Le génie pour l'industrie

UNIVERSITÉ DU LUXEMBOURG

Dependabot and Security Pull Requests: A Large Empirical Study

CYBERUS Summer School 2023

Realized by :

- **Mr REBATCHI Hocine**

Under the supervision of :

- **Prof. BISSYANDÉ Tegawendé** *(SnT - University of Luxembourg)*
- **Prof. MOHA Naouel** *(ÉTS)*

04/07/2023

Hocine REBATCHI

PhD Student (Applied Research profile)

Software Engineering

Software Security

Machine Learning

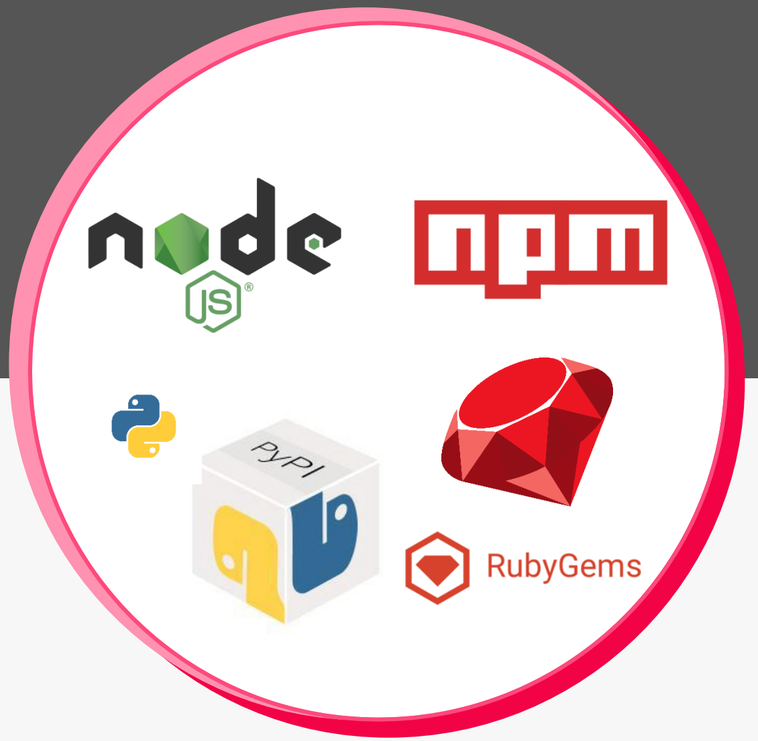
- **Software Engineer** + **Master Degree :** Fall 2015 – Fall 2020
- **PhD Student :** Winter 2021 – Fall 2024

Table of Contents

Context

Software development has a supply chain

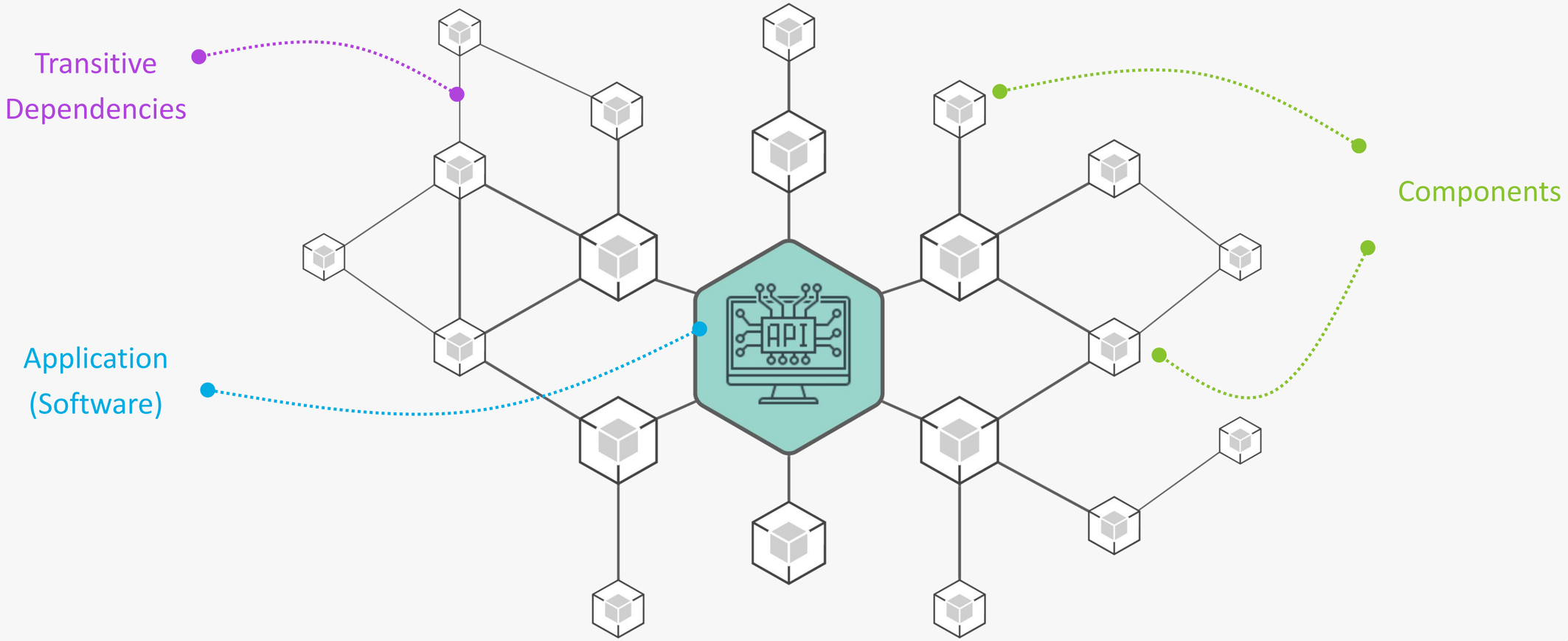
Depends on third-party components (packages, libraries)



85% - 97% of enterprise software code base from OS components

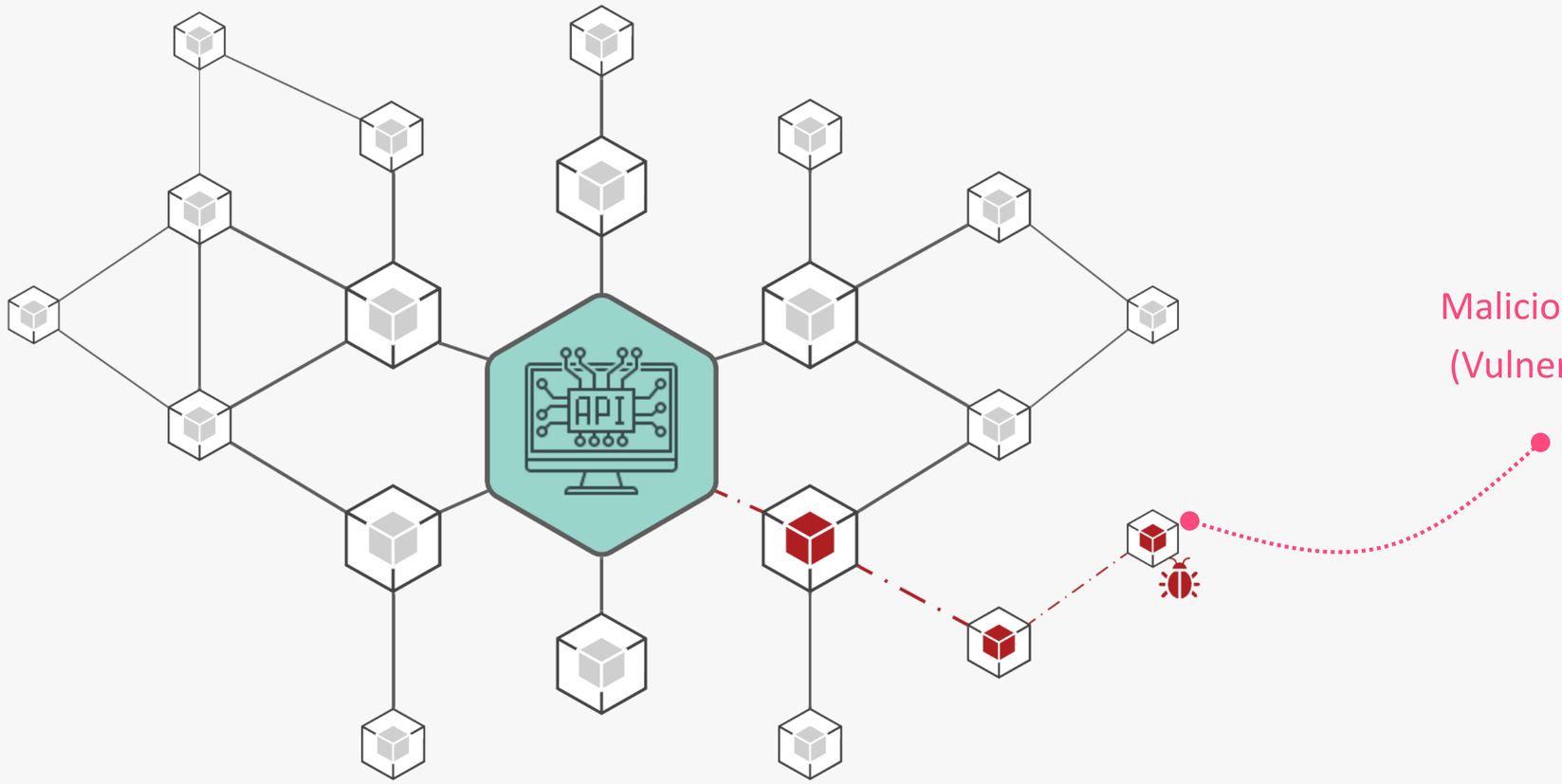
Background

Software Supply Chain



Background

Software Supply Chain Attack



Background

Vulnerability

What is a security vulnerability ?

- Security defects
- Security bugs
- Software weaknesses
- Etc.

According to Ghaffarian and Shahriari [1] :

"A software vulnerability is an instance of a flaw, caused by a mistake in the design, development, or configuration of software such that it can be exploited to violate some explicit or implicit security policies."

Background

Software Supply Chain Attack

What is a Software Supply Chain Attack (SSCA)?

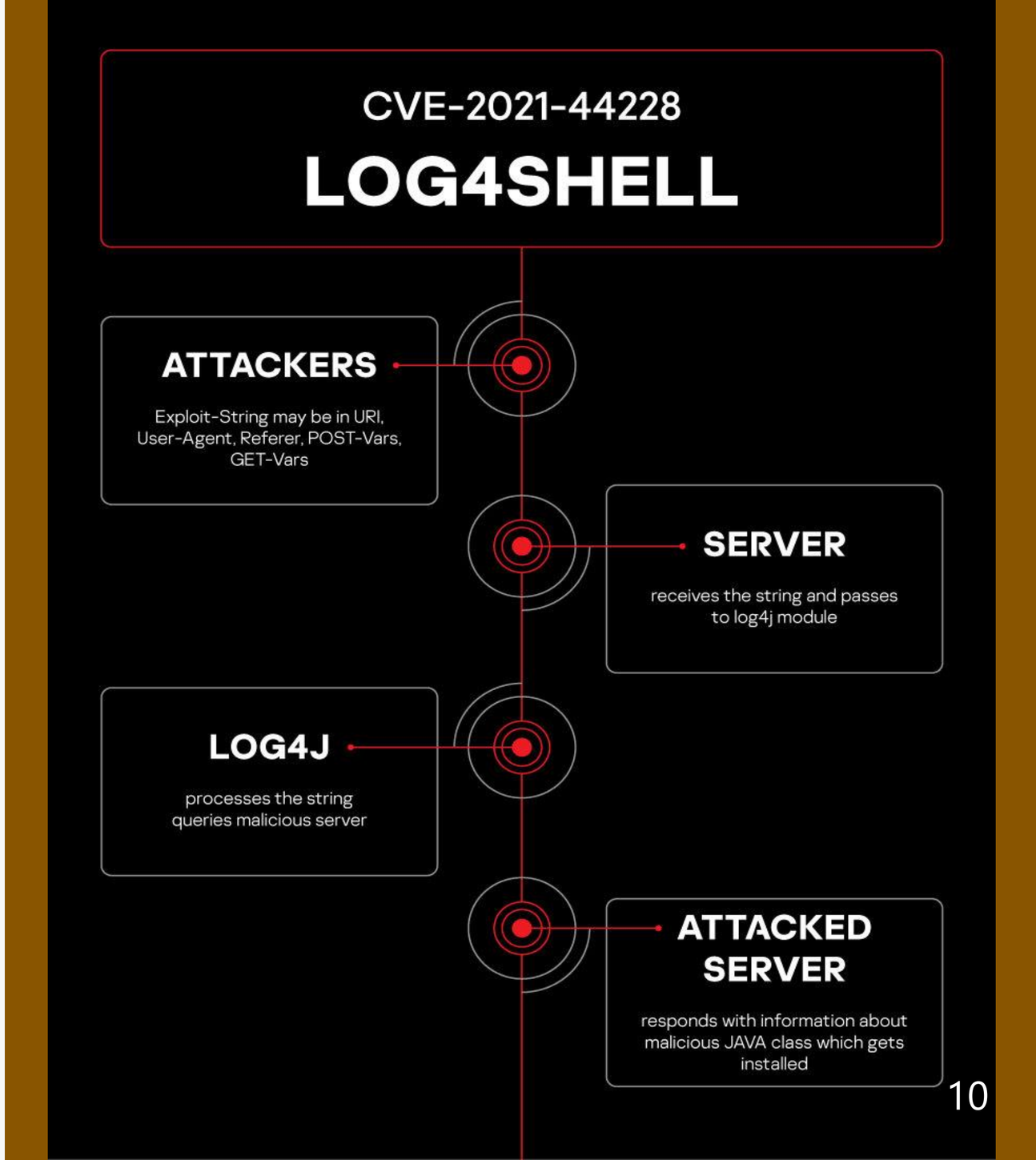
A technique in which an adversary slips **malicious code** or even a **malicious component** into a trusted piece of software or hardware. By compromising a single supplier, attackers can hijack the **distribution system** to turn any application into **Trojan horse** [2].

Data Exfiltration:
SolarWinds Attack

- In December 2020, Russian hackers from the Foreign Intelligence Service (SVR) hacked **SolarWinds**.
- In October 2019, they planted **malicious code** in **updates** of the network monitoring tool Orion to plant a **backdoor**.
- **18 000 users** were affected, and at least nine **US federal agencies** got infiltrated (e.g., NASA, the State Department, the Department of Defense, and the Department of Justice).

Remote Code Execution: Log4Shell Attack

- On December 2021, a **vulnerability** with **10/10 severity** was discovered in Apache **Log4j** library.
- Vulnerability consists of **abusing** the feature of specifying **code** through a **log message** and allowing the injected code to be **executed remotely** on a targeted server.
- Exploits: **Cryptomining**, **Reverse Shell** to bypass firewalls, turn targeted server into a **botnet**, **data exfiltration**, etc.

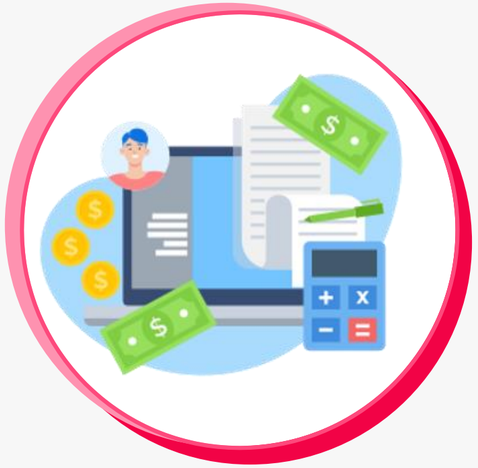


Research Problems

Handling Software Supply Chain Attacks

Software supply chain is **extensive**, and software is **updated** and **patched** on a regular basis

The **lacks** and **limitations** of the state-of-the-art (Accuracy & Time)



Manual analysis performed by experts require a lot of **time** and **effort**

The **fix delay** that expands the **window of exposure** causing more **casualties**

Scope

SCA Tools & Dependabot

Software Composition Analysis Tools

Tools that **identify** the **open-source software** in a codebase in order to evaluate **security**, **license compliance**, and code quality. The inspection concerns different components and packages against **security-related databases** (e.g., NVD) that contain information about common and **known vulnerabilities**.

Dependabot

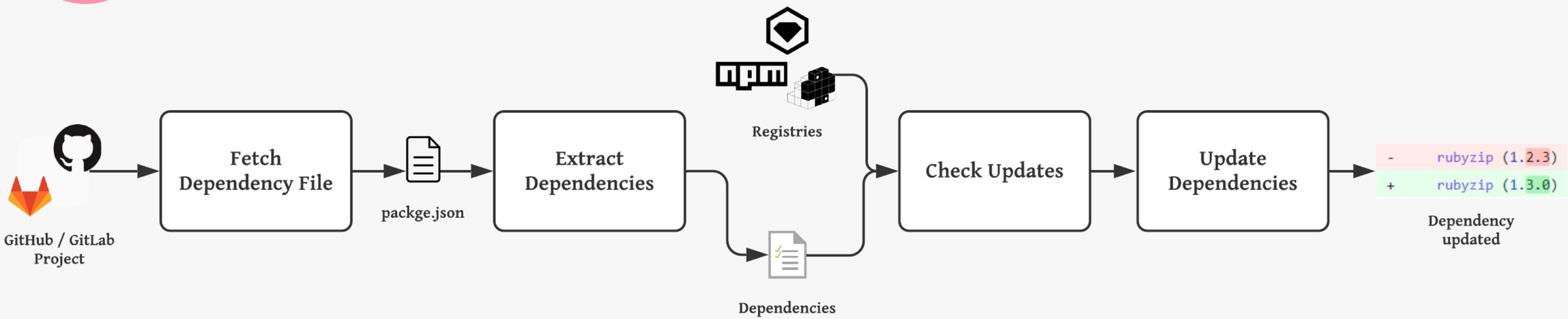
Automated tool that keeps **dependencies secure** and **up-to-date** by managing dependency updates, scanning **third party vulnerabilities** and sending security alerts.

It was released on **May 27, 2017**, and then, it got acquired by GitHub on **May 2019**. It currently supports **15** different programming languages.

Scope

SCA Tools & Dependabot

Dependabot



Overview of Dependabot working process

https://github.com/dependabot/dependabot-core

Research Questions

Scope of study

01

Dependabot popularity

- Level of popularity
- Popularity reasons

02

Vulnerabilities in dependencies

- Patterns of developers' practices and techniques

03

Security PRs management

- Receptiveness and responsiveness
- Threat lifetime & fix delay
- Most exploited vulnerabilities

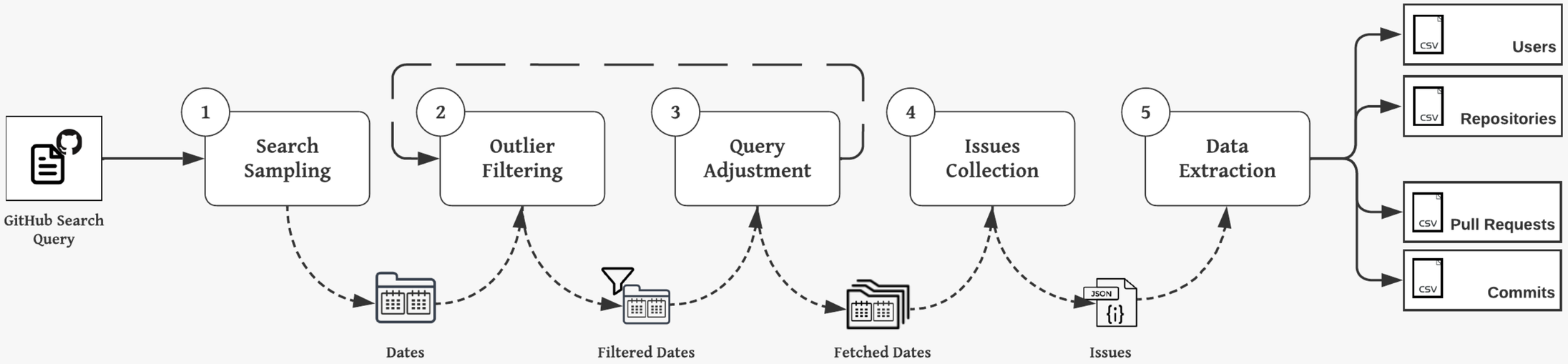
04

Merge decision & Merge speed

- Factors correlating with the acceptance and fast merges

Collected Datasets

GitHub-Miner : dataset collection pipeline



Data collection pipeline for GitHub-Miner

Collected Datasets

Dataset description

Dataset 1

Dependency Update :

6 573 489 PR-related issues created from 26/05/2017 to 15/06/2021 (63 Gb).

Dataset 2

Dependabot Security PRs :

384 764 pull requests created from 26/05/2017 to 15/06/2021 (4.34 Gb).

Dataset 3

Manual Security PRs :

100 102 pull requests created from 26/05/2017 to 15/06/2021 (1.66 Gb).

Research Methodology

RQ1

To what extent is Dependabot adopted ?

Why is Dependabot more adopted than other tools ?

Level of popularity

- Using **Dataset (1)** - Dependency Update -
- **Quantitative analysis** of the total number of PRs created by bots and users
- **Comparative analysis** of the history and evolution of dependency management activity

Research Methodology

RQ1

To what extent is Dependabot adopted ?

Why is Dependabot more adopted than other tools ?

Popularity reasons

- **Survey** with project owners from GitHub, randomly selected from **Dataset (1)**
- Content : demographic profile and experience + dependency management tools and their features + challenges encountered with possible improvements
- Response rate of **13%** (22/164)

Research Methodology

RQ2 What do developers do to handle security vulnerabilities in dependencies ?

Patterns of developers' practices & techniques

- Using **Dataset (2) & (3)** – Security PRs –
- Representative sample : more than 10% of total PRs (**50,000**) using Stratified Random Sampling
- **Manual qualitative analysis** of PR commits, patches, & comments

Research Methodology

RQ3

How fast are security pull requests handled ?

How long do vulnerabilities remain unpatched ?

Receptiveness and responsiveness

- Using **Dataset (2) & (3)** – Security PRs –
- **Comparative analysis** of the distribution of PRs
- **Manual analysis** for the reasons of closing and not handling security PRs
- Measure merge speed & close speed of PRs for Dependabot and developers

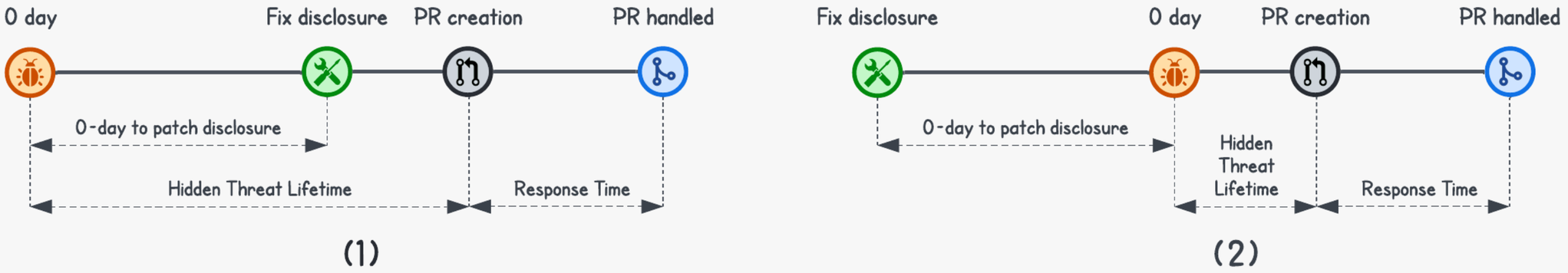
Research Methodology

RQ3

How fast are security pull requests handled ?

How long do vulnerabilities remain unpatched ?

Threat lifetime & fix delay



Timeline for vulnerabilities discovery time and fix time; (1) Patch disclosed after adding the vulnerable dependency, (2) Patch disclosed before adding the vulnerable dependency

Research Methodology

RQ3

How fast are security pull requests handled ?

How long do vulnerabilities remain unpatched ?

Most exploited vulnerabilities

- Using **Dataset sample** of RQ2 (50,000)
- **Quantitative analysis** of the Hidden Threat Lifetime by vulnerability type

Research Methodology

RQ4 What factors influence the decision and the time to accept security PRs ?

Factors correlating with the acceptance and fast merges

- Using **Dataset (2) & (3)** – Security PRs –
- **Data pre-processing** :
  - Cross-correlation analysis (redundancy, independence, significance)
  - Outlier filter
- **Statistical analysis** on the merge decision and the merge speed
- **Survey** with developers, randomly selected from Dataset (2) (response rate 14% = 18/128)

Research Methodology

RQ4

What factors influence the decision and the time to accept security PRs ?

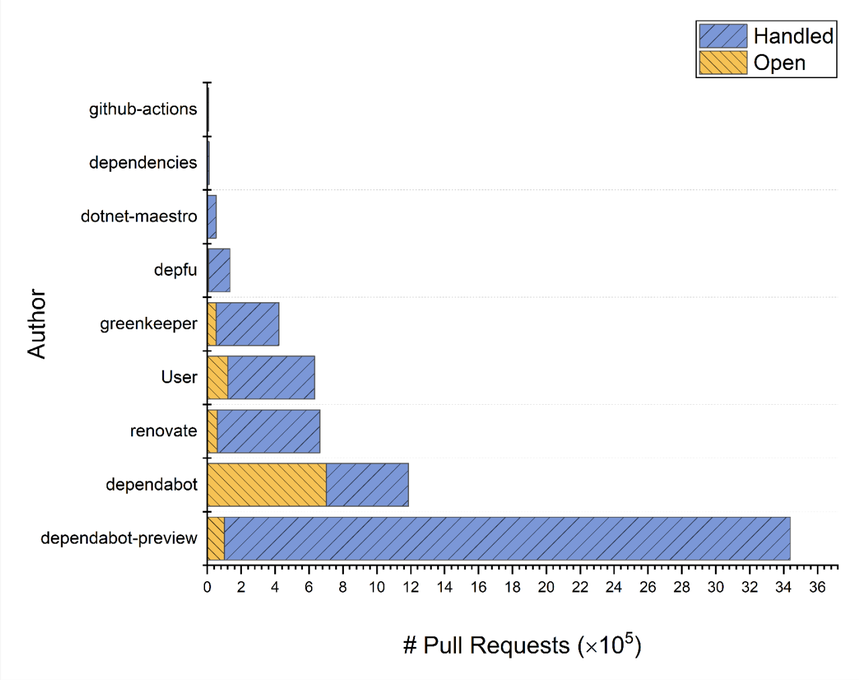
Category	Feature	Description
Repository	`age`	Age of the repository from its creation date to the PR creation time (in days)
	`recent_activity`	Time interval between the last update in the repository and the PR creation time (in days)
	`size`	Size of the repository (in Kb)
	`# watchers`	Number of GitHub users that register to watch the repository for new updates notifications
	`# open_issues`	Number of the total open issues that are registered and not handled in the repository
Pull Request	`# assignees`	Number of GitHub users that are assigned to the issues related to the PR
	`# requested_reviewers`	Number of GitHub users that are requested to review the code in the PR
	`# commits`	Number of commits that perform the changes suggested in the PR
	`# additions`	Number of lines of code added in the commits of the PR
	`# deletions`	Number of lines of code deleted in the commits of the PR
	`# changed_files`	Number of files changed by the commits of the PR
	`# comments`	Number of comments in the discussion history of the PR
	`discussion_size`	Size of the body of the PR (i.e., words count)
User	`experience`	Time interval between the creation date of the user account and the PR creation time
	`author_association`	Association of the GitHub user to the project repository (i.e., owner, contributor)
	`# followers`	Number of the GitHub user followers
	`# public_repos_gists`	Number of public repositories and gists created by the GitHub user
Dependency	`patch_level`	Specification of the version update of the dependency in the PR (i.e., patch, minor, major)
	`severity`	Level of severity for the dependency vulnerability (i.e., low, moderate, high, critical)
	`is_bloated`	Indicator if the dependency is used in the repository or bloated (not used)

Investigations & Findings

RQ1. Dependabot popularity

RQ1.1. Level of popularity

- **Dependabot** dominates the dependency management activity, with more than **70%** PRs
- **84%** of the total PRs in the dataset are **handled**
- **Auto-generated** PRs (90%) vs. **Manual** PRs (10%)



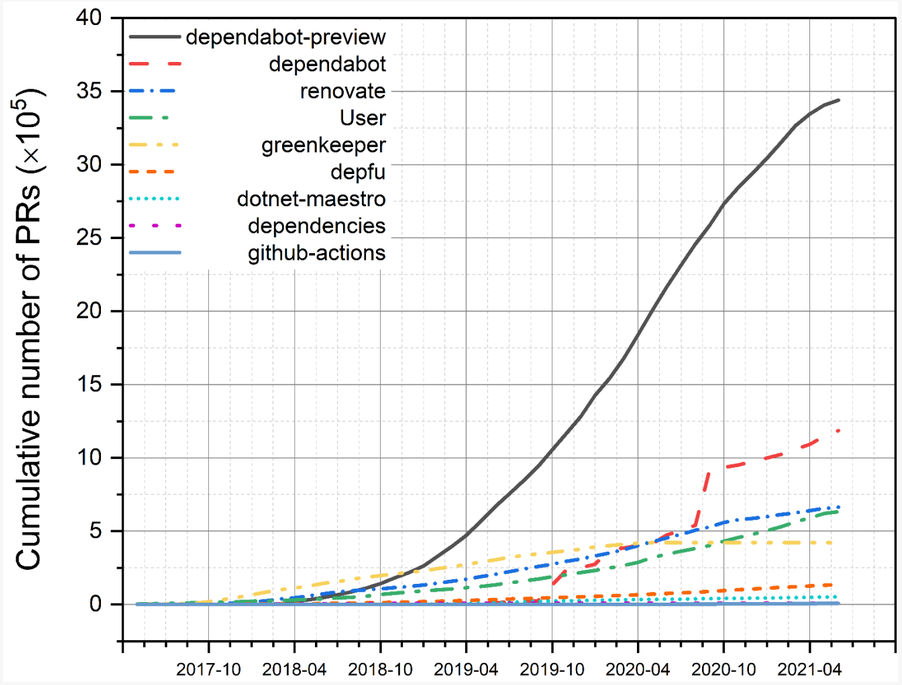
Pull Request distribution per author

Investigations & Findings

RQ1. Dependabot popularity

RQ1.1. Level of popularity

- **Dependabot** dominates the dependency management activity, with more than **70%** PRs
- **84%** of the total PRs in the dataset are **handled**
- **Auto-generated** PRs (90%) vs. **Manual** PRs (10%)
- **Dependabot** increasingly getting more popular, esp. from **2018** when most **PLs** were supported
- Dependabot creates on avg. **68,784 new PRs** per month



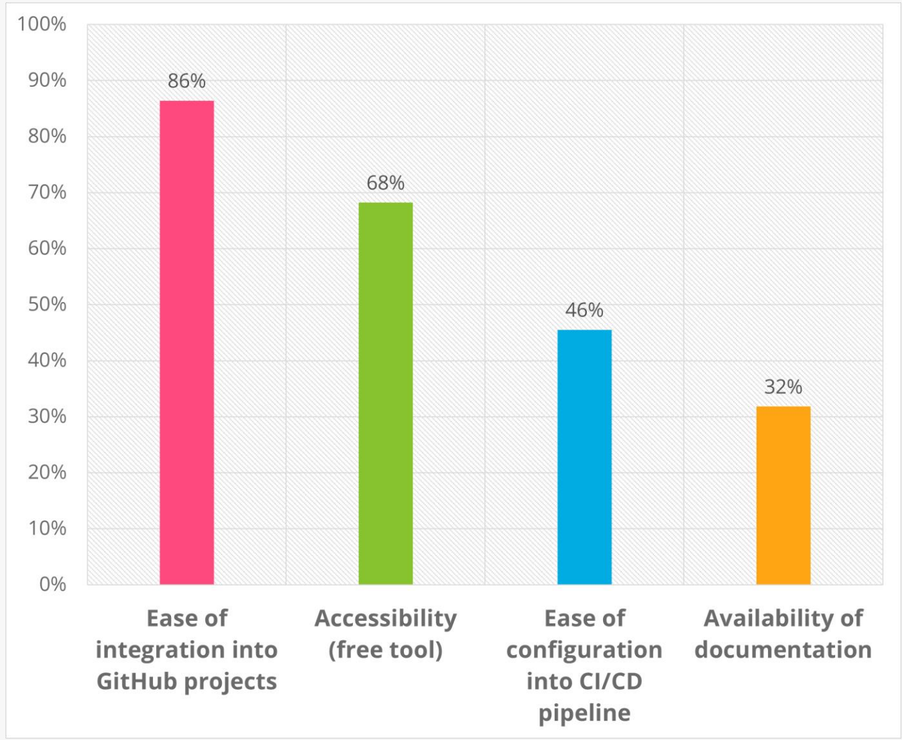
Pull Request creation history per author

Investigations & Findings

RQ1. Dependabot popularity

RQ1.2. Popularity reasons

- Main features :
  - ✓ **Efficiency** : adoption of automated dependency management
  - ✓ **Accessibility** : free tool + PLs
  - ✓ **Adaptivity** : CI/CD pipeline + modern software development
  - ✓ **Comprehensibility and support**



Selection rate of popularity reasons

Investigations & Findings

RQ2. Vulnerabilities in dependencies

RQ2. Patterns of developers' practices & techniques

Strategies :

Identify & Fix

Investigations & Findings

RQ2. Vulnerabilities in dependencies

RQ2. Patterns of developers' practices & techniques

Strategies :

Identify & Fix

SCA Tools

Investigations & Findings

RQ2. Vulnerabilities in dependencies

RQ2. Patterns of developers' practices & techniques

Strategies :

Identify & Fix

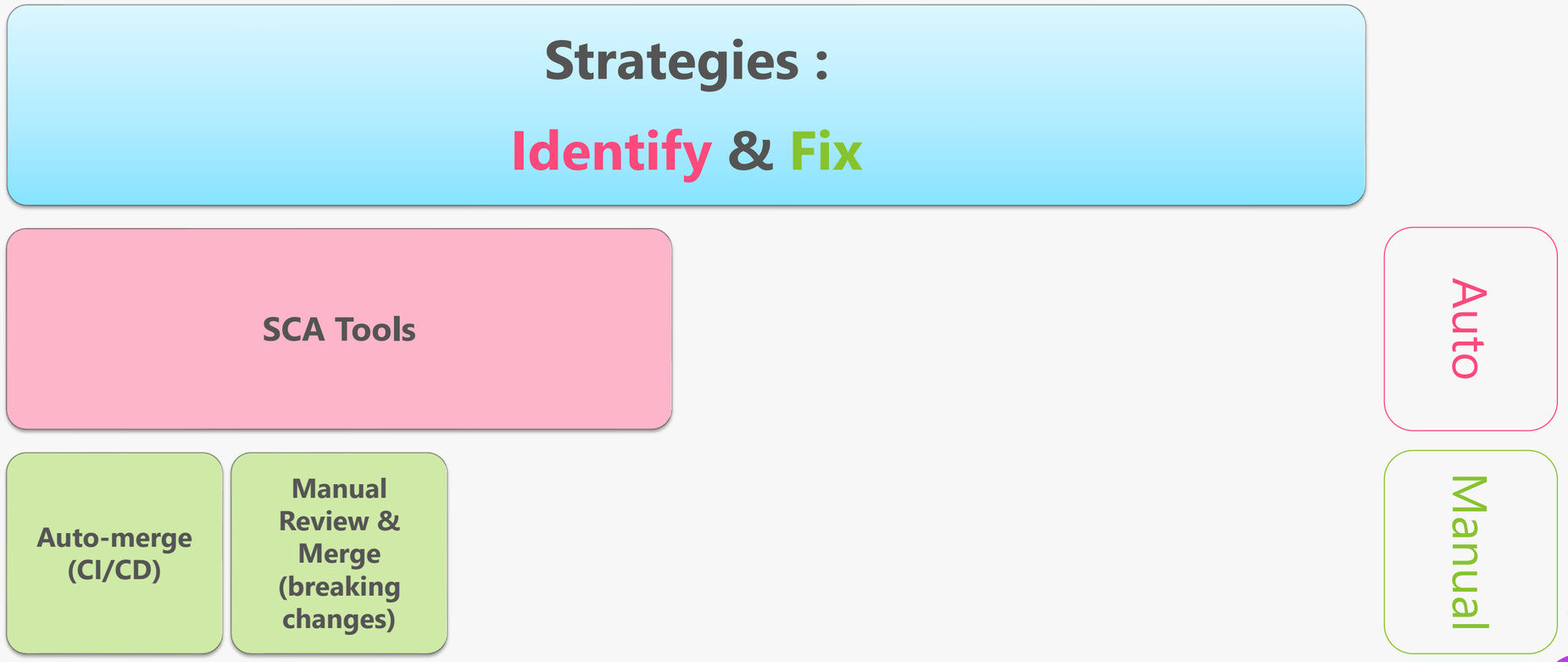
SCA Tools — Auto

Auto-merge (CI/CD) — Auto

Investigations & Findings

RQ2. Vulnerabilities in dependencies

RQ2. Patterns of developers' practices & techniques



Investigations & Findings

RQ2. Vulnerabilities in dependencies

RQ2. Patterns of developers' practices & techniques

Investigations & Findings

RQ2. Vulnerabilities in dependencies

RQ2. Patterns of developers' practices & techniques

Strategies :

Identify & Fix

SCA Tools

Manual Inspection

Auto-merge (CI/CD)

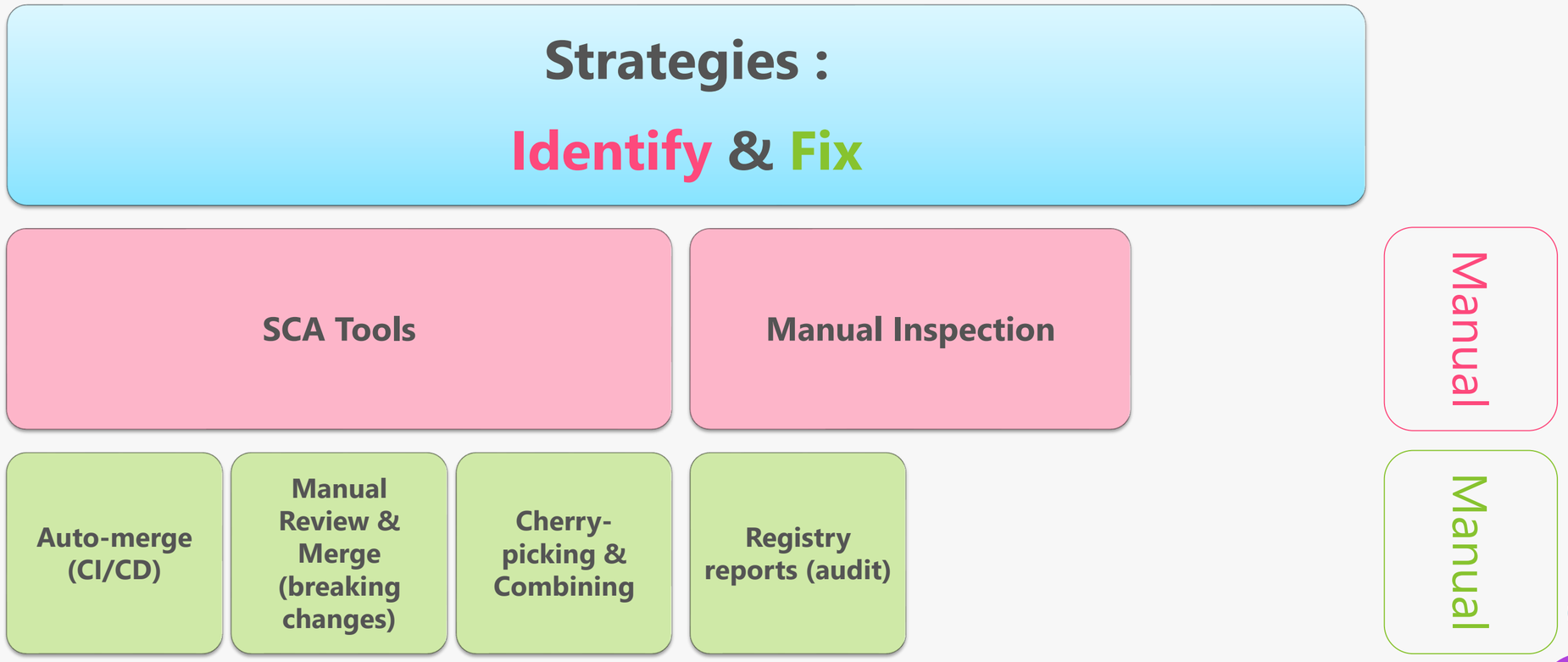
Manual Review & Merge (breaking changes)

Cherry-picking & Combining

Investigations & Findings

RQ2. Vulnerabilities in dependencies

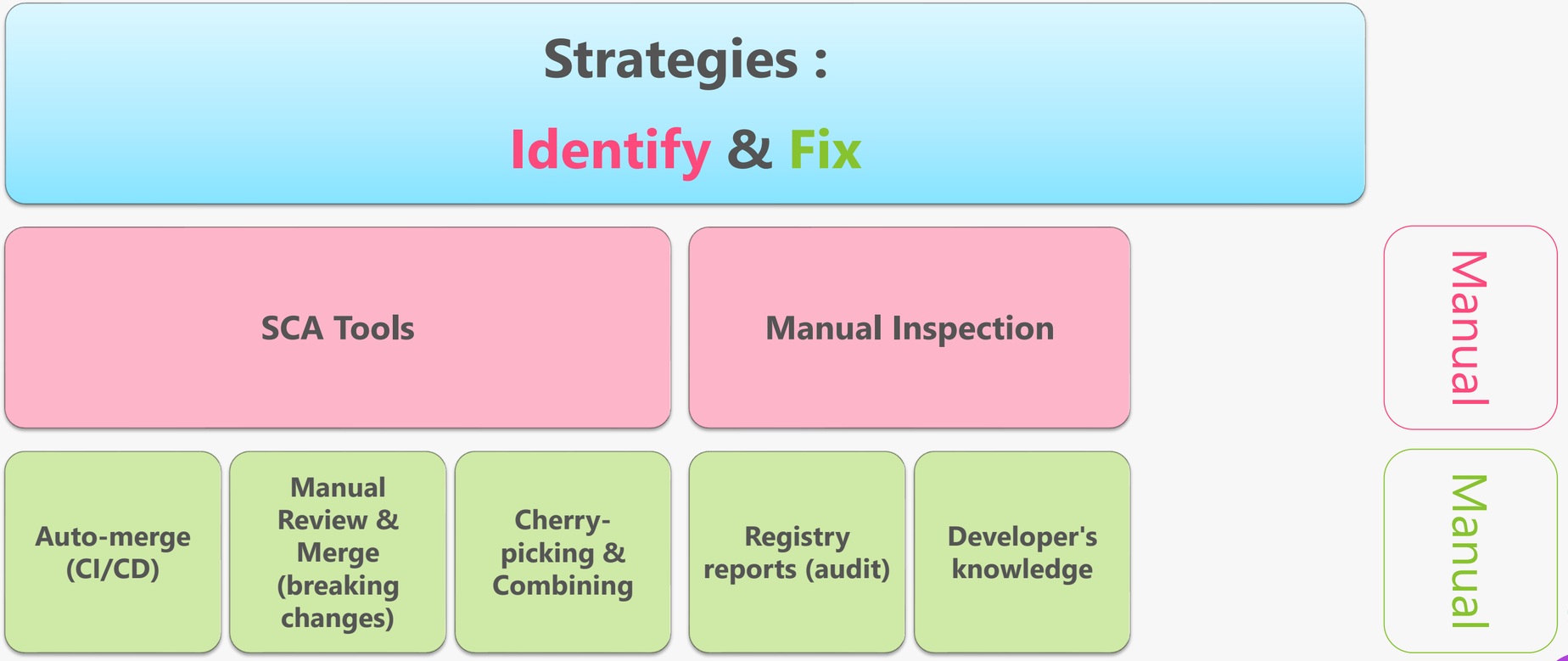
RQ2. Patterns of developers' practices & techniques



Investigations & Findings

RQ2. Vulnerabilities in dependencies

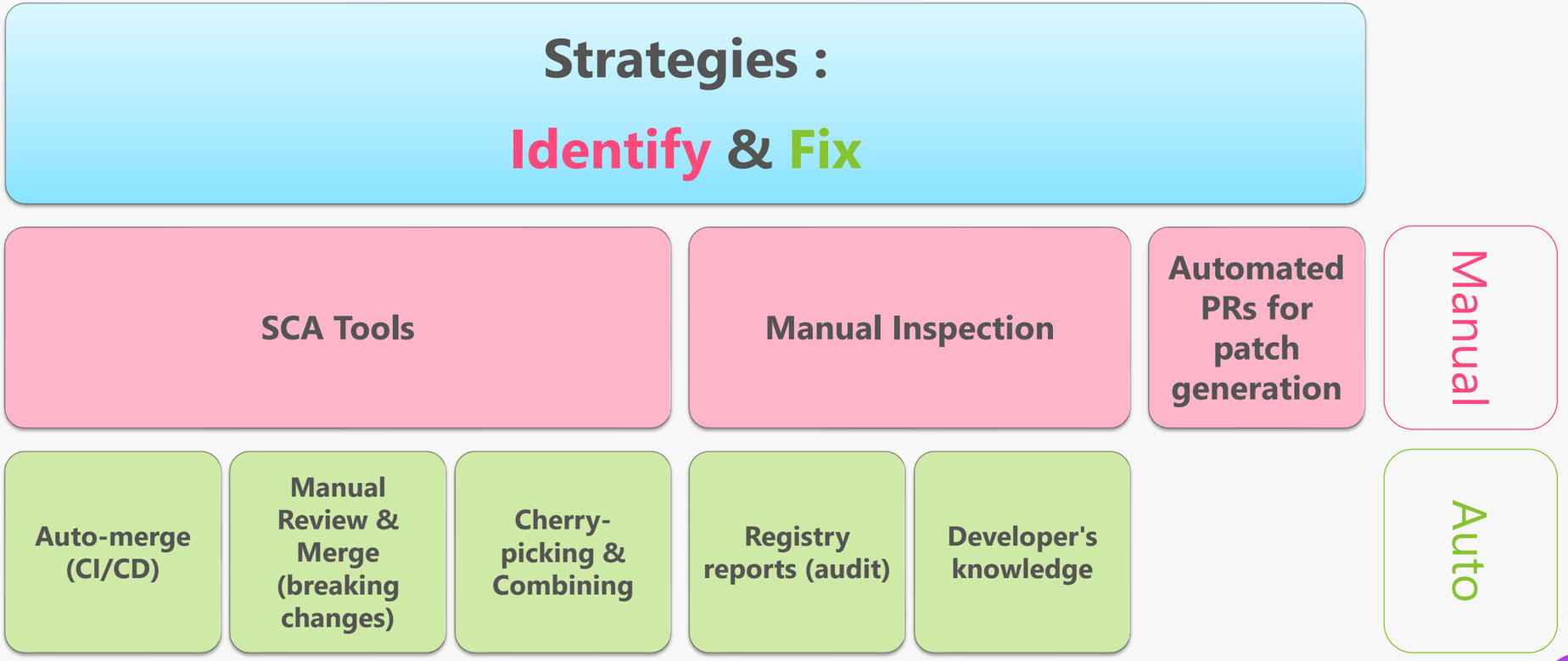
RQ2. Patterns of developers' practices & techniques



Investigations & Findings

RQ2. Vulnerabilities in dependencies

RQ2. Patterns of developers' practices & techniques



Investigations & Findings

RQ2. Vulnerabilities in dependencies

RQ2. Patterns of developers' practices & techniques

- Common actions on patches :
    1. Dependency upgrade (version, hash, transitive dependencies)
    2. Selective Dependency Resolution (version pinning, no ’^’, no ’~’)
    3. Dependency change (absence of new versions)
    4. Dependency downgrade (vulnerability-free)
    5. Dependency removal (bloated dependencies)

Investigations & Findings

RQ3. Security PRs management

RQ3.1. Receptiveness and responsiveness

Developers

- Devs are highly receptive to manual PRs; Close due to test runs and project requirement

State	Manual			Dependabot
	Contributor	*Owner*	*Total*	
Merged	16530	4070	20600 (70%)	94455 (26%)
Closed	4447	911	5358 (18%)	163837 (45%)
Open	2052	1386	3438 (12%)	105364 (29%)
Total	23029	6367	29396	363656

Distribution of security PRs per state and author

Investigations & Findings

RQ3. Security PRs management

RQ3.1. Receptiveness and responsiveness

Developers

- Devs are highly receptive to manual PRs; Close due to test runs and project requirement
- Contributors have significant impact on security fixes (Chi-squared test)

State	Manual			Dependabot
	Contributor	*Owner*	*Total*	
Merged	16530	4070	20600 (70%)	94455 (26%)
Closed	4447	911	5358 (18%)	163837 (45%)
Open	2052	1386	3438 (12%)	105364 (29%)
Total	23029	6367	29396	363656

Distribution of security PRs per state and author

Investigations & Findings

RQ3. Security PRs management

RQ3.1. Receptiveness and responsiveness

Developers

- Devs are highly receptive to manual PRs; Close due to test runs and project requirement
- Contributors have significant impact on security fixes (Chi-squared test)

Dependabot

- Merged PRs : 71% (66735/94455) manually merged, 29% auto-merged

State	Manual			Dependabot
	Contributor	*Owner*	*Total*	
Merged	16530	4070	20600 (70%)	94455 (26%)
Closed	4447	911	5358 (18%)	163837 (45%)
Open	2052	1386	3438 (12%)	105364 (29%)
Total	23029	6367	29396	363656

Distribution of security PRs per state and author

Investigations & Findings

RQ3. Security PRs management

RQ3.1. Receptiveness and responsiveness

Developers

- Devs are highly receptive to manual PRs; Close due to test runs and project requirement
- Contributors have significant impact on security fixes (Chi-squared test)

Dependabot

- Merged PRs : 71% (66735/94455) manually merged, 29% auto-merged
- Closed PRs : 8% manually closed (breaking changes, test runs fail, core dependents), 92% auto-closed (superseded, dependency updated or removed, peer requirement, & update errors)

State	Manual			Dependabot
	Contributor	*Owner*	*Total*	
Merged	16530	4070	20600 (70%)	94455 (26%)
Closed	4447	911	5358 (18%)	163837 (45%)
Open	2052	1386	3438 (12%)	105364 (29%)
Total	23029	6367	29396	363656

Distribution of security PRs per state and author

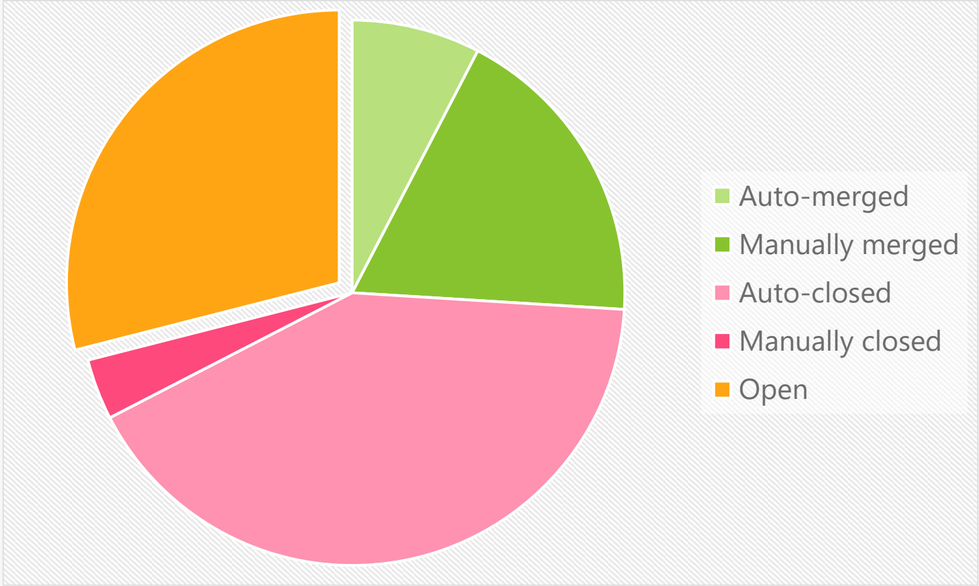
Investigations & Findings

RQ3. Security PRs management

RQ3.1. Receptiveness and responsiveness

Dependabot

- Open PRs due to :
  - Low priority for the update
  - Not enough time for review & check
  - Low severity and impact of vulnerability
  - High frequency of updates
  - Manual effort esp. when multiple repos use same dependency



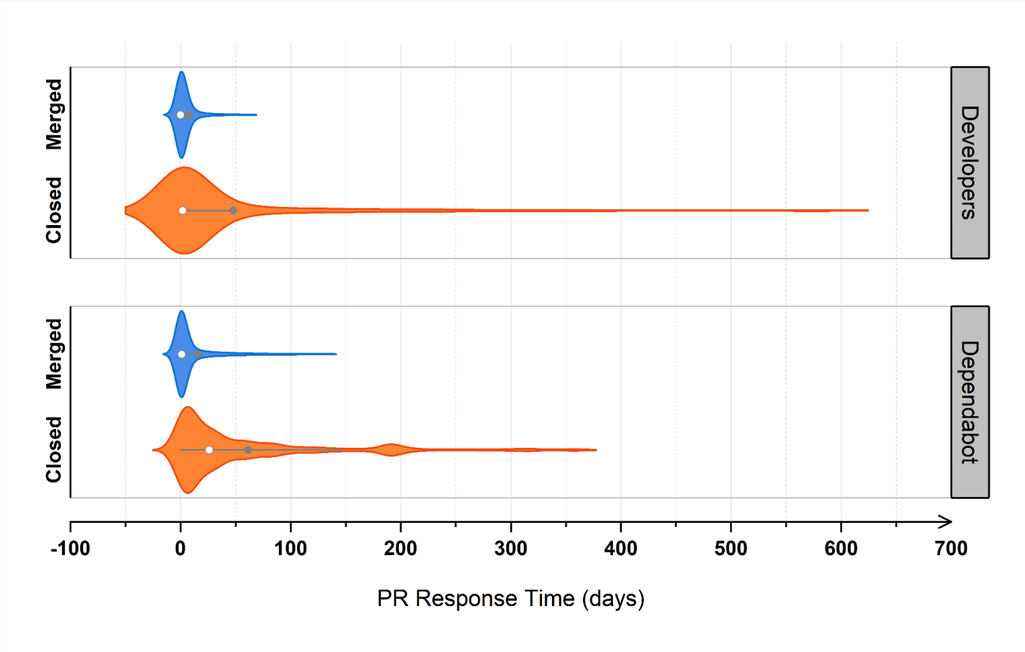
Dependabot PRs distribution per state

Investigations & Findings

RQ3. Security PRs management

RQ3.1. Receptiveness and responsiveness

- **Dependabot** PRs mostly merged in **less than 24 hours** (median: **1 day**, mean: 16) but take longer to be closed (median: 26, mean: 61 days)
- Manual PRs merged within **few hours** (median: **0 day**, mean: 7 days) and take longer to be closed (median: 2, mean: 48)



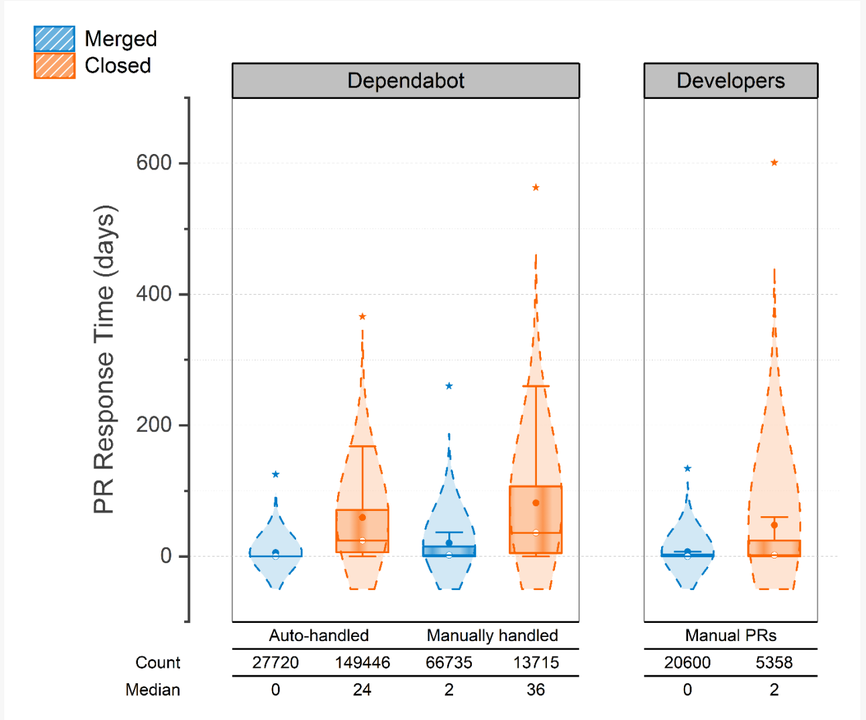
Violin-plot for the time to handle security PRs

Investigations & Findings

RQ3. Security PRs management

RQ3.1. Receptiveness and responsiveness

- PRs **merged** much **faster** than they are **closed**
- Dependabot's **auto-merge** performs best within **few minutes** (median: 0, mean: 5.7), developers merge their PRs faster (median: 0, mean: 7.3) than Dependabot's (median: 2, mean: 20)
- **Developers**' PRs **closed faster** (median: 2, mean: 48), and **Dependabot's** take **longer** whether automatically (median: 24, mean: 59) or manually (median: 36, mean: 82)



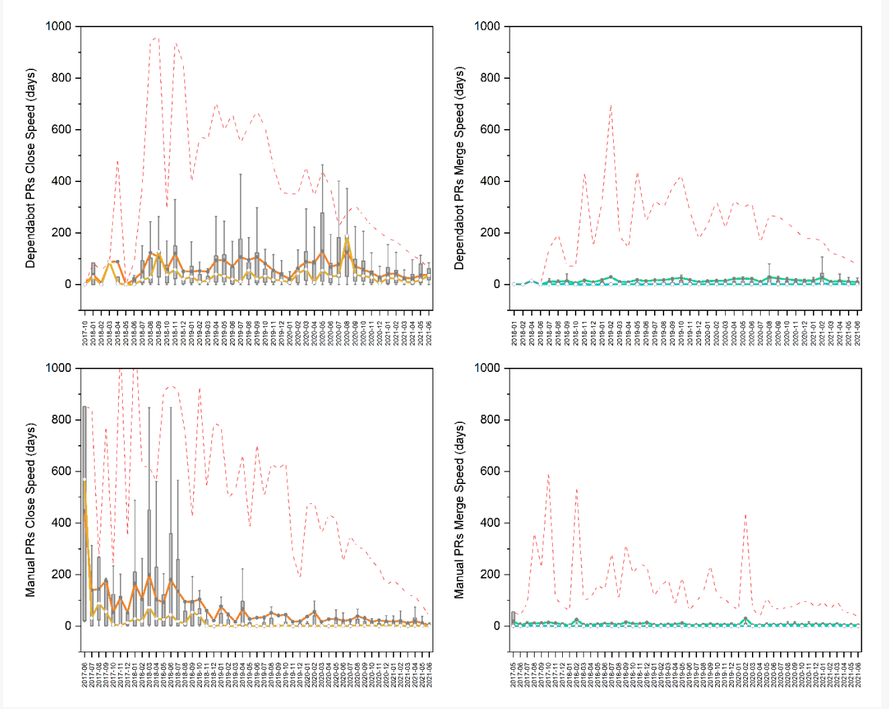
Box-plot for merge & close speed of security PRs

Investigations & Findings

RQ3. Security PRs management

RQ3.1. Receptiveness and responsiveness

- **Close speed** initially high (esp. developers'), then **decreases** over time
- Merge speed has weak variations for Dependabot & developers



Box-plot for the evolution of merge & close speed

Investigations & Findings

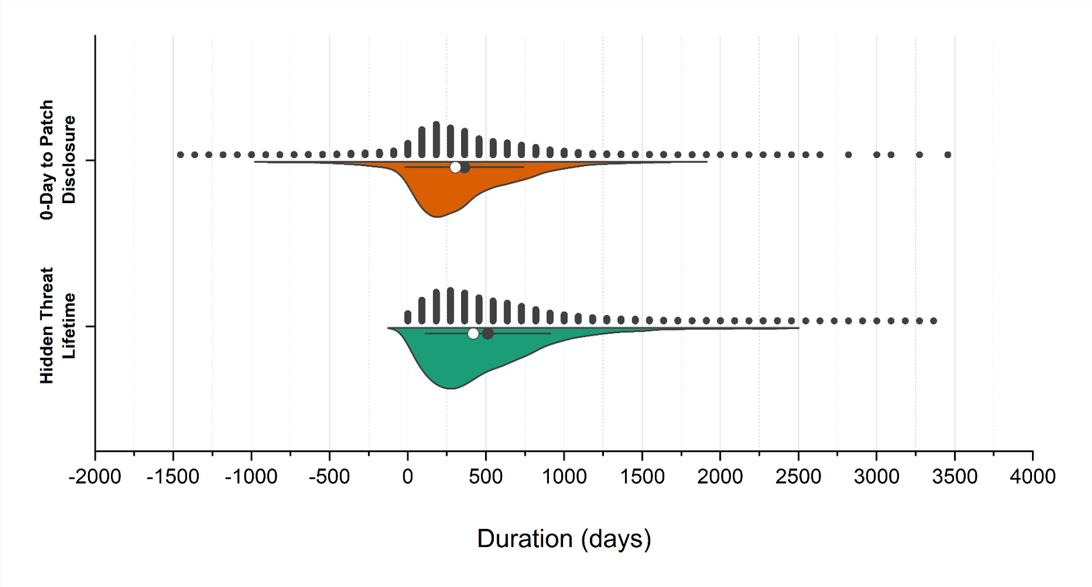
RQ3. Security PRs management

RQ3.2. Threat lifetime & fix delay

- Threats persist unknown in GitHub for **512 days** on avg. (median: 419 days)

  → Huge window of exposure **!**

- Patches disclosed after **362 days** on avg. (median: 305 days) from 0-day (manual expert inspection)
- Small gap between two metrics : fixes are made quickly in GitHub soon after disclosing patches in CVE databases



Violin-plot for threat lifetime & fix delay

Investigations & Findings

RQ3. Security PRs management

RQ3.2. Threat lifetime & fix delay

- Vulnerabilities with **serious severity** levels are the **most occurring** on GitHub
- Vulnerabilities with **highest severity** levels (critical) have **quickest fixes** (priority)

Severity Level	# PRS	Average Threat Lifetime (days)
High	6587	540
Moderate	2607	481
Critical	1653	427
Medium	691	653
Low	502	595

Threat lifetime based on the severity level

Investigations & Findings

RQ3. Security PRs management

RQ3.2. Threat lifetime & fix delay

- # PRs decreases as the update level gets higher
  - **Most fixes** are performed on **patch level**
- ATL increases as the update level gets higher
  - **Major updates** take the **longest** to be released (huge changes)

Update level	# PRs	ATL (days)
Patch	8316	435
Minor	4808	580
Major	713	957

Threat lifetime based on the update level

Investigations & Findings

RQ3. Security PRs management

RQ3.3. Most exploited vulnerabilities

- Most common : **Prototype Pollution**
  - ➢ Simple logic, targets *npm*, leads to subsequent attacks
- Highest ATL : **Cross-Site Scripting (XSS)**
  - ➢ Harder to manually inspect, time to implement fixes
- Lowest ATL : Usage of Broken / Risky Cryptographic Algorithms
  - ➢ Easier to pinpoint, predefined fixes

Vulnerability (Malicious Behavior)	# PRs	ATL
Prototype Pollution	4525	478
Regular Expression Denial of Service	1627	561
Denial of Service	539	543
Signature Malleabillity	435	499
ReDoS and Prototype Pollution	430	353
XSS Vulnerability	299	629
Command Injection	280	284
Path Traversal	243	536
Arbitrary Code Execution	236	375
Resource Allocation Without Throttling	225	276
Using Risky Cryptographic Algorithm	181	197
Potential Memory Exposure	174	619
OS Command Injection	172	440
Remote Memory Exposure	150	623
Arbitrary File Overwrite	143	442
Possible Information Leak / Session Hijack	126	464
Remote Code Execution	117	368

Most exploited vulnerabilities

Investigations & Findings

RQ4. Merge decision & Merge speed

RQ4. Factors correlating with the acceptance and fast merges

❑ **Merge decision :** Acceptance supported by

- Descriptions of small size (# comments, discussion) => Dependabot communication, more changes
- Collaboration (# assignees)
- Less changes (# additions, # changed files) => breaking changes, refactoring effort + tests + reviews
- Repository characteristics (activity, maturity)
- Update level & severity

Feature	Coef.	z	p-value
# comments	-2.1931	-202.477	< 0.001
discussion_size	-0.7786	-97.675	< 0.001
# assignees	0.2704	42.805	< 0.001
# additions	-0.2018	-36.532	< 0.001
# changed_files	0.1328	19.490	< 0.001
recent_activity	-0.0782	-14.361	< 0.001
# open_issues	0.0577	7.637	< 0.001
age	0.0193	3.492	< 0.001
# commits	-0.0125	-1.922	0.055
# watchers	-0.0104	-1.632	0.103
size	-0.0039	-0.682	0.495
patch_level	-	2105.170	< 0.001
severity	-	224.866	< 0.001

Tests results on merge decision for Dataset (2)

Investigations & Findings

RQ4. Merge decision & Merge speed

RQ4. Factors correlating with the acceptance and fast merges

❑ **Merge decision :** Acceptance supported by

- Descriptions of small size (# comments, discussion) => Dependabot communication, more changes
- Collaboration (# assignees)
- Less changes (# additions, # changed files) => breaking changes, refactoring effort + tests + reviews
- Repository characteristics (activity, maturity)
- Update level & severity
- Developer's experience, contribution, & association (owner vs. contributor)

Feature	Coef.	z	p-value
recent_activity	-14.0161	-65.587	< 0.001
# assignees	0.4259	22.609	< 0.001
discussion_size	-0.2729	-17.569	< 0.001
experience	0.2142	12.903	< 0.001
# watchers	-0.1185	-6.617	< 0.001
# additions	0.0741	4.417	< 0.001
# public_repos_gists	-0.0606	-3.708	< 0.001
age	0.0573	3.570	< 0.001
# comments	-0.0576	-3.568	< 0.001
size	-0.0413	-2.824	0.005
# changed_files	0.0202	1.165	0.244
# followers	-0.0137	-0.712	0.476
# open_issues	-0.0114	-0.676	0.499
# commits	0.0070	0.396	0.692
author_association	-	14.792	< 0.001

Tests results on merge decision for Dataset (3)

Investigations & Findings

RQ4. Merge decision & Merge speed

RQ4. Factors correlating with the acceptance and fast merges

❑ **Merge speed :** Fast merge supported by

- PR changes (# commits, # changed files) => code reviews, test runs, refactoring effort
- Efficient communication (# comments, discussion) => Dependabot actions, developer's feedback
- Project characteristics (maturity, size) => adaptability, more contributors
- Update level & severity

Feature	Coef.	z	p-value
# commits	1.6082	9.898	< 0.001
# changed_files	1.2759	7.728	< 0.001
# comments	0.7519	4.653	< 0.001
discussion_size	0.6749	4.160	< 0.001
age	-0.6404	-3.910	< 0.001
size	-0.6161	-3.276	0.001
# additions	0.2599	1.597	0.110
recent_activity	-0.2039	-1.263	0.207
# open_issues	-0.2479	-1.085	0.278
# assignees	-0.1043	-0.647	0.517
# watchers	-0.0785	-0.383	0.701
severity	-	16.791	0.002
patch_level	-	8.498	0.014

Tests results on merge speed for Dataset (2)

Investigations & Findings

RQ4. Merge decision & Merge speed

RQ4. Factors correlating with the acceptance and fast merges

❑ **Merge speed :** Fast merge supported by

- PR changes (# commits, # changed files) => code reviews, test runs, refactoring effort
- Efficient communication (# comments, discussion) => Dependabot actions, developer's feedback
- Project characteristics (maturity, size) => adaptability, more contributors
- Update level & severity
- Developer's workload, contribution & association

Feature	Coef.	z	p-value
# comments	4.1299	18.532	< 0.001
# public_repos_gists	3.4833	15.432	< 0.001
age	2.3858	10.257	< 0.001
# commits	0.8149	3.610	< 0.001
discussion_size	0.7945	3.251	0.001
size	-0.4939	-2.217	0.027
# watchers	-0.5248	-2.104	0.035
experience	-0.2772	-1.158	0.247
# changed_files	0.2295	0.983	0.326
# followers	-0.1798	-0.814	0.416
recent_activity	0.1535	0.714	0.475
# additions	0.1190	0.527	0.598
# open_issues	-0.0707	-0.282	0.778
# assignees	0.0215	0.093	0.926
author_association	-	7.903	< 0.001

Tests results on merge speed for Dataset (3)

Tool Designers

Concerns

- Overwhelming alerts, pollute project history & notifications
- Breaking changes, manual effort
- Frequency of updates, time to merge
- Threat lifetime, unknown vulnerabilities
- Tool adoption

Alternatives

- ✓ Improve bot-human interaction (combine PRs w/ edits & selection)
- ✓ Locate code fragments that require refactoring
- ✓ Support auto-merge w/ restriction options (update level)
- ✓ Effective & efficient tools that rely on available data
- ✓ Features : efficiency (configuration + integration), accessibility, adaptivity, comprehensibility & support

Implications

Repository Owners / Maintainers

Concerns

- Not using tools / bots to handle vulnerabilities in dependencies
- Fix delay after discovering vulnerabilities
- Negative hidden threat lifetime
- Common attacks

Alternatives

- ✓ Maintain regular level of awareness (inspection & audits, security reports, vulnerability DBs, advisories, etc.)
- ✓ Narrow window of exposure (e.g., suggest substitute packages in absence of safer versions)
- ✓ React to fix disclosures (disable/remove vulnerable versions, inform users about threats during installation)
- ✓ List of most exploited vulnerabilities (e.g., security evaluation like *OWASP Top Ten*)

Implications

Developers

Concerns

- Factors impact handling security PRs
- Auto-closed PRs (superseded)
- Bloated dependencies

Alternatives

- ✓ Be concise and *make a long story short* (consider description size and # comments)
- ✓ React to open PRs and not ignore them for too long
- ✓ Keep dependency graph clean from redundant and unused dependencies

Implications

Contributions

Knowledge & Insights

Adoption of bots in fixing vulnerabilities in dependencies, developers' patterns to handle SSCAs, threat lifetime, & management of security PRs

Dataset & Reference

Dataset of 9,288,808 PRs-related issues in 979,179 projects for more than 10 PLs, for general purposes (security, pull-based, etc.)

Data Collection Pipeline

Pipeline to extract issues, pull requests, repositories, commits and users' data from GitHub

Thank You !

Any Questions **?**

• • •

Bibliography

- [1] Ghaffarian, Seyed Mohammad, and Hamid Reza Shahriari. "Software vulnerability analysis and discovery using machine-learning and data-mining techniques: A survey." ACM Computing Surveys (CSUR) 50, no. 4 (2017): 1-36.
- [2] GREENBERG, A., 2021. What Is a Supply Chain Attack?. [online] Wired. Available at: https://www.wired.com/story/hacker-lexicon-what-is-a-supply-chain-attack [Accessed 6 July 2021].
- [3] Alon, Uri, Meital Zilberstein, Omer Levy, and Eran Yahav. "code2vec: Learning distributed representations of code." Proceedings of the ACM on Programming Languages 3, no. POPL (2019): 1-29.
- [4] Fang, Chunrong, Zixi Liu, Yangyang Shi, Jeff Huang, and Qingkai Shi. "Functional code clone detection with syntax and semantics fusion learning." In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 516-527. 2020.
- [5] Sachdev, Saksham, Hongyu Li, Sifei Luan, Seohyun Kim, Koushik Sen, and Satish Chandra. "Retrieval on source code: a neural code search." In Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, pp. 31-41. 2018.
- [6] Hashemi, Hashem, and Ali Hamzeh. "Visual malware detection using local malicious pattern." Journal of Computer Virology and Hacking Techniques 15, no. 1 (2019): 1-14.
- [7] Keller, Patrick, Laura Plein, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. "What You See is What it Means! Semantic Representation Learning of Code based on Visualization and Transfer Learning." arXiv preprint arXiv:2002.02650 (2020).
- [8] Ragkhitwetsagul, Chaiyong, Jens Krinke, and Bruno Marnette. "A picture is worth a thousand words: Code clone detection based on image similarity." In 2018 IEEE 12th International workshop on software clones (IWSC), pp. 44-50. IEEE, 2018.
- [9] Chess, B., F. DeQuan Lee, and J. West. "Attacking the build through cross-build injection: how your build process can open the gates to a trojan horse." (2007): 24-25.
- [10] Eggers, Shannon. "A novel approach for analyzing the nuclear supply chain cyber-attack surface." Nuclear Engineering and Technology 53, no. 3 (2021): 879-887.
- [11] Ohm, Marc, Henrik Plate, Arnold Sykosch, and Michael Meier. "Backstabber's knife collection: A review of open source software supply chain attacks." In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 23-43. Springer, Cham, 2020.
- [12] Duan, Ruian, Omar Alrawi, Ranjita Pai Kasturi, Ryan Elder, Brendan Saltaformaggio, and Wenke Lee. "Measuring and preventing supply chain attacks on package managers." (2020).
- [13] Vu, Duc Ly, Ivan Pashchenko, Fabio Massacci, Henrik Plate, and Antonino Sabetta. "Towards using source code repositories to identify software supply chain attacks." In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, pp. 2093-2095. 2020.
- [14] Ohm, Marc, Arnold Sykosch, and Michael Meier. "Towards detection of software supply chain attacks by forensic artifacts." In Proceedings of the 15th international conference on availability, reliability and security, pp. 1-6. 2020.
- [15] Garrett, Kalil, Gabriel Ferreira, Limin Jia, Joshua Sunshine, and Christian Kästner. "Detecting suspicious package updates." In 2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER), pp. 13-16. IEEE, 2019.
- [16] Ohm, Marc, Lukas Kempf, Felix Boes, and Michael Meier. "If You've Seen One, You've Seen Them All: Leveraging AST Clustering Using MCL to Mimic Expertise to Detect Software Supply Chain Attacks." arXiv e-prints (2020): arXiv-2011.
- [17] Lin, Guanjun, Jun Zhang, Wei Luo, Lei Pan, and Yang Xiang. "POSTER: Vulnerability discovery with function representation learning from unlabeled projects." In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 2539-2541. 2017.
- [18] Russell, Rebecca, Louis Kim, Lei Hamilton, Tomo Lazovich, Jacob Harer, Onur Ozdemir, Paul Ellingwood, and Marc McConley. "Automated vulnerability detection in source code using deep representation learning." In 2018 17th IEEE international conference on machine learning and applications (ICMLA), pp. 757-762. IEEE, 2018.
- [19] Lin, Guanjun, Wei Xiao, Jun Zhang, and Yang Xiang. "Deep learning-based vulnerable function detection: A benchmark." In International Conference on Information and Communications Security, pp. 219-232. Springer, Cham, 2019.
- [20] Zhou, Yaqin, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks." Advances in neural information processing systems 32 (2019).
- [21] Wang, Huanting, Guixin Ye, Zhanyong Tang, Shin Hwei Tan, Songfang Huang, Dingyi Fang, Yansong Feng, Lizhong Bian, and Zheng Wang. "Combining graph-based learning with automated data collection for code vulnerability detection." IEEE Transactions on Information Forensics and Security 16 (2020): 1943-1958.

Appendix

Software Supply Chain Attack

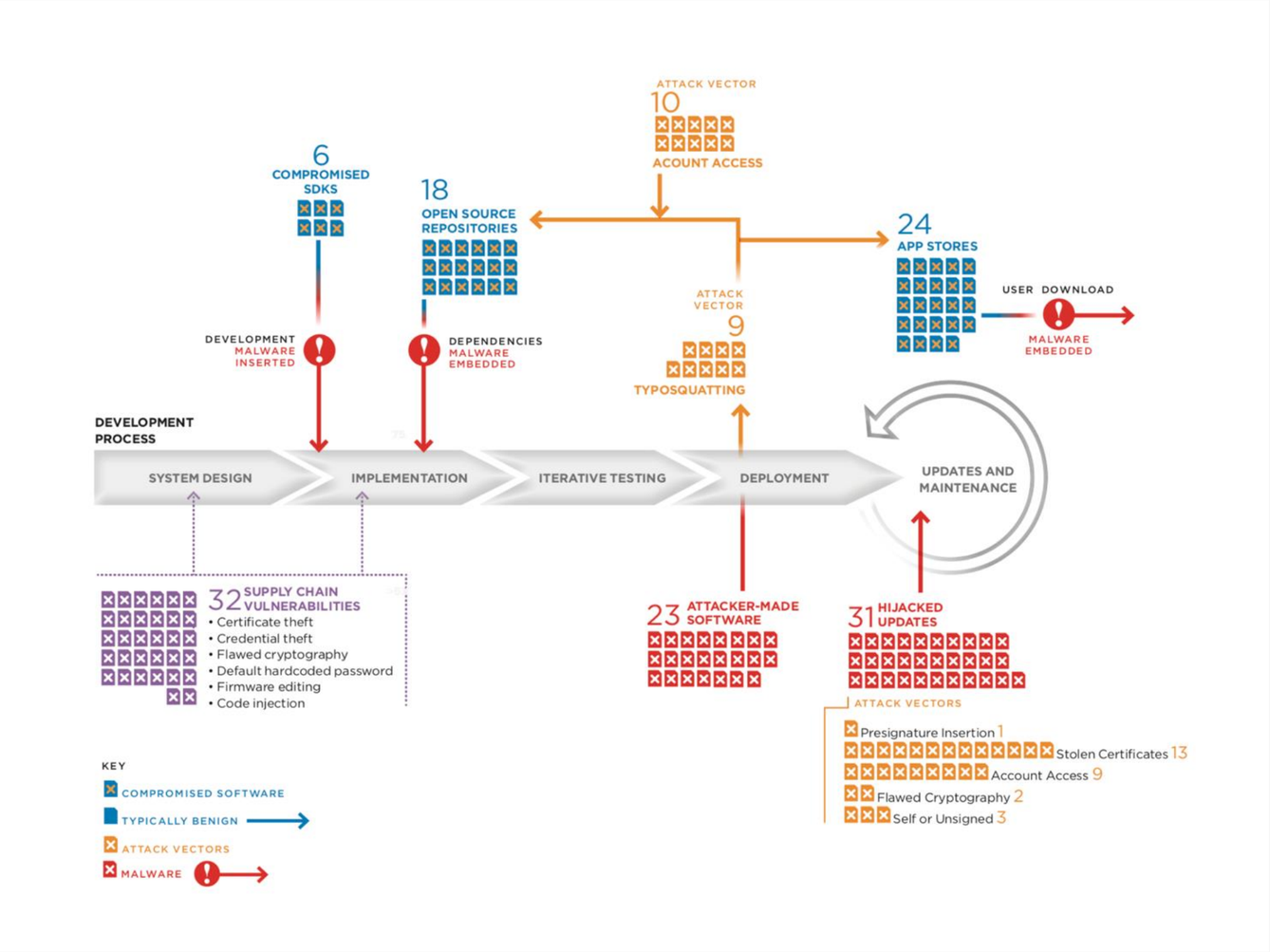
What is a Software Supply Chain Attack (SSCA)?

A technique in which an adversary slips **malicious code** or even a **malicious component** into a trusted piece of software or hardware. By compromising a single supplier, attackers can hijack the **distribution system** to turn any application into **Trojan horse** [2].

- Attack vectors (Strategies)
  - Social engineering
  - Typo-squatting (E.g., **jellyfish** and **jellyfish**)
  - Combo-squatting (E.g., **python-ftp** and **pyftpdlib**)
  - Etc.
- Purpose
  - Stealing credentials
  - Data exfiltration
  - Cryptocurrency mining
  - Etc.

Appendix

Software Supply Chain Life Cycle



https://www.atlanticcouncil.org/in-depth-research-reports/report/breaking-trust-shades-of-crisis-across-an-insecure-software-supply-chain/