# Diplomat

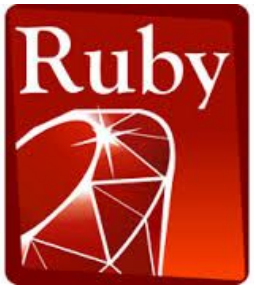
## Using Delegations to Protect Community Repositories

Trishank Karthik Kuppusamy, Santiago Torres-Arias, Vladimir Diaz, Justin Cappos

NYU | TANDON SCHOOL OF ENGINEERING

# Community repositories

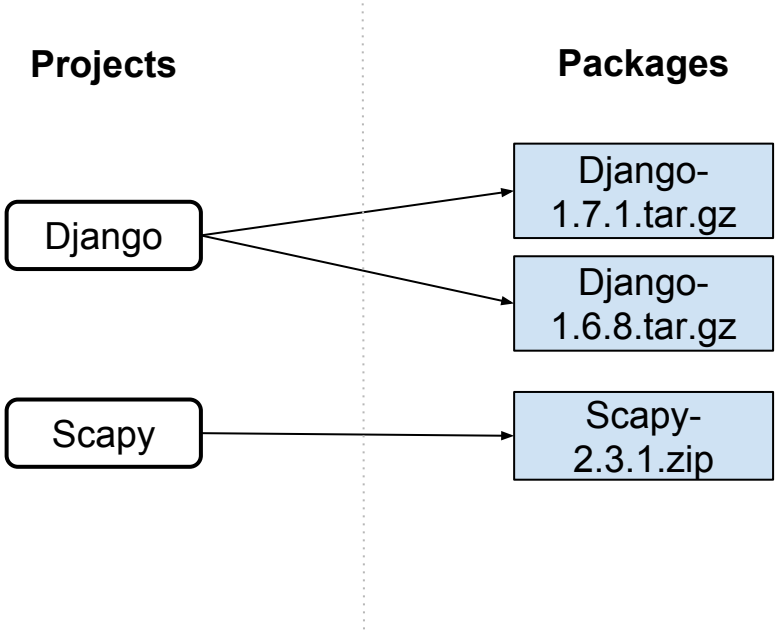
# Community repositories: examples

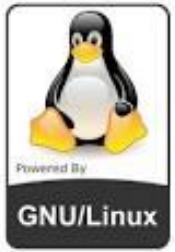



# Community repositories: definition

- All software by 3rd-party **developers**.
- Software organized by **projects**.
- A project may release many **packages**.
- > 10K projects, 100K packages (e.g., on PyPI).
- A **new** project/package added **every few minutes** (e.g., on PyPI).



# Great! What is the problem?







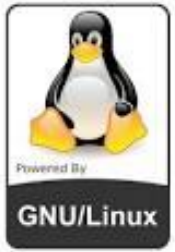






Windows


# What do these organizations share?













Windows


# Users were attacked via software updates.

# Repository compromise: impact

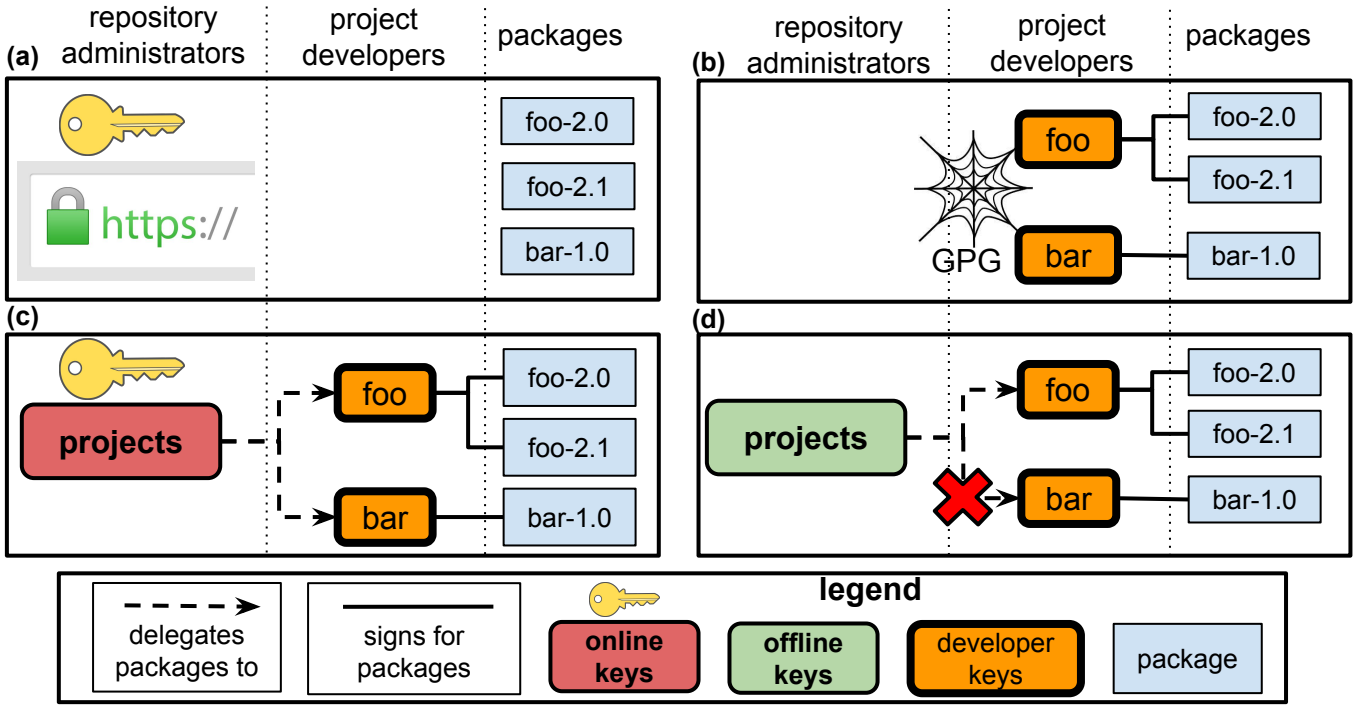
- High impact: **malware** can be **installed** by **millions** of unsuspecting **users**.
- Microsoft Windows Update (2012): Flame malware spread via MitM attack.
- South Korea cyberattack (2013): **$756,000,000 USD** in economic damage due to malware spread partly via automatic software updates.

# Goal: compromise-resilience

- Cannot prevent a compromise.
- But must at least limit its impact.
- Attackers can compromise as few users as possible.

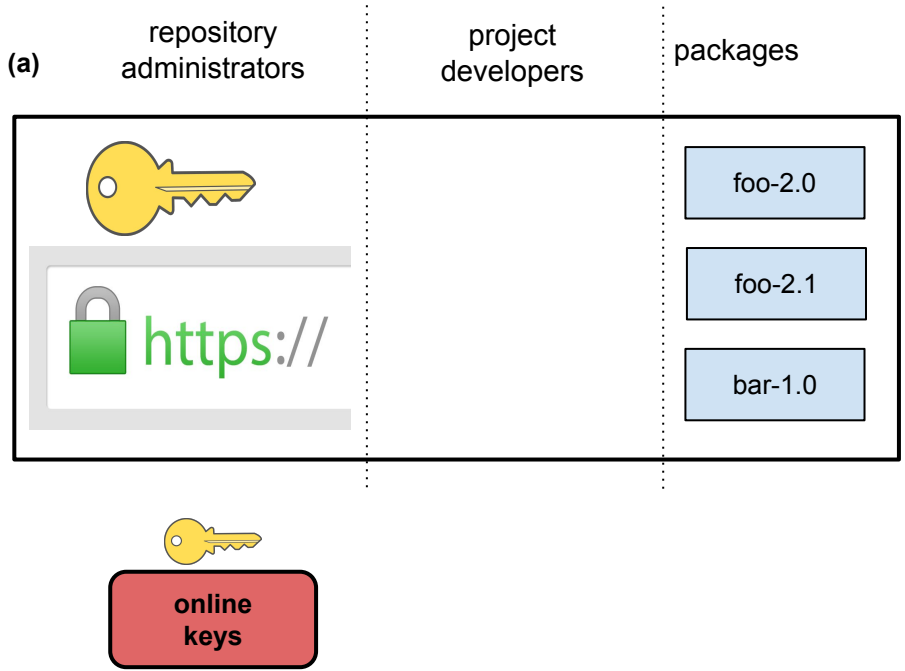
# Previous security systems

# Overview


(a)
repository administrators
project developers
packages
https://
foo-2.0
foo-2.1
bar-1.0
(b)
repository administrators
project developers
packages
foo
bar
GPG
foo-2.0
foo-2.1
bar-1.0
(c)
projects
foo
bar
foo-2.0
foo-2.1
bar-1.0
(d)
projects
foo
bar
foo-2.0
foo-2.1
bar-1.0
legend
delegates packages to
signs for packages
online keys
offline keys
developer keys
package

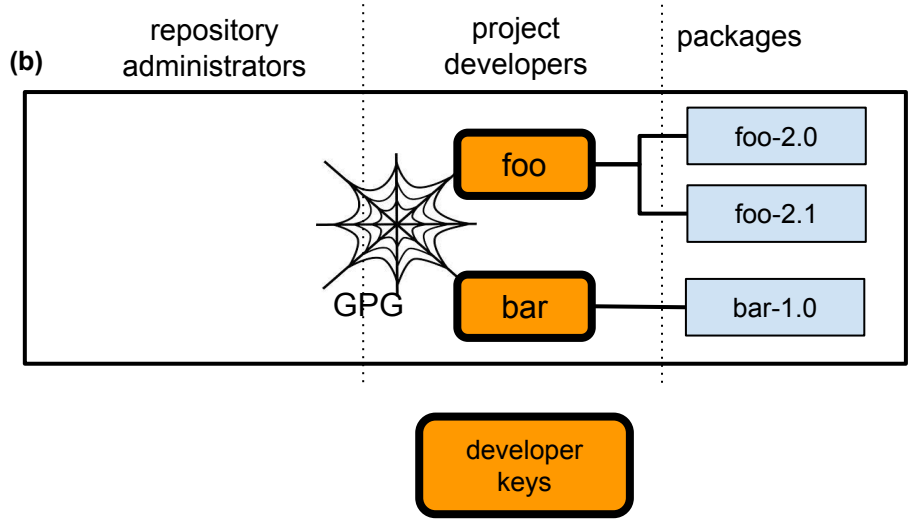
# (a) Repos sign packages with online keys

- Repositories sign packages with a transport mechanism (e.g., TLS, CUP).
- Signing private keys kept **online**.
- Not compromise-resilient.



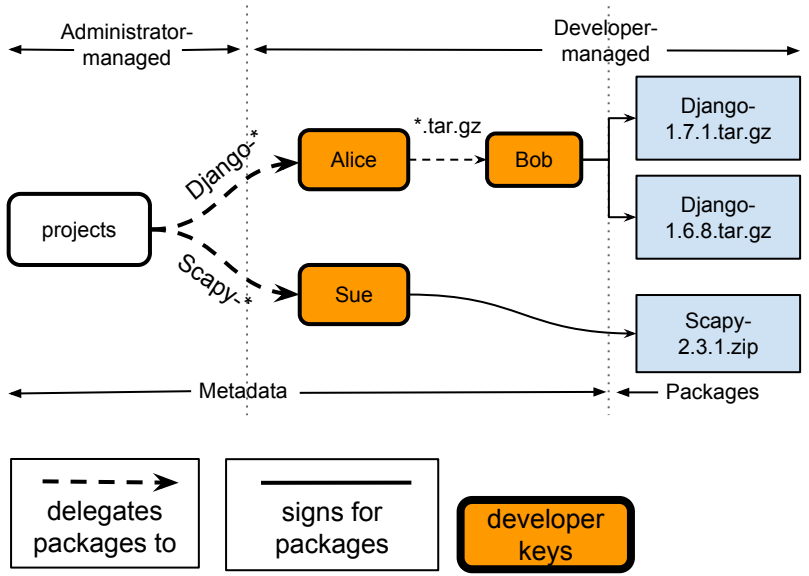
# (b) Devs sign packages with offline keys

- Developers sign packages with (e.g., GPG, RSA) **offline** private keys.
- Compromise-resilient!
- But, unusable key distribution & revocation.



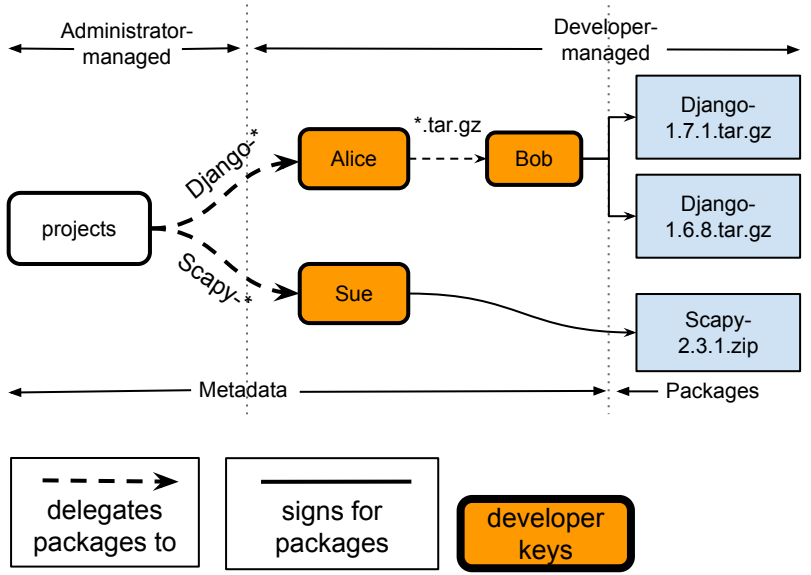
# Interlude: Delegations with TUF

- TUF (our previous system) uses **delegations**.
- Bind public keys to projects.
- “Survivable key compromise in software update systems,” Samuel et. al., CCS 2010.



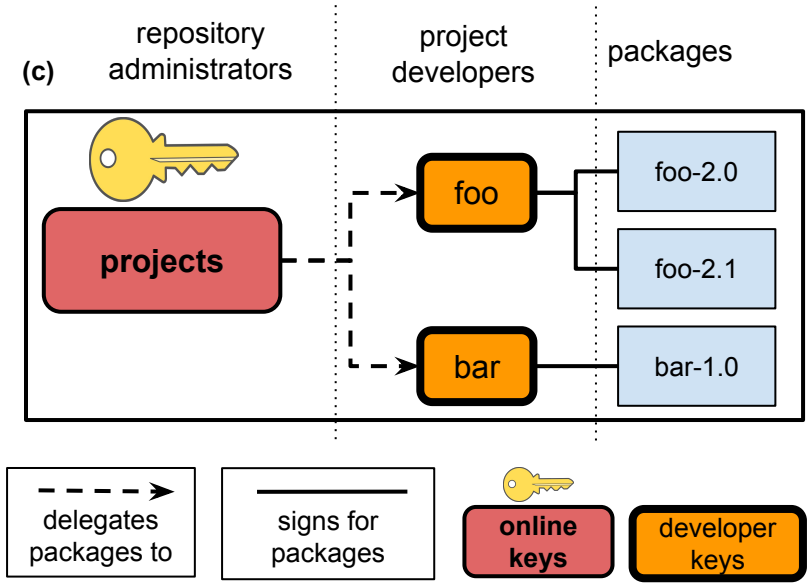
# Interlude: Delegations with TUF

- How to **sign** delegations?
- Use **online** or **offline** keys?



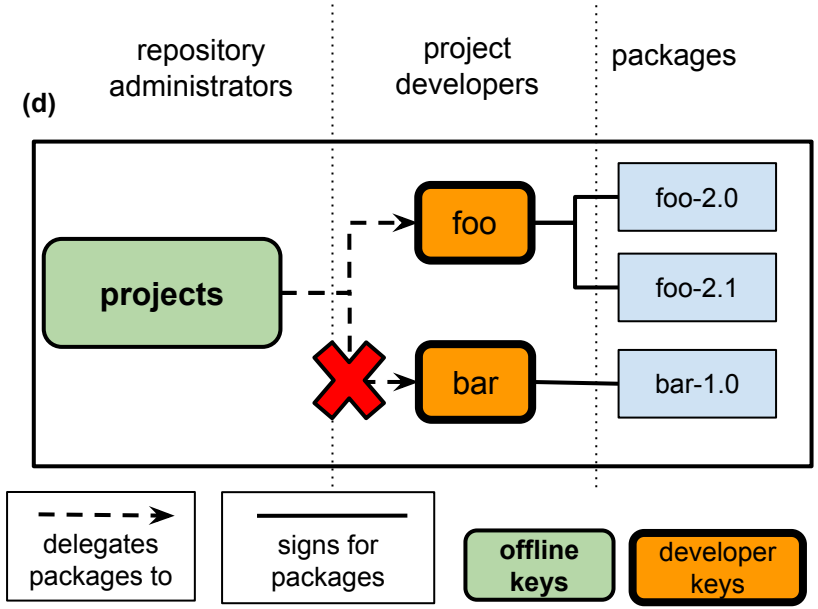
# (c) Repos delegate projects with online keys

- Repositories delegate projects to developers with **online** keys.
- Immediate project registration!
- But, not compromise-resilient.



# (d) Admins delegate projects with offline

- Administrators delegate projects to developers with **offline** keys.
- Compromise-resilient!
- But, no immediate project registration.



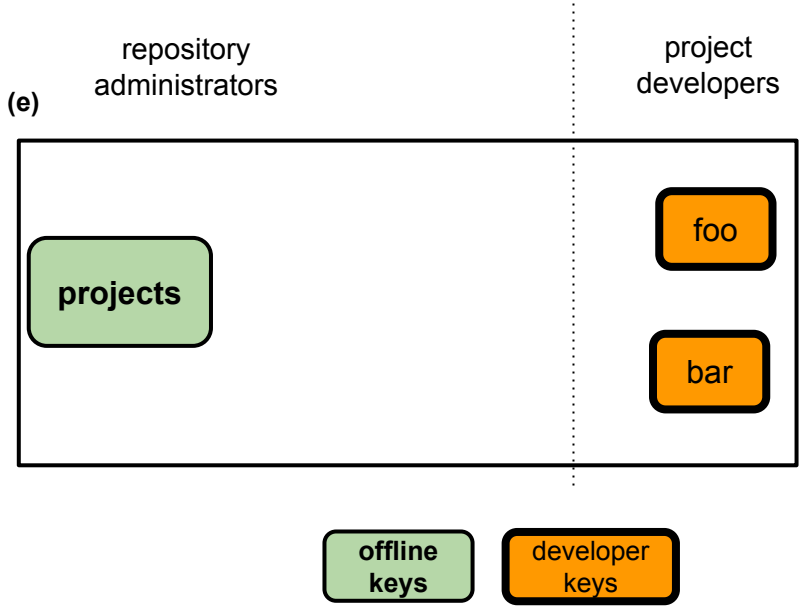
# Either or

- Previous systems force community repositories to choose **either** compromise-resilience, **or** immediate project registration.

# Diplomat: a new security system

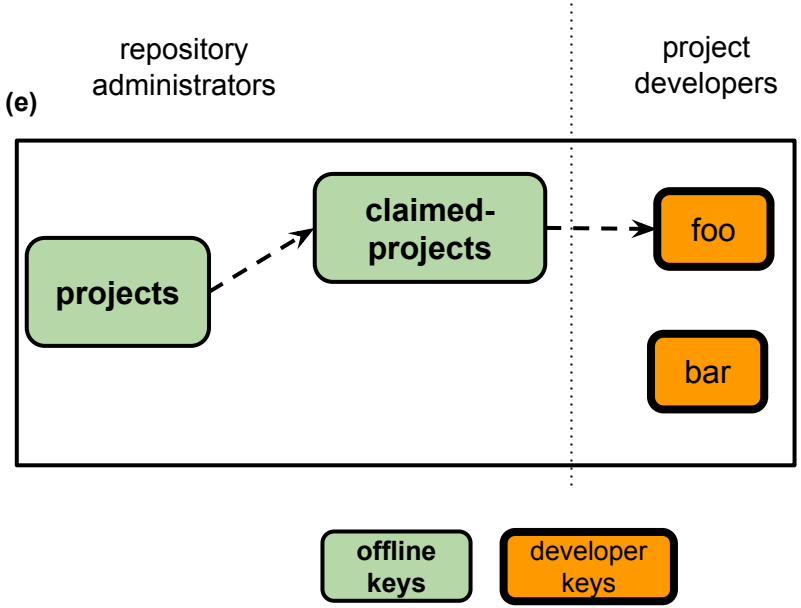
# New idea

- What if….



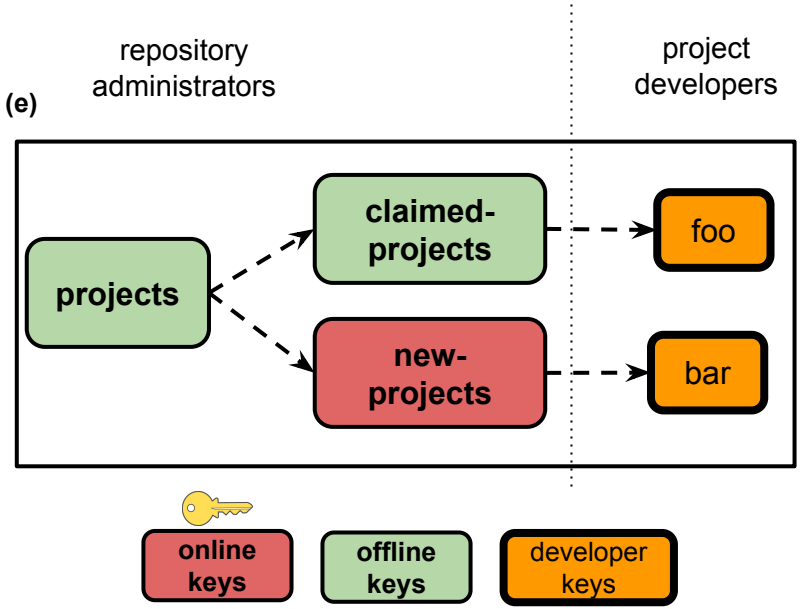
# New idea: a middle way?

- What if….
- Sign delegations to **most** projects with **offline** keys…



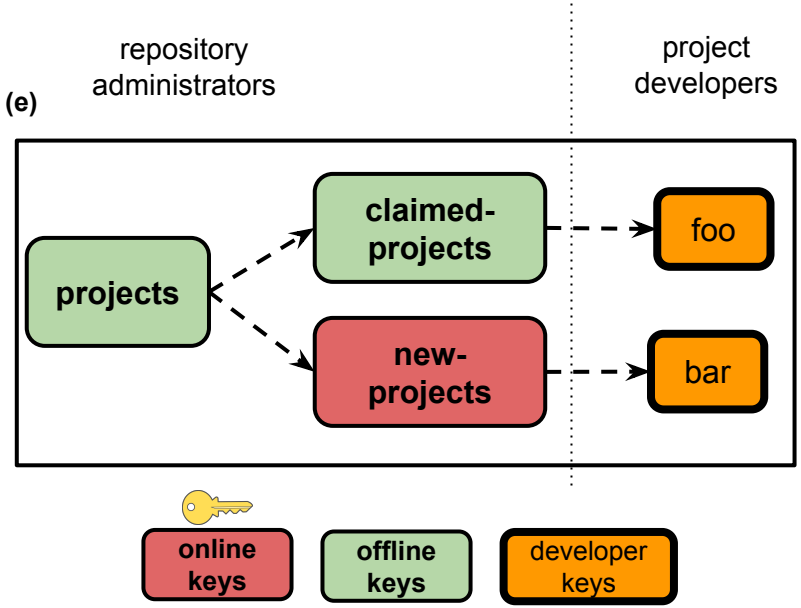
# New idea: a middle way?

- What if….
- Sign delegations to **most** projects with **offline** keys.
- Sign only delegations to **new** projects with **online** keys.



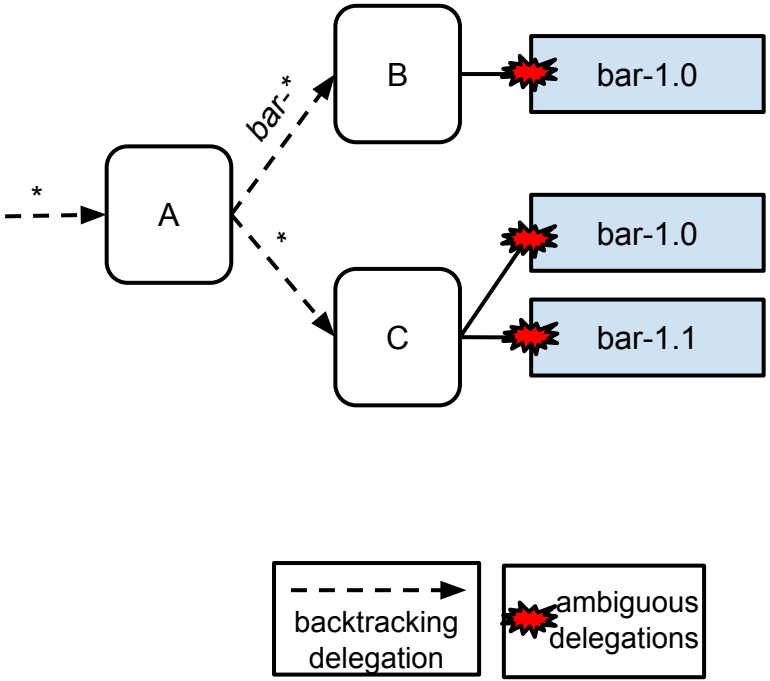
# New idea: a middle way?

- **Both** compromise-resilience and immediate project registration via **multiple** delegations.



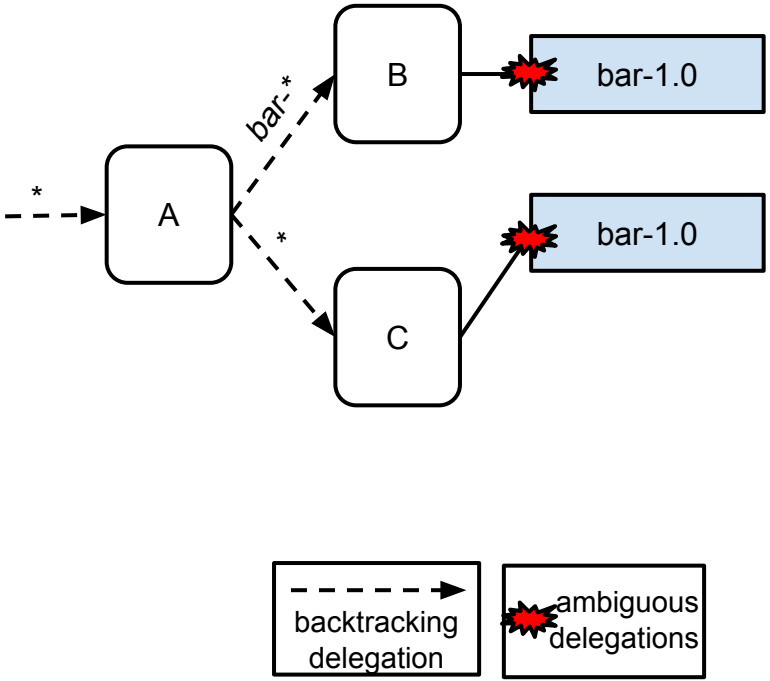
# Ambiguous delegations

- What if A delegates the bar project to **both** B **and** C?
- Should a package manager trust B **or** C for the bar project?



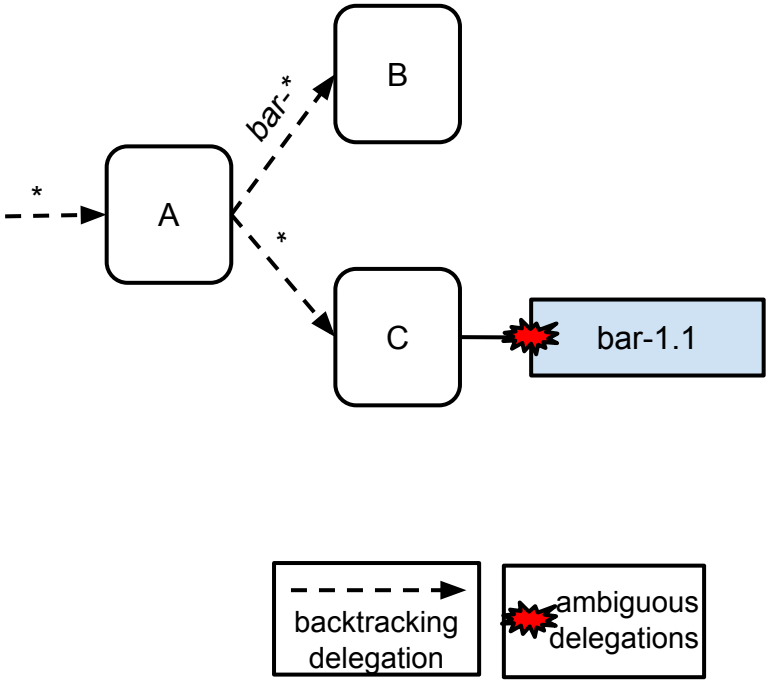
# Ambiguous delegations: ordering problem

- What if **both** B **and** C sign the **same** bar-1.0 package?



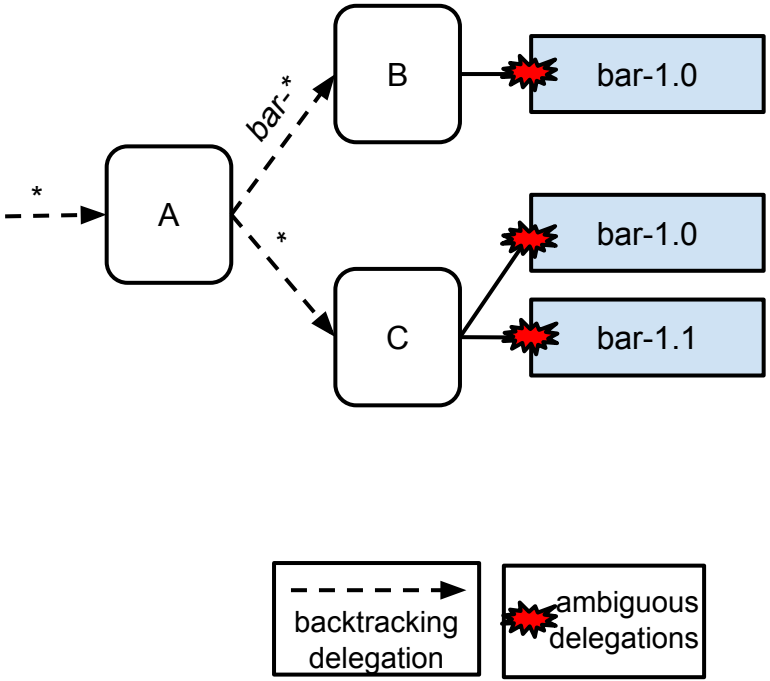
# Ambiguous delegations: failover problem

- What if B does **not** sign the bar-1.1 package, but C **does**?



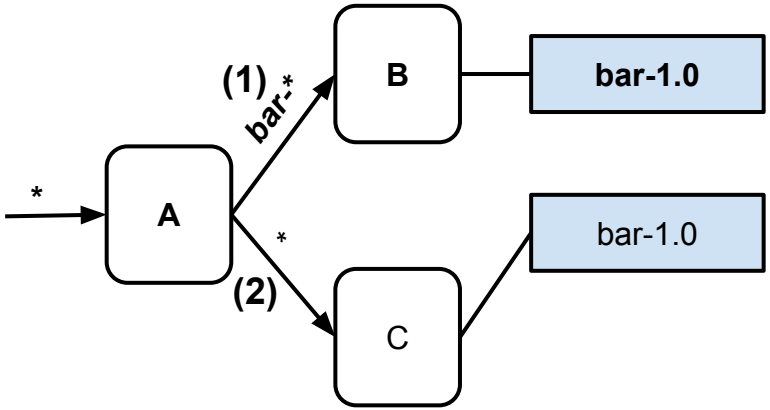
# Ambiguous delegations

- No clear answer.
- How does A say what it really means?
- "**Only** trust **B** for **bar**, and **C** for **everything else**."



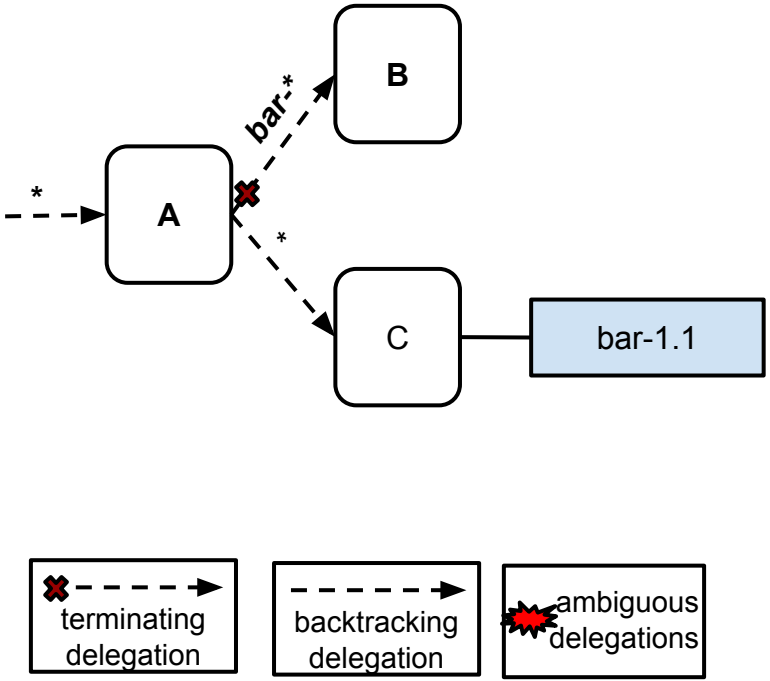
# Prioritized delegations: ordering problem

- A **prioritizes** delegation to B **before** C.
- Package manager will check B **before** C.



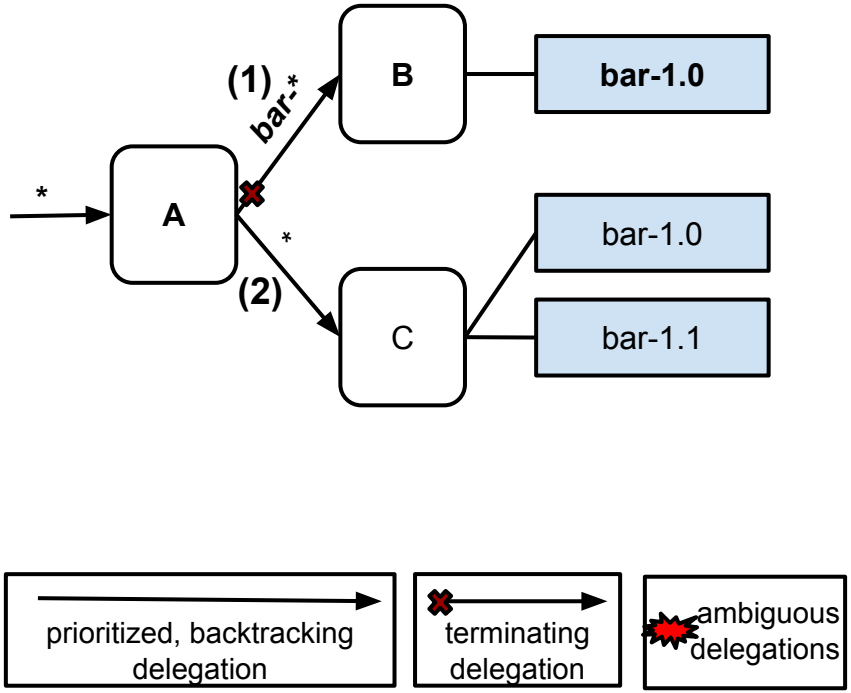

# Terminating delegations: failover problem

- A **terminates** the bar project at B.
- Package manager will search for bar **only** in B.



# Prioritized & terminating delegations

- Conflict resolution with preorder DFS.
- If delegator signed for package, return that.
- Otherwise, visit delegatees in order of priority.
- If delegation is terminating, return after delegatee visit.

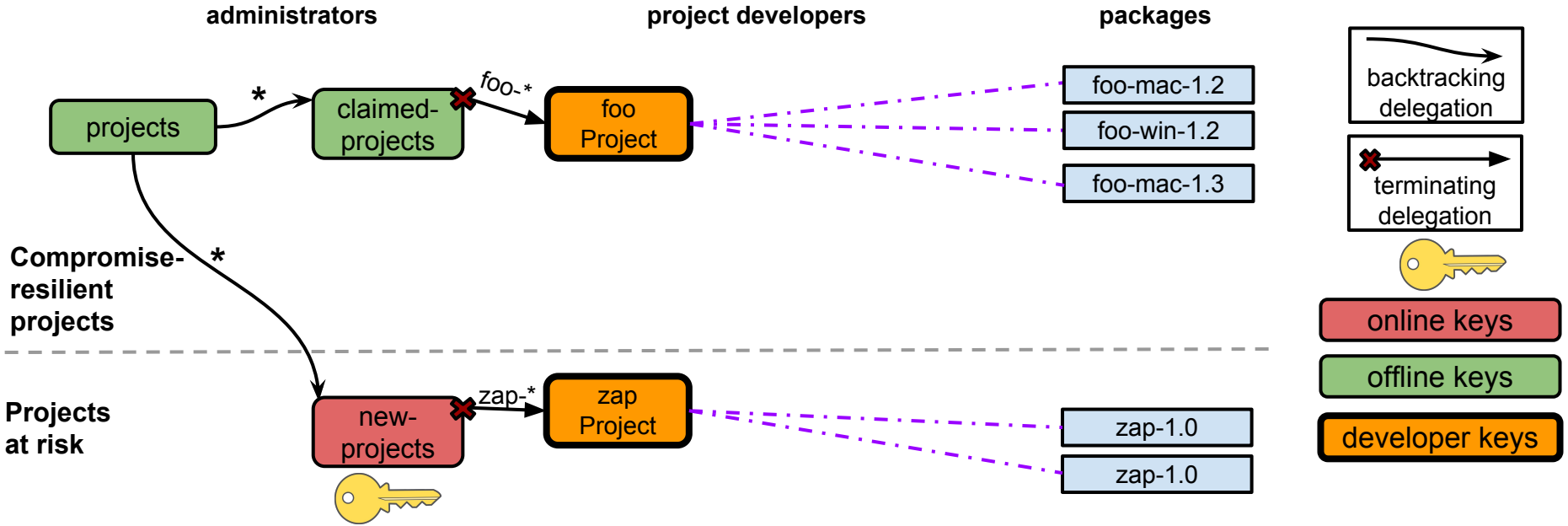


# Building usable security models

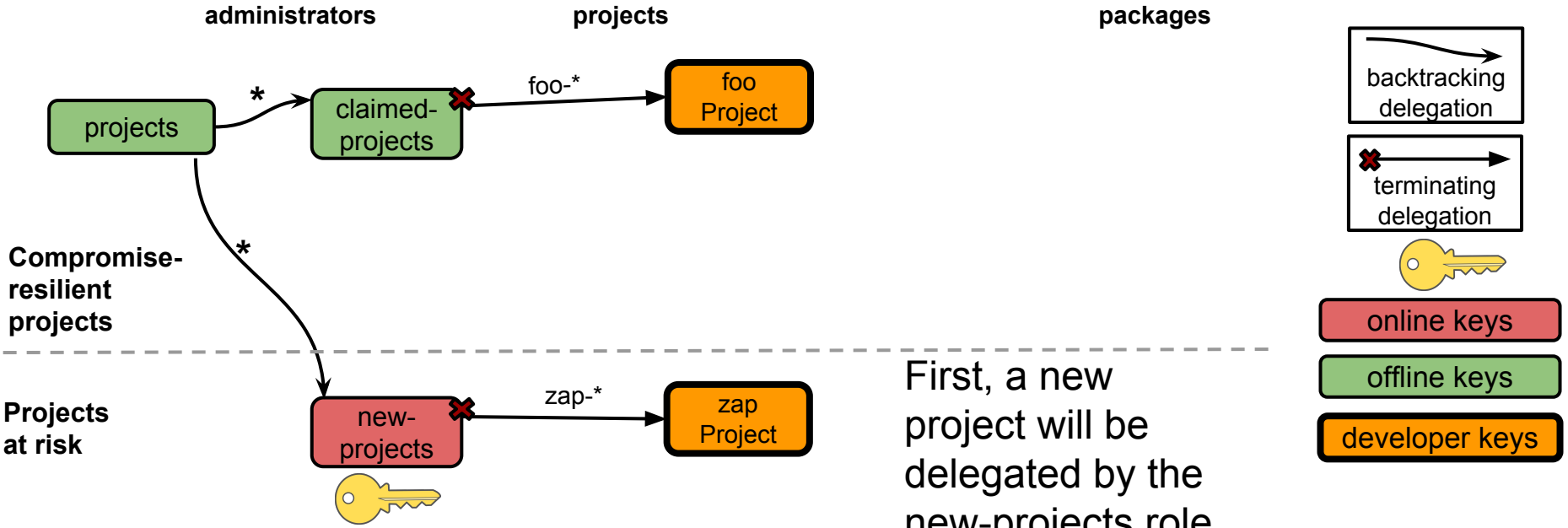
# Usable security models

- Developed from collaboration with real-world community repositories.
- **Legacy** model (**PEP 458**).
- **Maximum** model (**PEP 480**).

# Legacy/maximum security model

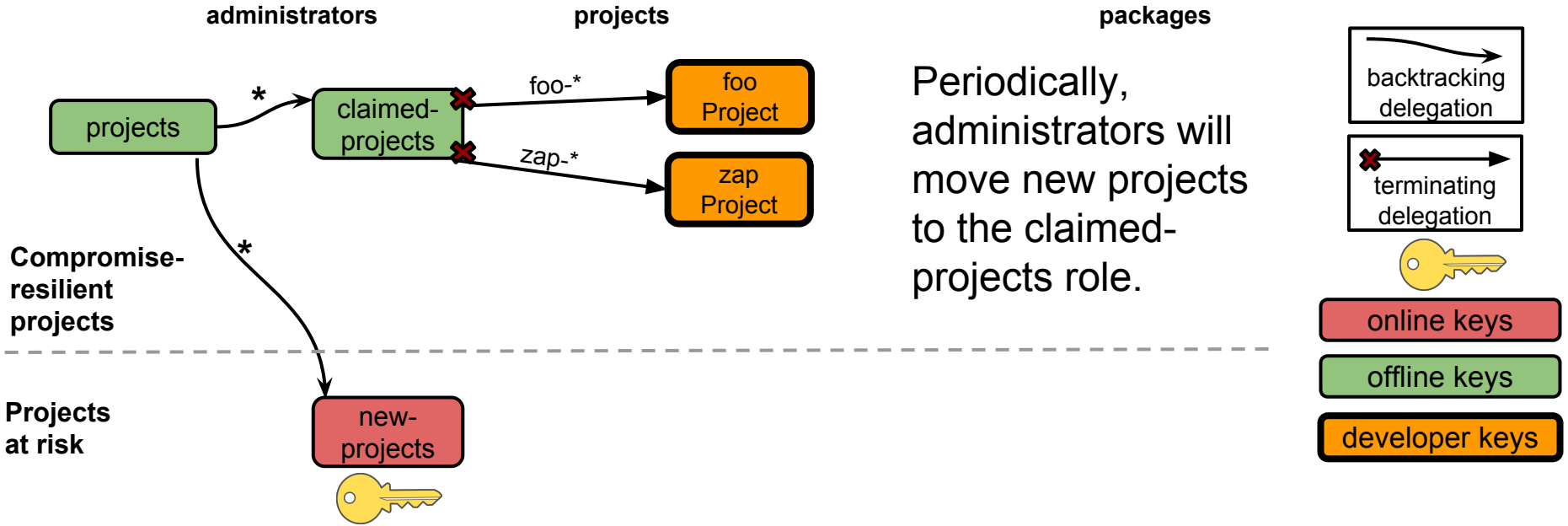


# Periodic task: claiming new projects




First, a new project will be delegated by the new-projects role.

# Periodic task: claiming new projects

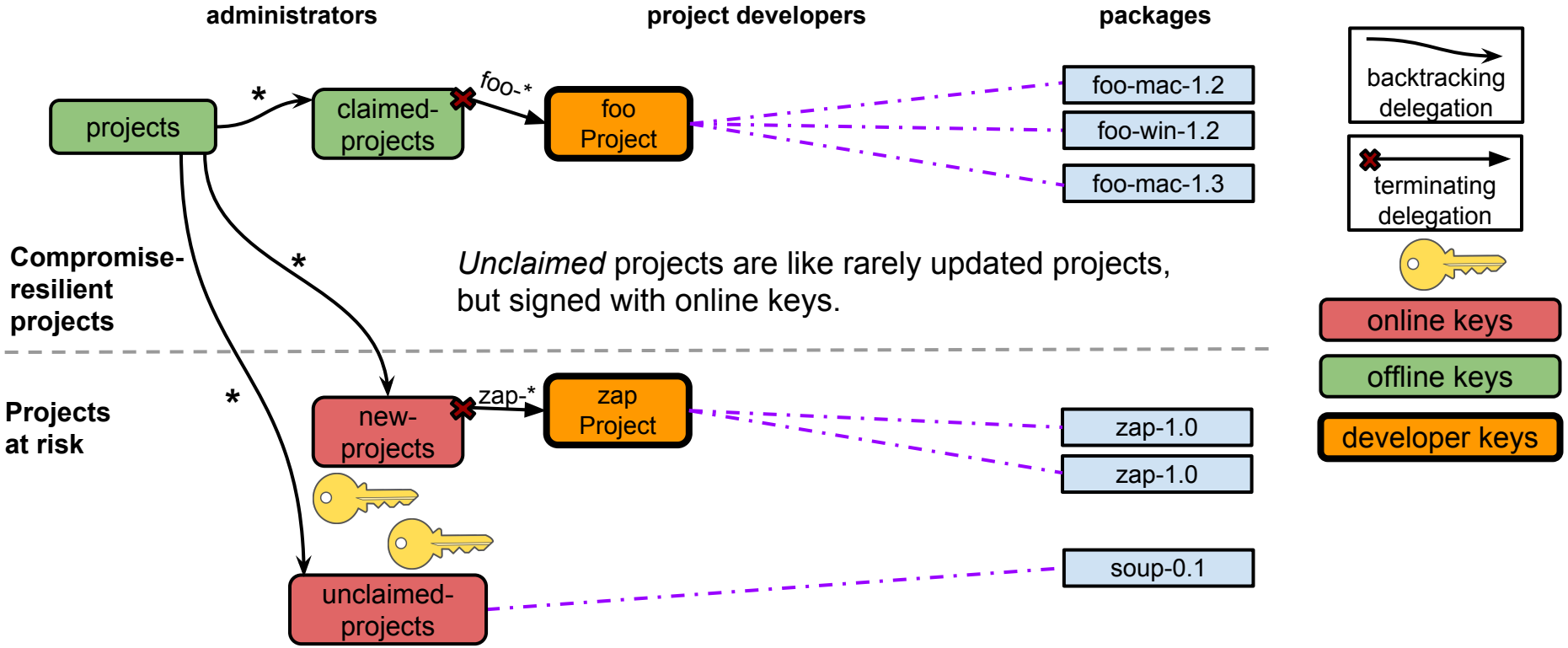



Periodically, administrators will move new projects to the claimed-projects role.

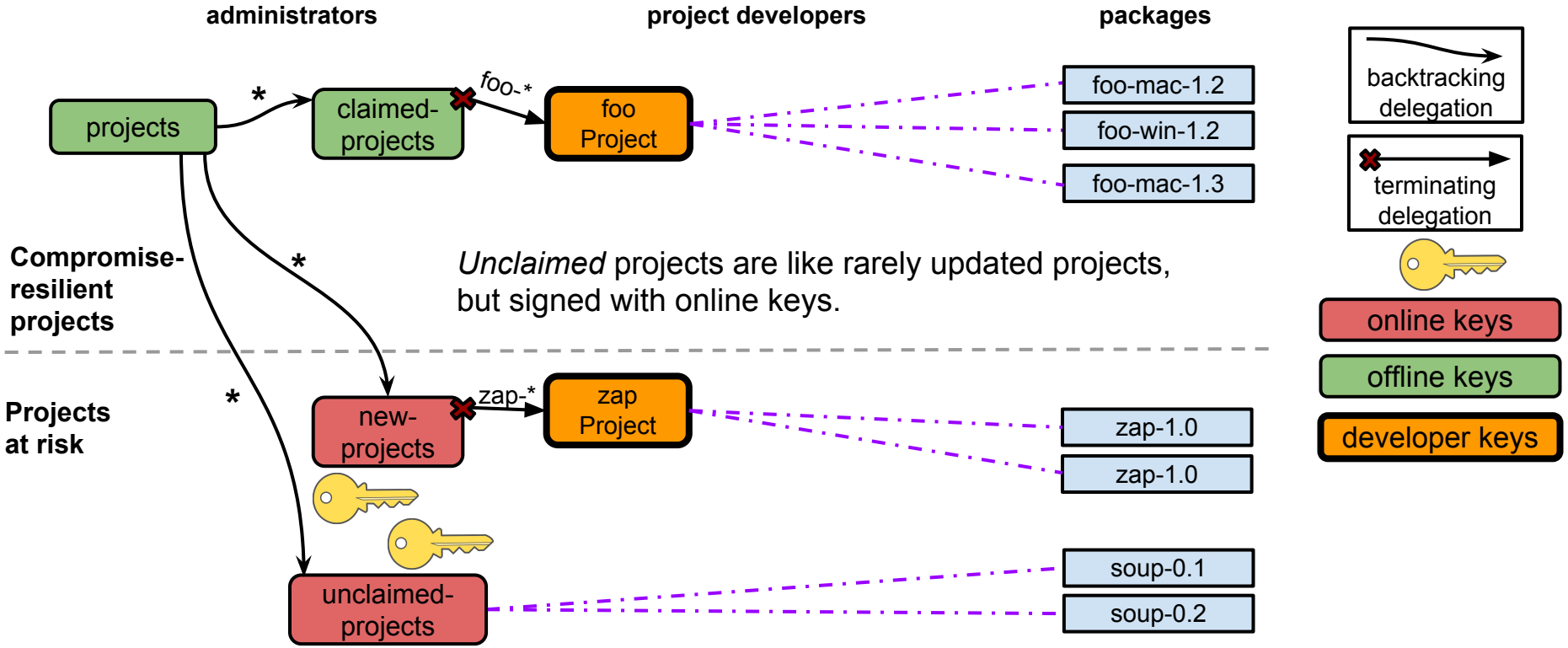
# Projects unsigned by developers

- **Developers** may **not sign** projects for various reasons
  - e.g., project no longer actively maintained
- **Idea**: why not let **administrators sign** on behalf of **developers**?

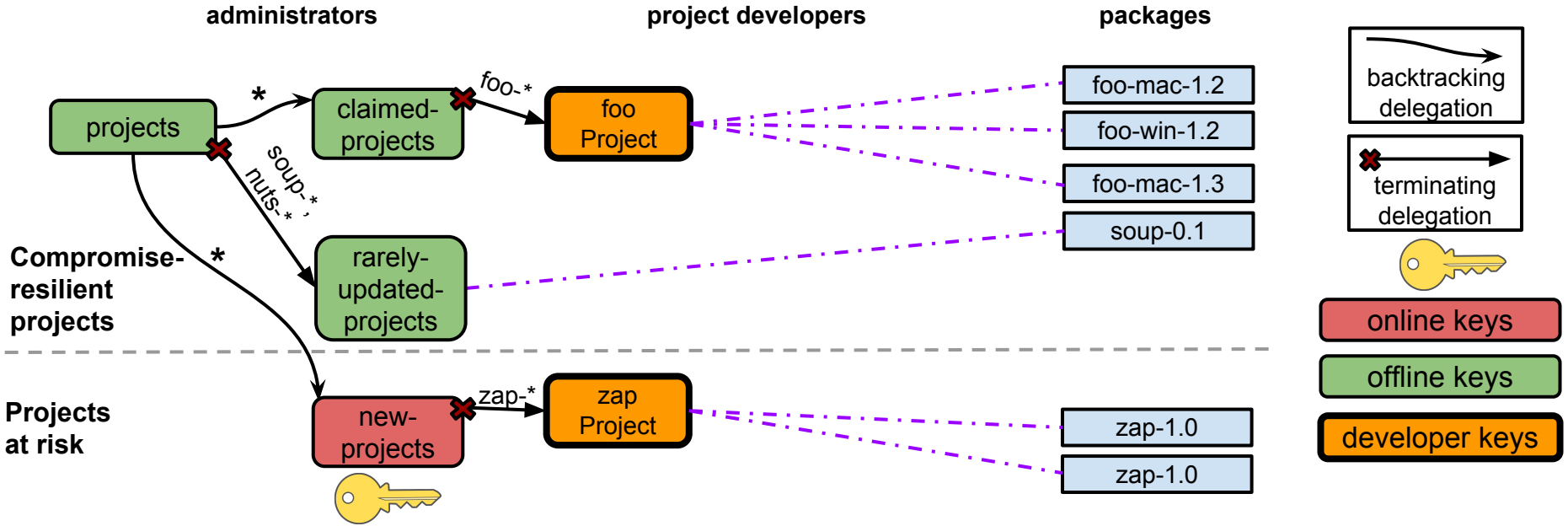
# Legacy security model



# Legacy security model

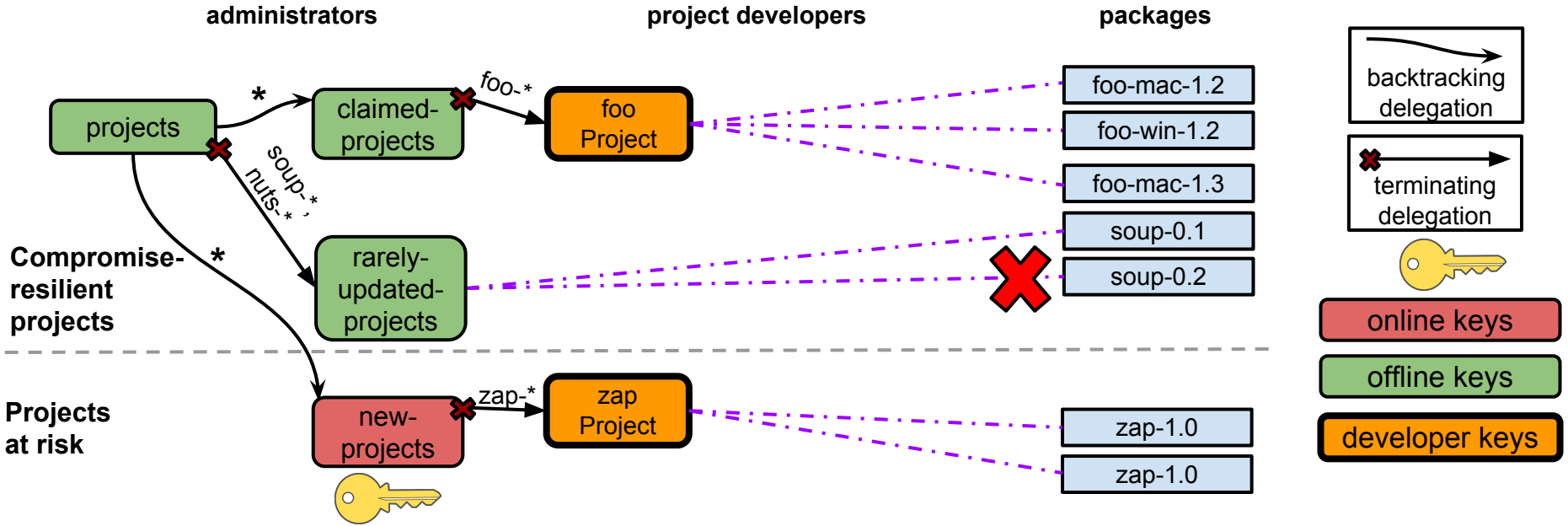


# Maximum security model




*Rarely updated* projects are not actively maintained by developers, and signed by administrators instead.

# Maximum security model




*Rarely updated* projects are not actively maintained by developers, and signed by administrators instead.

# Legacy vs maximum

| | **Legacy** | **Maximum** |
|---|---|---|
| **Claimed projects** | Compromise-resilient | Compromise-resilient |
| **New projects** | **Not** compromise-resilient | **Not** compromise-resilient |

online keys

offline keys

# Legacy vs maximum

| | Legacy | Maximum |
|---|---|---|
| **Claimed projects** | Compromise-resilient | Compromise-resilient |
| **New projects** | **Not** compromise-resilient | **Not** compromise-resilient |
| **Projects signed by administrators on behalf of developers** | **Not** compromise-resilient | |

online keys

offline keys

# Legacy vs maximum

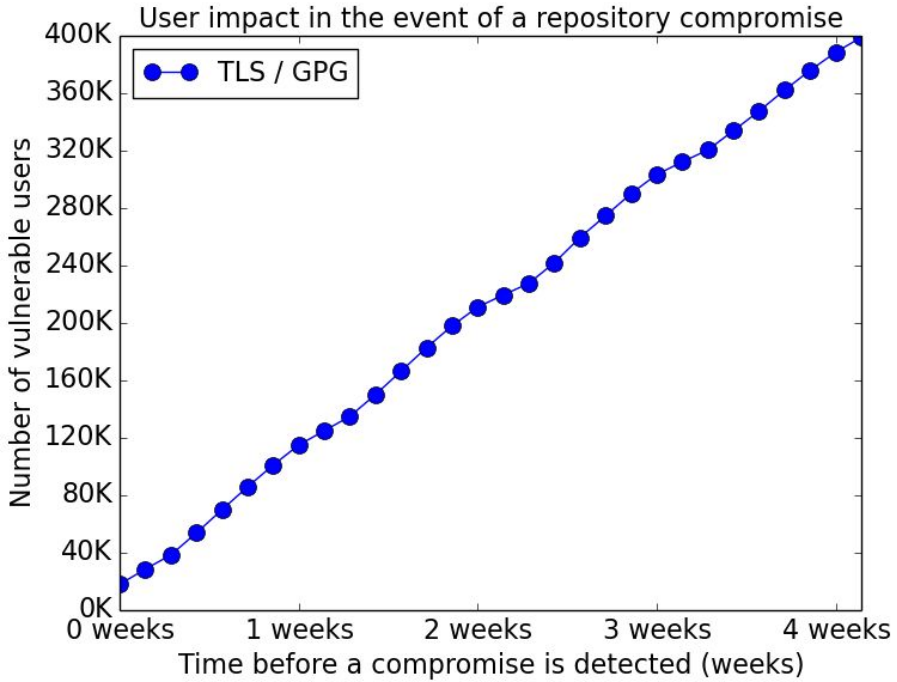
| | Legacy | Maximum |
|---|---|---|
| **Claimed projects** | Compromise-resilient | Compromise-resilient |
| **New projects** | **Not** compromise-resilient | **Not** compromise-resilient |
| **Projects signed by administrators on behalf of developers** | **Not** compromise-resilient | Compromise-resilient |

# Usability

- UX for users, developers & administrators.
- Revoking/replacing project/developer keys.
- Smooth transition from legacy to maximum.
- Securely recovering from a repository compromise.
- Please see paper for details!

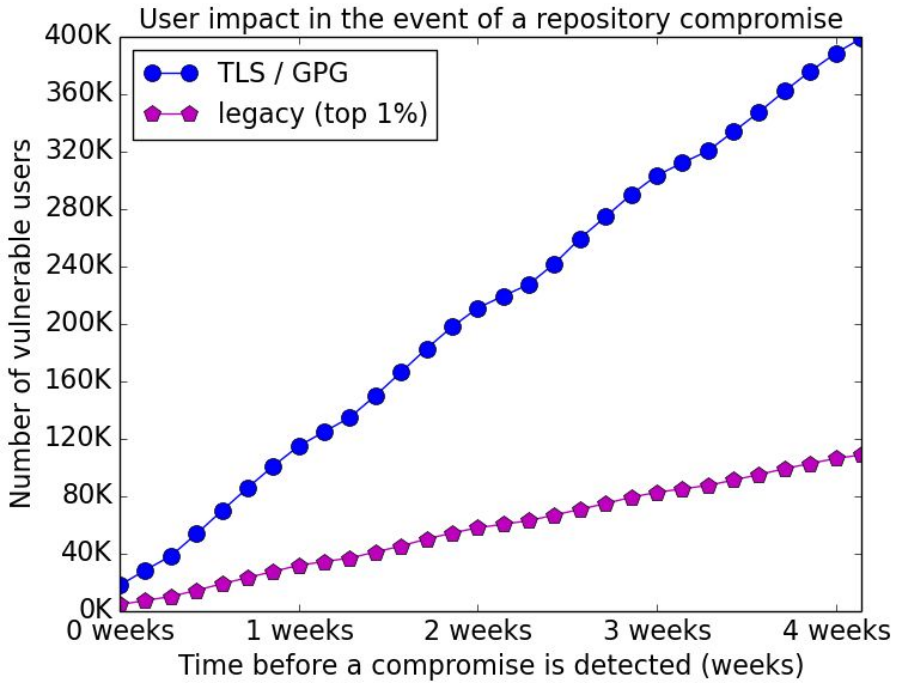
# Evaluation on PyPI: TLS/GPG

1. What if PyPI was compromised undetected for a month?
2. Sanitized download log from >1m to 400K users.
   a. See paper for details.
3. What if PyPI had used only TLS/GPG (i.e., no compromise-resilience)?



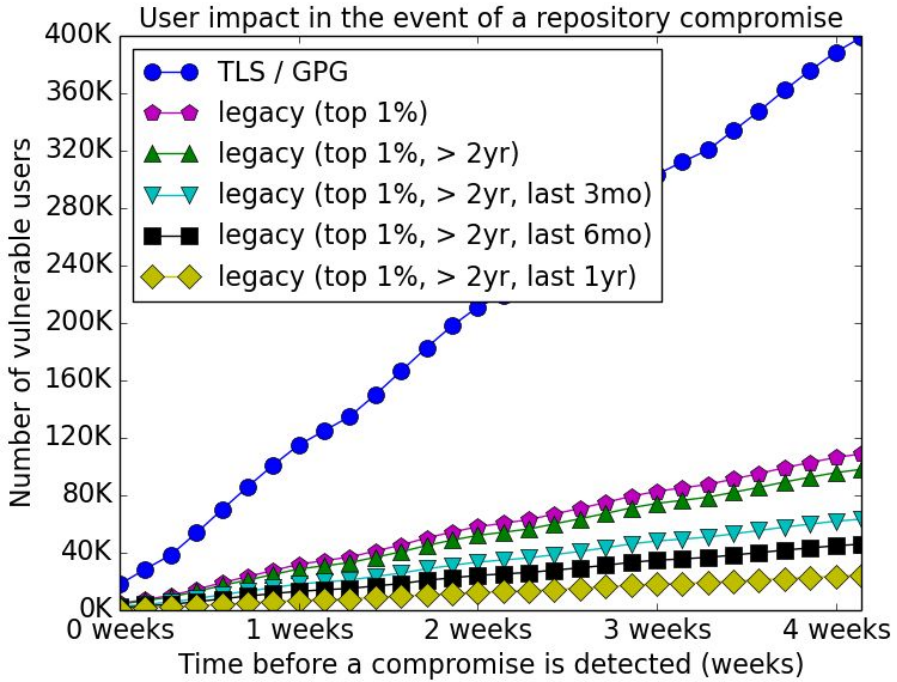
# Evaluation on PyPI: legacy (popular)

1. Claim top 1% popular projects: protect 73% users.



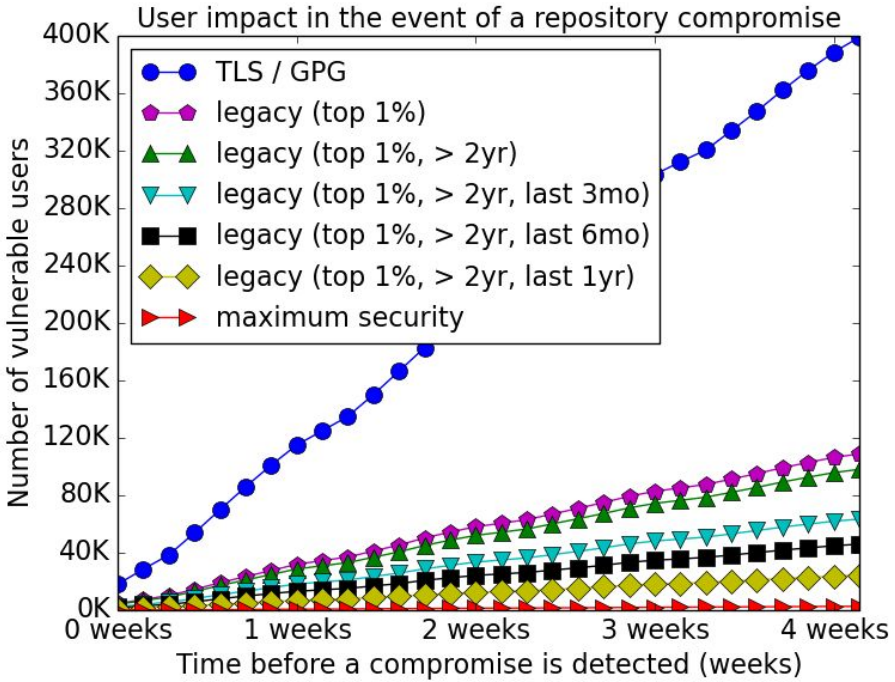
# Evaluation on PyPI: legacy (hybrid)

1. Claim top 1% popular projects: protect 73% users.
2. Claim rarely updated projects: protect 75% users.
3. Claim projects on update: protect 94% users.



# Evaluation on PyPI: maximum

Protect >99% users.

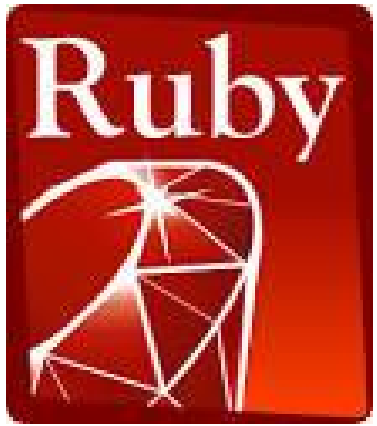


# Conclusion

# Deployments & Integrations





# Q & A

Thanks!

Questions?

**https://theupdateframework.com**

**trishank@nyu.edu**