Cobbles and Potholes

On the Bumpy Road to Secure Software Supply Chains

Keynote at the Eclipse SAM IoT conference, September 17-18

Henrik Plate (SAP Security Research)

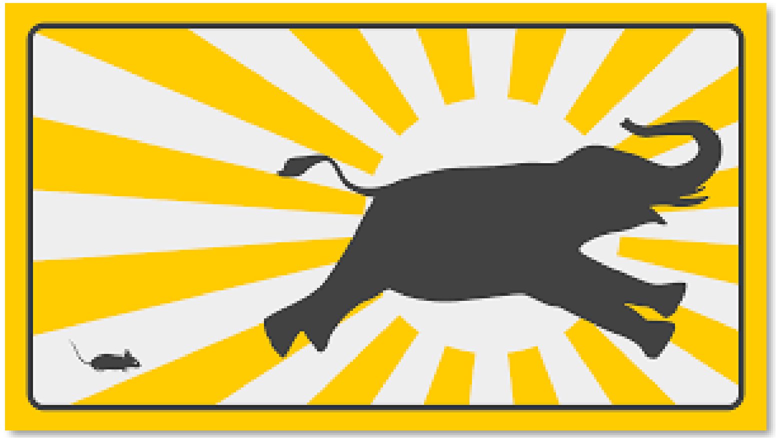
PUBLIC

On the Bumpy Road to Secure Software Supply Chains
Agenda

- About me (and my employer and my team)
- **Cobbles** Dependencies with Known Vulnerabilities
- **Potholes** Supply Chain Attacks
- Closing Remarks & Safeguards

Disclaimers

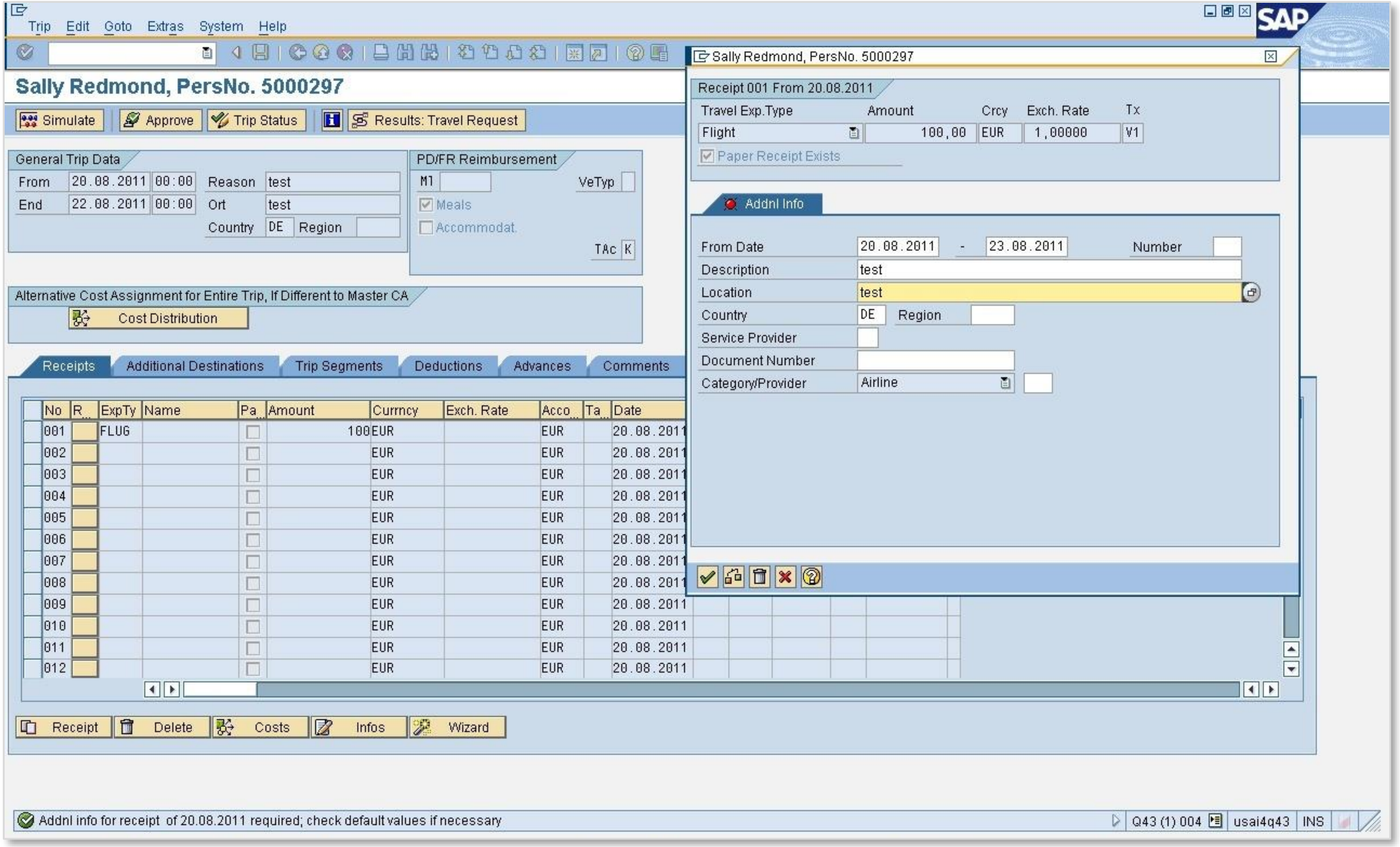
Level of Detail and Fear Mongering



About me
Henrik Plate

- German, 46 years old, living in France for 17 years
- Software developer for > 30 years
- Security researcher for > 10 years (at SAP Labs France)
- Open source contributor for too few years ☹ but catching up with Eclipse Steady
- Cycling enthusiast

About SAP
Open-Source?



About SAP

Active Contributor, User, and Creator of Open Source Software

- Among Top-10 commercial contributors on GitHub, and Top-10 committers to Kubernetes
- Supports numerous foundations as active or sponsoring member
(e.g., ESF, OpenJDK, Cloud Foundry Foundation, Linux Foundation)
- Sep 2019-2020: 3025 unique contributors with 73482 commits to 1153 GitHub repos

Corona-Warn-App

- Published on GitHub
- 50 days from development start to launch
- > 12 Mio. downloads in the first week
- Decentralized data storage and no access to personal or location data on device

About SAP Security Research
Applied Research

- Bridging Academia with SAP Product Development
- 30 researchers, with 50+ peer reviewed publications since 2017 [1]
- 8 strategic research areas [2]

References:
[1] Google Scholar: https://scholar.google.fr/citations?hl=en&user=FOEVZyYAAAAJ&view_op=list_works&sortby=pubdate
[2] Brochure: https://www.sap.com/documents/2017/12/cc047065-e67c-0010-82c7-eda71af511fa.html

Secure ~~Internet of Things~~
Distributed Enterprise Systems

Secure End-to-End Communication (from device to backend)

- Example paper: Towards End-to-End Data Protection in Low-Power Networks [2]
- Goal: Ensure end-to-end security from low-power devices to backend applications
- Encryption scheme for authentication and confidentiality (device-specific keys, frequent changes, …)
- Solution has been deployed on the water distribution network of the City of Antibes in France

Security for Distributed AI-based Software

- Example paper: Security for Distributed Deep Neural Networks [1]
- Goal: Confidentiality of input/output data streams and safeguarding Intellectual Property
- Fully Homomorphic Encryption (FHE) protects weights and biases of all layers (the IP)
- Feasibility evaluated on a Convolutional Neuronal Network (CNN) for image classification deployed on distributed infrastructures

References:
[1] Gomez, L., et al.: Security for Distributed Deep Neural Networks Towards Data Confidentiality & Intellectual Property Protection (2019)
[2] Mikhalev V., et al.: Towards End-to-End Data Protection in Low-Power Networks (2017)

SAP Security Research

Cobbles

Dependencies with Known Vulnerabilities

Heartbleed and Equifax

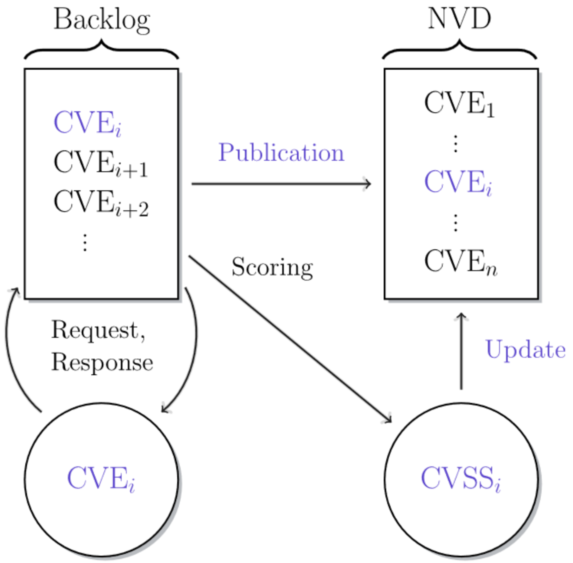
Entering the Hamster Wheel

- **Check** for new vulnerability disclosures (hopefully automated)
- Dismiss false-positives, **assess** true positives (keep fingers crossed for false-negatives)
- **Mitigate** (from *piece-of-cake* to *ridiculously expensive*)
- **Release patch** (cloud ☺ on-premise 😐 devices ☹)

Known Vulnerabilities
CVE and NVD?

Common Vulnerabilities Enumeration (CVE)

- De facto international standard for identification and naming of publicly known vulnerabilities
- Anybody can request CVEs from MITRE or numbering authorities
- 141,567 CVE entries (Sep 15th)



A simplified model for CVSS processing [1]

National Vulnerability Database (NVD)

- Complements CVEs with severity scores (CVSS) and affected products (CPE)
- Searchable

References:
[1] Ruohonen, J.: A look at the time delays in CVSS vulnerability scoring (2019)

Example
Eclipse Mojarra and CVE-2018-14371

The `getLocalePrefix` function in `ResourceManager.java` in Eclipse Mojarra before 2.3.7 is affected by Directory Traversal via the `loc` parameter. A remote attacker can download configuration files or Java bytecodes from applications.

CVSS Base Score: 7.5 (high)

References: fix-commit and issue

Affected products:

- `cpe:2.3:a:eclipse:mojarra:*` up to (excluding) 2.3.7

Problems for app-specific impact assessment:

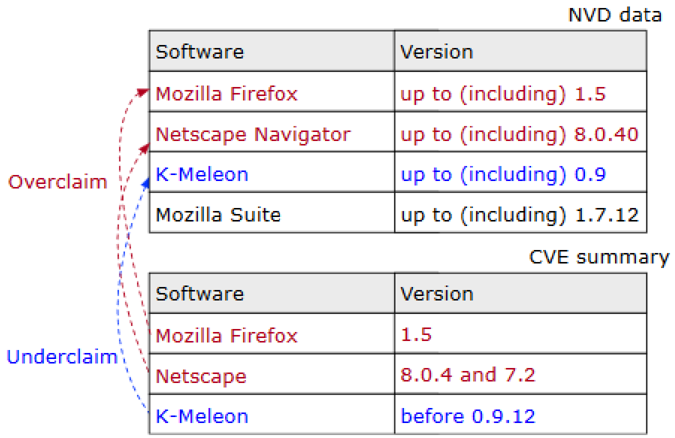
- Short CVE descriptions and varying quality of referenced information
- Transitive dependencies may not be known/understood
- CPE identifier != package identifier (30 search hits for "mojarra" on Maven Central don't include `org.glassfish:javax.faces`)
- Coarse-granular reference of entire projects, ignoring reusable components and code [3] (700+ artifact versions contain the resp. classes)
- Error-prone (2.3.5 and 2.3.6 were also affected)
- Late CVE publication [1,2]
- Some ecosystems are not well covered, e.g., npm

References:
[1] Anwar, A., et al.: Cleaning the NVD: Comprehensive Quality Assessment,Improvements, and Analyses (2020)
[2] The State of Exploit Development: 80% of Exploits Publish Faster than CVEs (2020)
[3] Plate, H., et al.: Impact assessmentfor vulnerabilities inopen-sourcesoftware libraries (2015)

Inconsistencies in Public Security Reports [1]

78,296 CVE IDs and 70,569 vulnerability reports of the past 20 years

- Strict matching: Software names and versions match exactly
- Loose matching: Software names match, versions are subsets (underclaim and overclaim)



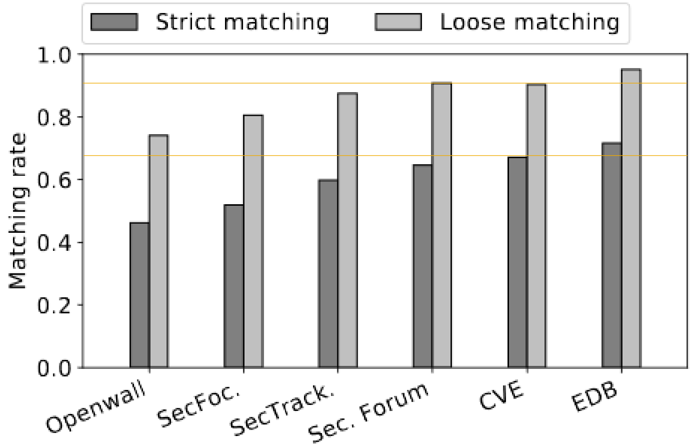
NVD data

Software	Version
Mozilla Firefox	up to (including) 1.5
Netscape Navigator	up to (including) 8.0.40
K-Meleon	up to (including) 0.9
Mozilla Suite	up to (including) 1.7.12

CVE summary

Software	Version
Mozilla Firefox	1.5
Netscape	8.0.4 and 7.2
K-Meleon	before 0.9.12

Figure 9: Example of underclaimed and overclaimed versions.



~90% loose matching rate CVE/NVD

~70% strict matching rate CVE/NVD

Figure 10: Matching rate for different information sources.

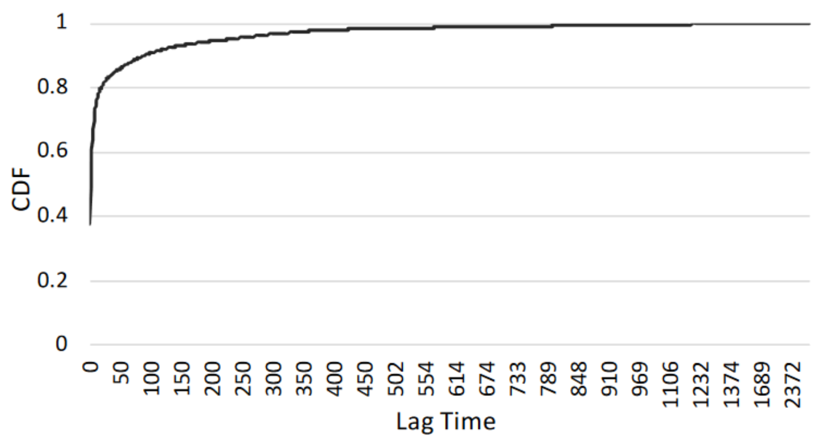
On report-level, across all information sources: Loose matching rate = 90.05%, strict matching rate = 59.82%

References:
[1] Dong, Y., et al.:Towards the Detection of Inconsistencies in Public Security Vulnerability Reports (2019)

NVD Publication Lags
Response Windows are Getting Smaller

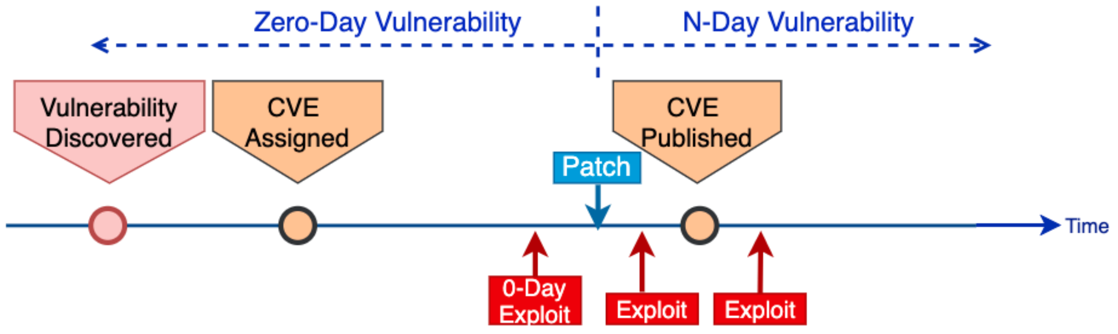
Anwar et al. analyzed lag between publication of 107.2K CVEs and referenced web pages [1]:

- ~38% have a lag of zero day
- ~28% have a lag of more than a week



Palo Alto Networks correlated 11K exploits from EDB with CVE and patch information [2]:

- 14% exploits are published before the patch
- 23% within a week after the patch
- 80% before the CVEs are published



Equifax and CVE-2017-5638

- 3 days between patch (March 7th), data breach and CVE publication (both March 10th) [3]

References:
[1] Anwar, A., et al.: Cleaning the NVD: Comprehensive Quality Assessment,Improvements, and Analyses (2020)
[2] Palo Alto Networks: The State of Exploit Development: 80% of Exploits Publish Faster than CVEs (2020)
[3] Fruhlinger, J.: https://www.csoonline.com/article/3444488/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html (2020)

Open Source Vulnerability Detection
Two Approaches

Metadata-based

- Primarily rely on package names and versions, package digests, CPEs, etc.
- Example: OWASP Dependency Check (light-weight, maps against CVE/NVD)

Code-based

- Detect the presence of code (no matter the package)
- Example: Eclipse Steady (heavy-weight, requires fix-commits)
- Supports impact assessments (static and dynamic analyses), esp. important for later lifecycle phases and non-cloud
- Supports update metrics to avoid regressions [1]

Fig. 2. Static and dynamic paths to vulnerable method

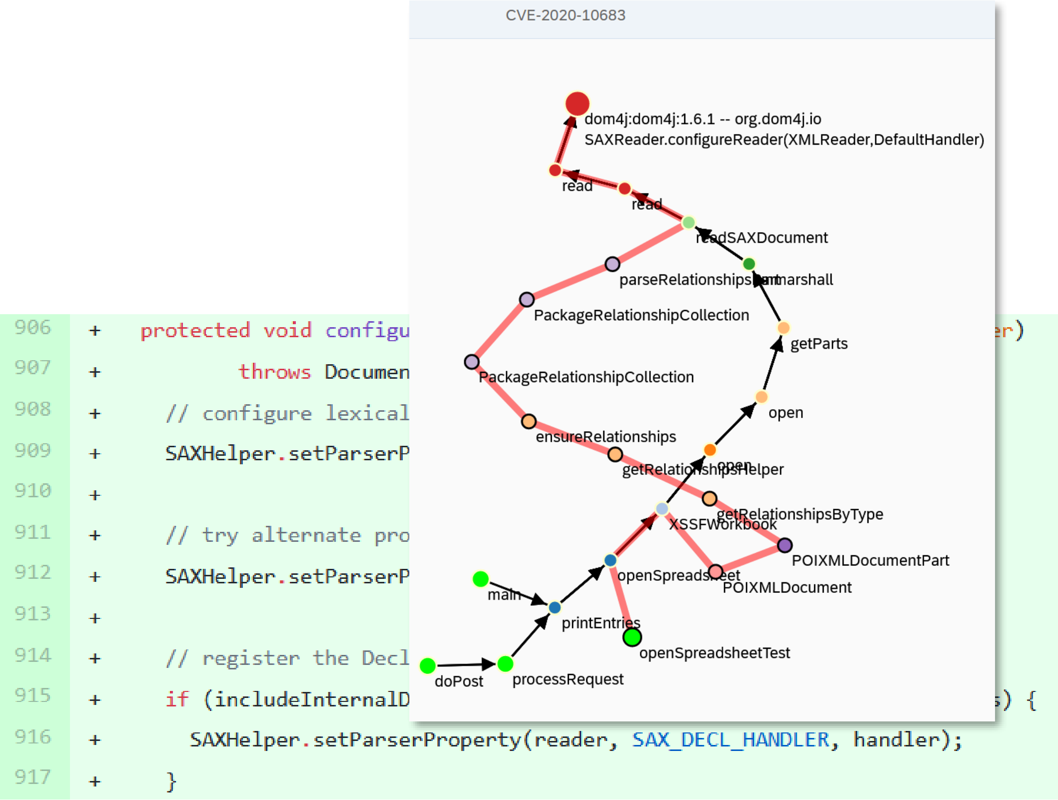


Fig. 1. Fix-commit for CVE-2020-10683

References:
[1] Ponta, S., et al.: Beyond Metadata: Code-Centric and Usage-Based Analysis of Known Vulnerabilities in Open-Source Software (2018)

Vulnerability Data about Open-source Software

Should Be Open Too!

Today

- Information about open source vulnerabilities is scattered
- Mining is labor-intense despite advances in AI-based commit classification [2,3,4]
- Providers step-in (and compete) with proprietary databases

This does not scale, and has the paradoxical consequence that data about open-source software is not open

Project KB supports the creation, management and aggregation of a distributed, collaborative and open knowledge base [1]

References:
[1] EclipseCon 2020: Vulnerability data about open-source software should be open too!
[2] Sabetta, A., et al.: A Practical Approach to the Automatic Classification of Security-Relevant Commits (2018)
[3] Zhou, Y., et al.: Automated identification of security issues from commit messages and bug reports (2017)
[4] Cabrera Lozoya, R., et al.: Commit2Vec: Learning Distributed Representations of CodeChanges (2019)

Leaving the Hamster Wheel
Auto-Upgrade and Semantic Versioning

Obvious: Scan early, often and automated (e.g., as part of commit-triggered build pipelines)

Analysis of 7,470 Node.js projects that use automated PRs or badges [1]

- Projects with automated PRs upgraded 1.6 times more often than baseline
- However, only a third of pull requests were actually merged
- Most significant concerns are breaking changes, understanding the implications of changes, and migration effort

Studies on semver and breaking changes underline the need of code-level analyses [2,3,4]

The number of major, minor and patch releases that contain breaking changes.

Update type	Contains at least 1 breaking change				*Total*
	Yes	*%*	*No*	*%*	
Major	4268	35.8%	7624	64.2%	11,892
Minor	10,690	35.7%	19,267	64.3%	29,957
Patch	9239	23.8%	29,501	76.2%	38,740
Total	24,197	30.0%	56,392	70.0%	80,589

Analysis of the number of breaking and non-breaking changes, edit script size, and release intervals of major, minor, and patch releases.

Type	*#Breaking*		*#Non-break.*		*Edit script*		*Days*	
	μ	σ^2	μ	σ^2	μ	σ^2	μ	σ^2
Major	58.3	337.3	90.7	582.1	50.0	173.0	59.8	169.8
Minor	27.4	284.7	52.2	255.5	52.7	190.5	76.5	138.3
Patch	30.1	204.6	42.8	217.8	22.7	106.5	62.8	94.4
Total	32.0	264.3	52.2	293.3	37.2	152.3	67.4	122.9

References:
[1] Mirhosseini, S, et al.: Can Automated Pull Requests Encourage SoftwareDevelopers to Upgrade Out-of-Date Dependencies? (2017)
[2] Raemaekers, S., et al.: Semantic versioning and impact of breaking changes in the Maven repository (2017)
[3] Decan, A., et al.: What Do Package Dependencies Tell Us About Semantic Versioning? (2019)
[4] Lam, P., et al.: Putting the Semantics into Semantic Versioning (2020)

Dependencies with Known Vulnerabilities
Take-Aways

- CVE/NVD has issues with quality, timeliness and coverage
- No other public database with comprehensive and high-quality information about OSS vulnerabilities is available

Open and comprehensive knowledge base with high-quality and code-level information about vulnerabilities in open source projects

- Automate detection and fixing to cope with shortened response windows
- Code-based approaches improve over approaches based on metadata

SAP Security Research

Potholes

Supply Chain Attacks

“Installing code from a package manager has the same level of security as `curl site.com | bash`” [1]

References:
[1] Saccone, S: NPM Hydra Worm Disclosure (2016)

NPM package `event-stream`
November 2018

- 1.5+ million downloads/week, 1600 dependent packages
- When contacted by mail, the original developer handed-over the ownership to “right9control”
- Added dependency on the malicious package `flatmap-stream`
- Malicious code (and encrypted payload) only present in published NPM package
- Malware and decryption only ran in the context of a release build of the bitcoin wallet `copay`
- `Credentials.getKeys` was monkey-patched and exfiltrated wallet credentials
- Malware was discovered only by incident: Use of deprecated command resulting in a warning

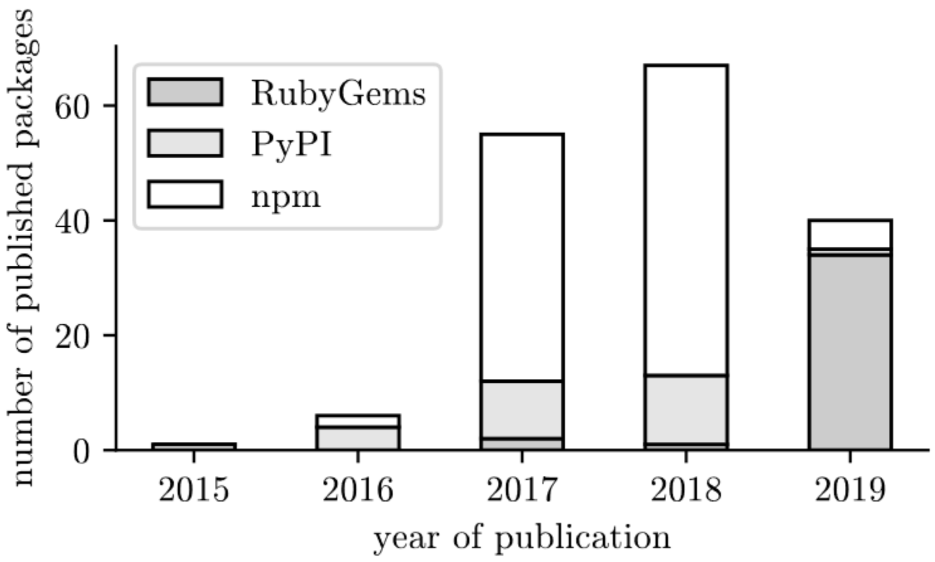
References:
- https://www.theregister.co.uk/2018/11/26/npm_repo_bitcoin_stealer/
- https://medium.com/intrinsic/compromised-npm-package-event-stream-d47d08605502

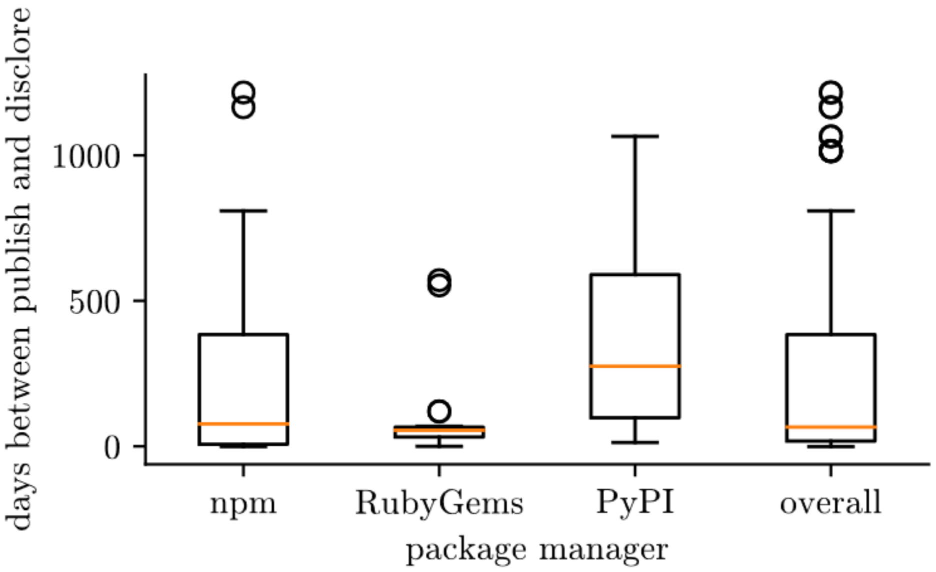
SAP Security Research

Increasing Number of Supply Chain Attacks

Open dataset with 174 malicious packages [1], for which the actual code could be obtained

Manual classification by Ohm et al. [2]: Temporal aspects, trigger, injection technique, conditional execution, primary objective, targeted OS, use of obfuscation, and clusters/campaigns

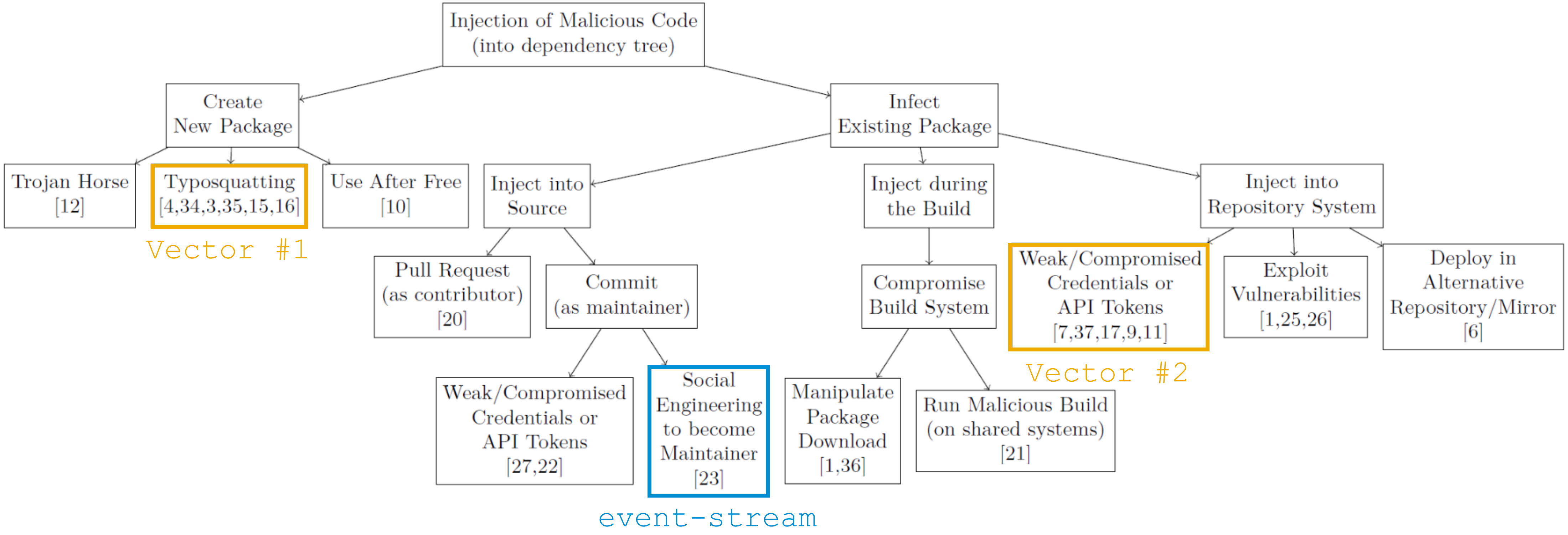




References:
[1] https://github.com/dasfreak/Backstabbers-Knife-Collection
[2] Ohm, M., et al.: Backstabber's Knife Collection (2020)

Attack Tree

Make Downstream Users Depend on a Malicious Package



References:

- Ohm, M., et al.: Backstabber's Knife Collection (2020)

“Attacks abuse
users' trust in the authenticity of packages hosted on external servers,
and their adoption of automated build systems that encourage this practice” [1]

References:
[1] Chess, B., et al.: Attacking the Build through Cross-Build Injection: How Your Build Process Can Open the Gates to a Trojan Horse. (2007)

A Closer Look at Trust
The npm Ecosystem

Metrics defined by Zimmermann et al. [1]

- Package Reach (PR) and Maintainer Reach (MR)
- Implicitly Trusted Packages (ITP) and Maintainers (ITM)

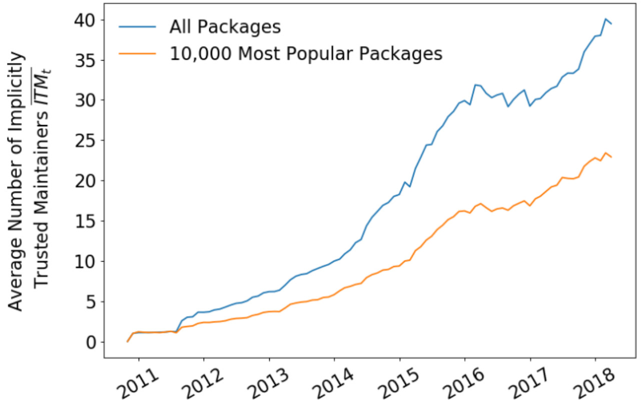


Figure 8: Evolution of average number of implicitly trusted maintainers over years in all packages and in the most popular ones.

Dual-use

- Attackers: “Those maintainers/projects are attractive targets”
- Defenders: “Those require special support and care”

Metric to reflect cost/benefit considerations of attackers, in order to protect likely targets

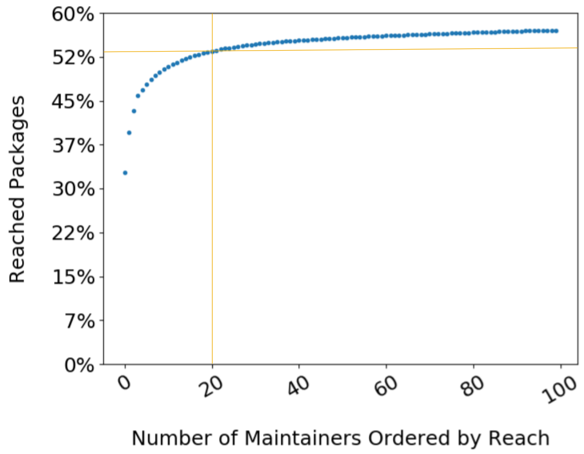


Figure 11: Combined reach of 100 influential maintainers.

References:
[1] Zimmermann, M., et al.: Small World with High Risks (2019)

Trusting Developers to Understand Password Security
Gathering weak npm credentials [1]

Valid credentials of 17088 accounts were bruteforced or leaked.

16901 accounts have published something (~13% of all 125665 accounts).

Directly affected packages: 73983 (14%), indirectly affected packages: ~ 54%

4 users from the top-20 list were affected:

- One who controls > 20 million downloads/month improved the previously revoked password by adding ”!”
- One of those set their password back to the leaked one shortly after it was reset.

References:
[1] Chalker: Gathering weak npm credentials (2017)

“To what extent should one trust a statement that a program is free of Trojan horses? Perhaps it is more important to trust the people who wrote the software.” [1]

References:
[1] Thomson, K.: Reflections on Trusting Trust (1984)

Vulnerable or Malicious?
Telling things apart is difficult!

1) Don’t look at the source code repository but at distributed packages

2) Technically, vulnerable and malicious code can be identical, intention makes the difference
   - Attackers could (re)introduce vulnerabilities and plausibly deny intention
   - Example: Attempt to add the following to `sys_wait4()` in the Linux kernel 2.6 [1]

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
    retval = -EINVAL;
```

= != ==

3) Research and, as far as known, recent attacks, focus on interpreted languages [2,3,4,5]
Detection gets more difficult with compilation, code generation, re-bundling, re-packaging, …

References:
[1] Wysopal, C., End, C.: Static Detection of Application Backdoors (2010)
[2] Vu, Duc-Ly, et al.: Poster: Towards Using Source Code Repositories to Identify Software Supply Chain Attacks (2020)
[3] Pfretzschner, B., et al.: Identification of Dependency-based Attacks on Node.js (2017)
[4] Garret, K., et al.: Detecting suspicious package updates (2019)
[5] Taylor, M., et al.: SpellBound: Defending Against Package Typosquatting (2020)

Fear, Uncertainty and Doubt?
Yes and no…

SCA tool vendors issue yearly reports

- “In the past 12 months, the number of next generation cyber attacks aimed at actively infiltrating open source increased 430%” [1]

  On low numbers: 216 → 929

Still, they pinpoint real deficiencies and threats:

- Promising attack opportunities with comparably little effort
- Upon large-scale exploitation, cf. Hydra Worm, users of affected ecosystems may lose trust and pivot to other ecosystems

Comparative study of the security maturity of ecosystems à la “Measuring and Preventing Supply Chain Attacks” [2]

TABLE I: Framework for comparison of registries.

Features			Registries: PyPI	Npm	RubyGems
Functional	Access	Password	●	●	●
Functional	Access	Access Token	◑	●	●
Functional	Access	Public Key Auth	○	○	○
Functional	Access	Multi-Factor Auth	◑	◑	◑
Functional	Publish	Upload	●	●	●
Functional	Publish	Reference	○	○	○
Functional	Publish	Signing	◑	◑	◑
Functional	Publish	Typo Guard	○	●	●
Functional	Publish	Namespace	○	◑	○
Functional	Manage	Yank Package	◑	◑	◑
Functional	Manage	Deprecate Package	○	◑	◑
Functional	Manage	Add Collaborator	◑	◑	◑
Functional	Manage	Transfer Ownership	◑	◑	◑
Functional		Reputation	●	●	●
Functional	Select	Code Quality	○	○	○
Functional	Select	Security Practice	○	○	○
Functional	Select	Known Issue	○	○	○
Functional		Typo Detection	○	○	●
Functional	Install	Hook	●	◑	○
Functional	Install	Dependency Locking	○	◑	◑
Functional	Install	Native Extension	◑	◑	◑
Functional	Install	Embedded Binary	◑	◑	◑
Review	Metadata	Dependency Check	○	○	○
Review	Metadata	Update Inspection	○	○	○
Review	Metadata	Binary Inspection	○	○	○
Review	Metadata	PM Account	○	○	○
Review	Static	Stylistic Lint	○	○	○
Review	Static	Logical Lint	○	○	○
Review	Static	Suspicious Logic	○	○	○
Review	Dynamic	Install	○	○	○
Review	Dynamic	Embedded Binary	○	○	○
Review	Dynamic	Import	○	○	○
Review	Dynamic	Functional	○	○	○
Remediation	Remove	Package	●	●	●
Remediation	Remove	Publisher	●	●	●
Remediation	Remove	Installed Package	○	○	○
Remediation	Notify	PM	○	○	○
Remediation	Notify	Dependent PM	○	○	○
Remediation	Notify	Dev	○	○	○
Remediation	Notify	Advisory DB	○	●	●

unsupported - ○, optional - ◑, enforced - ●

References:
[1] Sonatype: State of the Software Supply Chain (2020)
[2] Duan, R., et al.: Measuring and Preventing Supply Chain Attacks onPackage Managers (2020)

Apropos effort and trust…
XcodeGhost

- Modified versions of Apple's Xcode development environment
- Distributed on 3rd party download sites (popular due to slow downloads from Apple servers)
- The modified the linker links the malicious "CoreServices" object file to the executable of any compiled app (whereby this is hidden from Xcode's UI but only visible in the compile logs)
- Used HTTP to upload device information and receive commands from a C&C server
- Control infected apps to open arbitrary URLs in any scheme (http://, itunes://, …)
- Several thousand apps were infected, including the popular messaging app WeChat

Too far fetched? A comparable attack targeted AndroidAudioRecorder [3]

References:
[1] Palo Alto Networks: More Details on the XcodeGhost Malware and Affected iOS Apps (2015)
[2] Palo Alto Networks: Update: XcodeGhost Attacker Can Phish Passwords and Open URLs through Infected Apps (2015)
[3] Braun, M.: A confusing dependency (2018)

Supply Chain Attacks
Take-Aways

- Many people thank you for putting trust in their security capabilities
- Number of dependencies and actors + complexity of build processes and infrastructures result in a considerable the attack surface
- Noticeable increase in supply chain attacks targeting open source ecosystems
- Python, Node.js and Ruby ecosystems are the primary targets
(but some ecosystems like Java have not been analyzed in a systematic fashion)

Protection against malicious open source components

- All dependencies matter (not only compile/runtime ones as for known vulnerabilities)
- The truth is in downloaded packages (source code visible in GitHub etc. does not matter)

Active field of research, e.g., as part of EU Research Project SPARTA [1]

References:
[1] https://www.sparta.eu/

Open Source Security

Closing Remarks & Safeguards

Closing Remarks

A comprehensive and comparative study of the effectiveness and costs of existing safeguards for different ecosystems (and a gap analysis)

The next page contains a selective and opinionated list of (mostly) technical safeguards...

Of course, commercial users should support their respective upstream projects or infrastructure providers (PyPI, e.g., has < 10 admins for > 450K package owners and > 260K projects [1,2])

Out of scope of this presentation

- Government regulations and standards
- Liability of commercial software vendors
- *“Moral responsibility”* of open source project maintainers vis-à-vis downstream users

References:
[1] https://warehouse.readthedocs.io/application/
[2] https://pypi.org/

Selective List of (Mostly) Technical Safeguards

For Roles (U)ser, (M)aintainer and (P)ackage Repository

	Safeguard	U	M	P	Cost Guesstimate
Well-known and cheap (mostly)	Integrate open source vulnerability scanners into CI/CD pipelines	X			$
	Use (enforce) multi-factor authentication		X	X	$
	Version pinning and automated PRs for upgrades	X			$
	Disable script execution during package installation	X			$
	Containerize and constrain builds, no caching, minimal release builds	X	X		$
	Use of security, health and quality metrics, e.g., CII Badge Program	X	X	X	$
	Establish internal repository mirrors	X			$
	Establish a process to identify vetted and sanctioned components	X		X	$$
Mitigate untrusted build env.	Build dependencies from source	X			
	Reproducible builds		X	X	
	Create and verify PGP signatures, incl. a trusted list of providers	X	X	X	
	Slice dependencies to reduce attack surface [1,2]	X			

References:
[1] Soto-Valero, C., et al.: A Comprehensive Study of Bloated Dependencies inthe Maven Ecosystem (2020)
[2] Haas, R., et al.: Is Static Analysis Able to Identify Unnecessary Source Code? (2019)

Thank you.

Henrik Plate

Senior Researcher

SAP Security Research

SAP Labs France, 805 Av. Maurice Donat

F-06250 Mougins