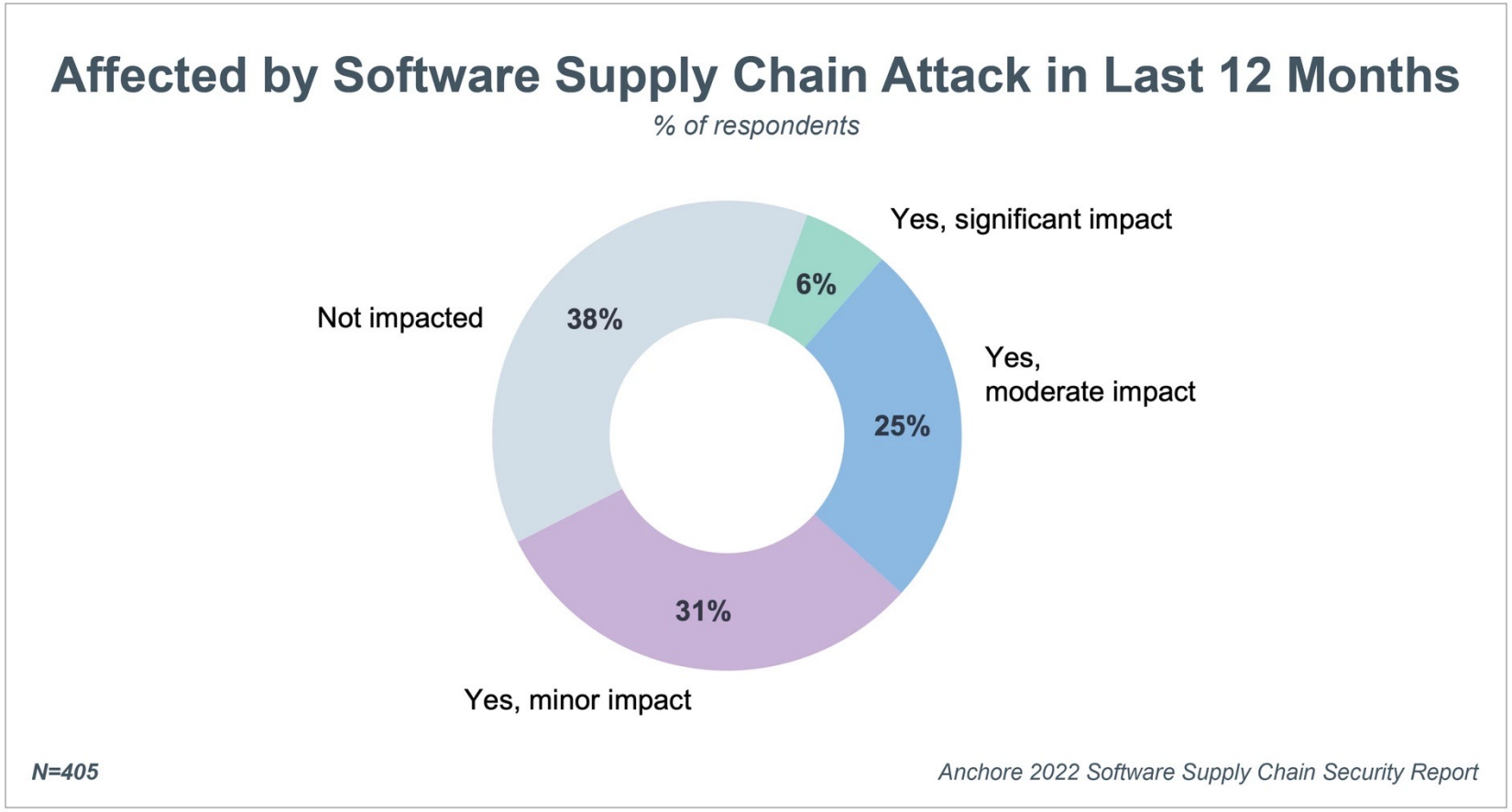
Supply Chain Attack

https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/supply-chain-malware

- multiple types of attacks
  - compromised software building/updating tools or infrastructure
  - stolen code-sign certificates to sign malicious apps
  - malicious code shipped in external components
- big impact, in particular, for 3rd-party software

Reported cases

https://anchore.com/blog/2022-security-trends-software-supply-chain-survey/



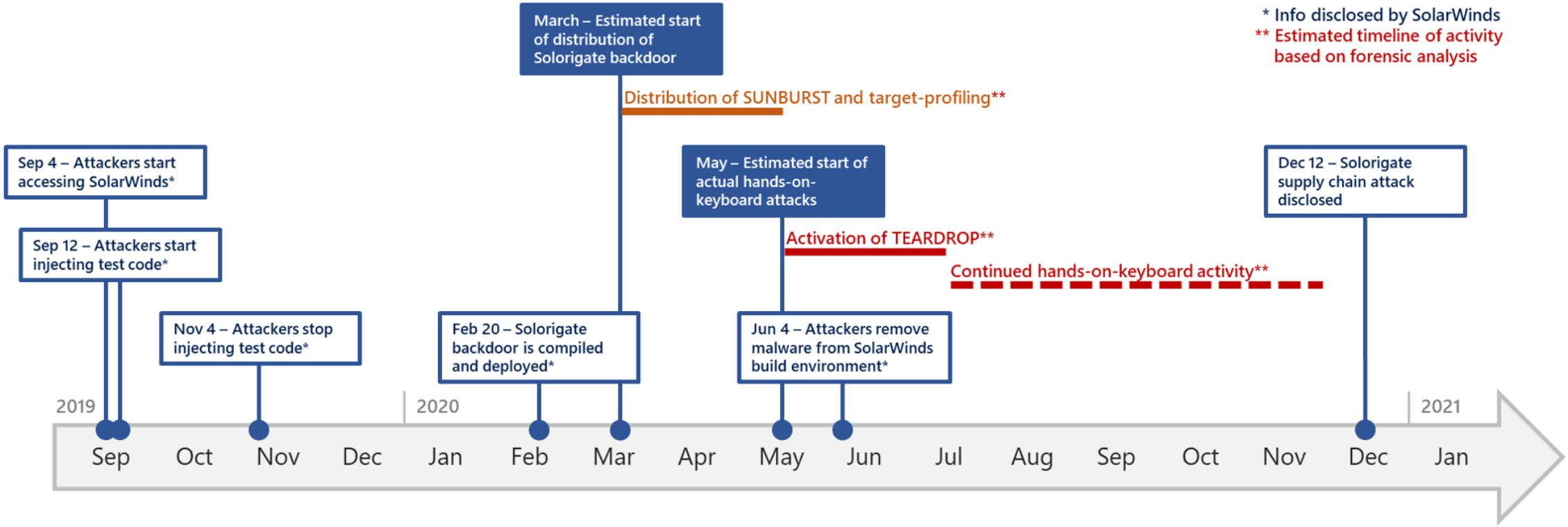
Impact on AppSec

https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

- number of attacks tripled in 2021
- launchpad for other attack vectors
  - sensitive data exfiltration
  - remote code execution
  - ransomware and cryptominers
- driven by popularity of untrusted software distribution pipelines
  - NPM
  - PyPI, NuGET, etc.
- new entry in the latest OWASP Top 10 report

SUNBURST

https://microsoft.com/security/blog/2021/01/20/deep-dive-into-the-solorigate-second-stage-activation-from-sunburst-to-teardrop-and-raindrop/



Catastrophes in the making

- what about attacks we never find out about?
- exploit zero-day vulnerabilities
- software flaws, bugs and security issues
  - overlooked by original authors
  - discovered by potential attackers
- particularly harmful in 3rd-party components
- high-profile examples
  - Heartbleed (CVSS 7.5)
  - Log4Shell (CVSS 10)

Assessing risks

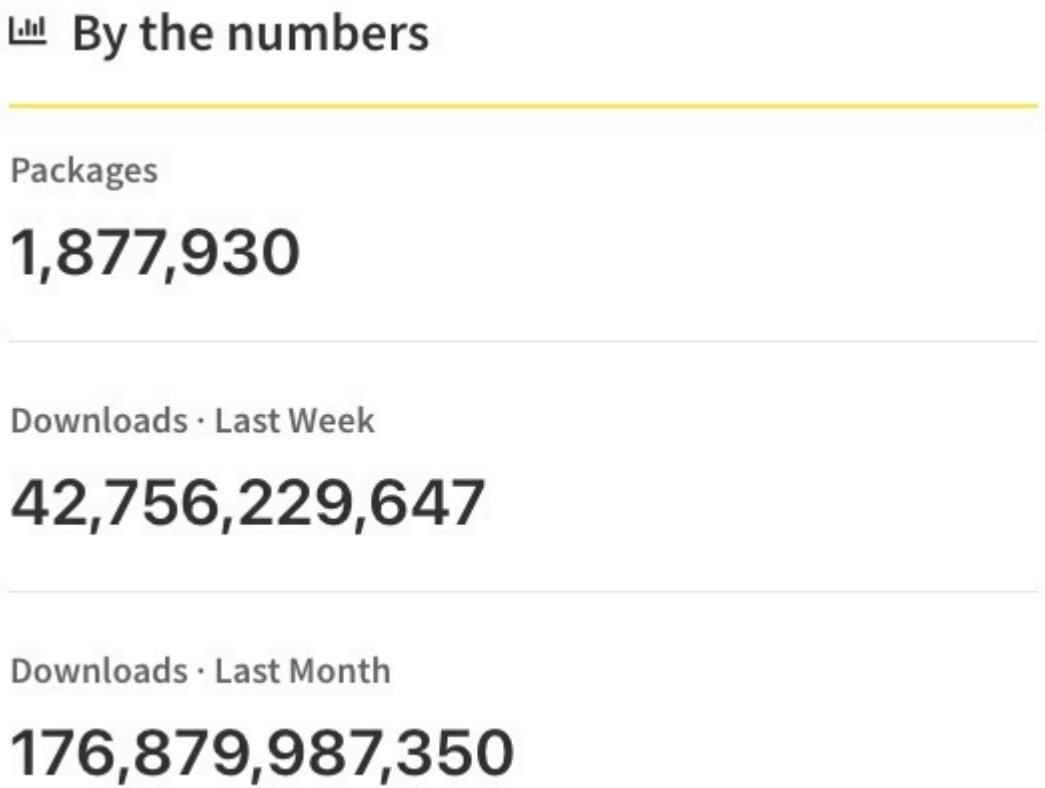
- keeping up-to-date with security fixes is crucial
- known exploits after disclosure (e.g. Equifax)
- smaller size and scope can make software more “auditable”
- Log4Shell a result of feature-creep?
- plugins reduce number of impacted users
- though smaller software
  - leads to bigger dependency trees
  - increases maintenance burden

Open source

- GitHub democratized access to free and open software
- containers brought DevOps to the masses
- SemVer made sense of release versioning
- more individual users writing code
- exponential increase in points of failure
- exploitation by predatory players
- lack of incentives and burnout
- Are “many eyes” enough?

NPM

- open source posterboy
- tasty target for threat actors
- championed small modules
- low signal-to-noise ratio
- born in the age of SemVer
- optimistic default update model
- lack of determinism and trust



https://www.npmjs.com/

Dependency hell

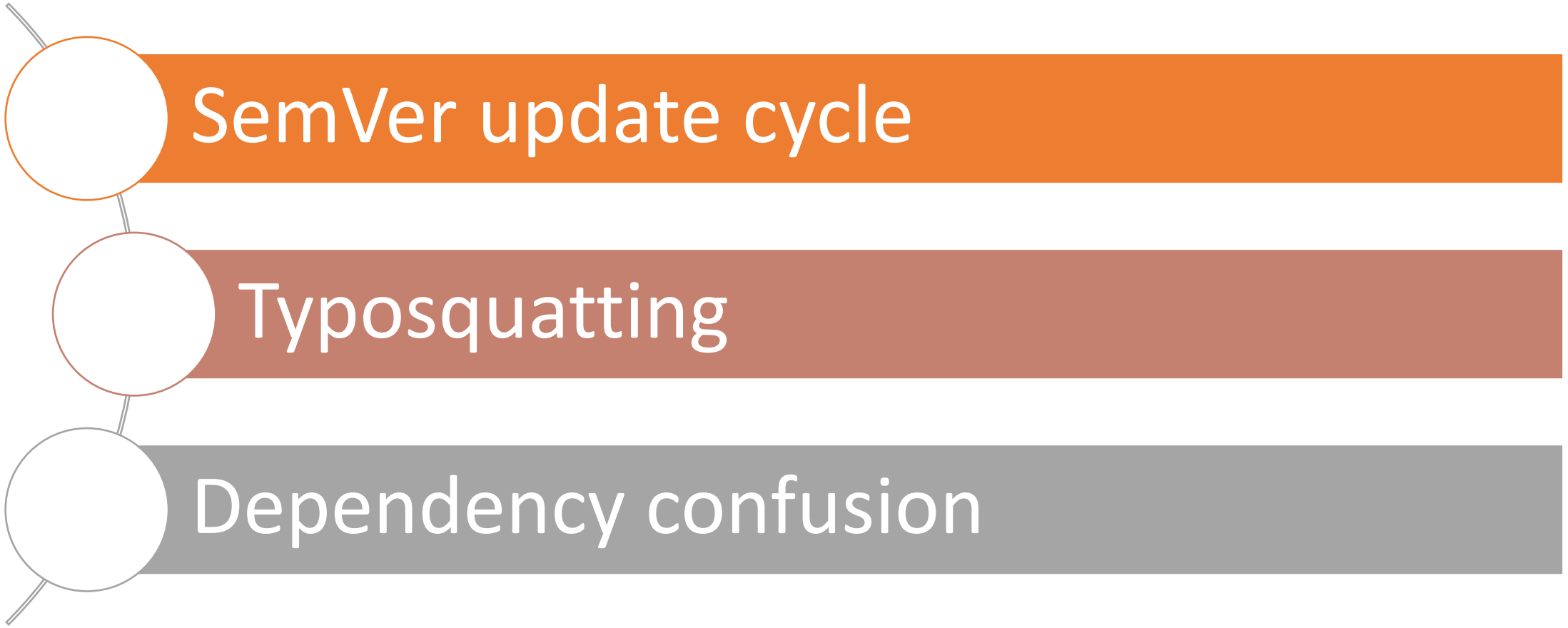
https://www.usenix.org/conference/usenixsecurity19/presentation/zimmerman

*“On average, packages implicitly trust **79** third-party packages and **39** maintainers”*

*“Popular packages often influence more that **100,000** other packages”*

“Some maintainers have an impact on hundreds of thousands of packages”

Favorite attack vectors



All your base are belong to us

- exfiltrating sensitive information
  - access tokens (`eslint-scope` and more)
  - environment variables (`cross-env`)
- stealing bitcoin and cryptocurrency
  - access to wallet private keys (`event-stream` and `electron-native-notify`)
  - cryptominers (`ua-parser-js` and `bluebird`)
  - ransomware
- doing research and proving a point

Frustration “Kik”s in

- risk not always due to malicious intent
- secure and invulnerable code is not enough
- other purely social factors can play a major role
- some notable examples
  - `left-pad`
  - `bebop`
  - `Faker.js`
  - `colors.js`

Securing access

- strong passwords and regular rotation
- 2nd factor authentication
  - TOTP
  - passwordless (WebAuthn)
- limited scope of secrets or access tokens
  - readonly, CIDR-based
  - DO NOT hardcode
  - vaults instead of env variables
- network access control (intranet, proxies, etc.)
- private package registry mirrors (e.g. Verdaccio, Artifactory)

Ensuring trust

- deterministic and reproducible builds
  - checksums
  - lockfiles (`npm-shrinkwrap.json`, `package-lock.json`)
  - Gitian
- trusted contributions with GPG keys
- static analysis and secure coding practices (e.g. Fortify, Sonarqube)
- 3rd-party certification (e.g. Nodesource)
- Software bill of materials (SBOM)

Looking back

- supply chain attacks are a foot in the door
  - zero-day vulnerabilities
  - bad access control policies
  - poor opsec and security hygiene
  - social and human factors
- allow a multitude of additional threats
- can have devastating consequences
- affect everyone from indie developer to nation state
- risk fairly mitigated through common practices

Moving forward

- NPM has taken notice
- internal tools to automate typosquatting detection
- features for publishers and consumers
  - 2nd-factor authentication
  - `npm audit`
  - read-only and CIDR tokens
  - profiles
- more it can do within GitHub/Microsoft
  - from trusted contributions to package signing
  - Actions and integrated infrastructure for reproducible builds
  - Copilot AI for secure coding?

Q&A

Thank You.