

Deep Dive: TUF

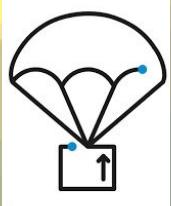
Trishank Karthik Kuppusamy, Datadog



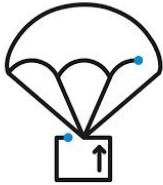
Justin Cappos, NYU



Kubecon North America 2018



Repository compromise



Software updates

- Experts agree: software updates the most security practice (USENIX SOUPS 2015)
- Updates fix security vulns
- However, important problem is often neglected...



“...no one can hack my mind”: Comparing Expert and Non-Expert Security Practices

Julia Ion
Google
juliaion@google.com

Rob Reeder
Google
reeder@google.com

Sunny Consolvo
Google
sconsolvo@google.com

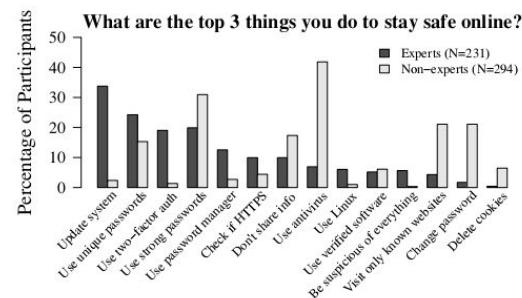


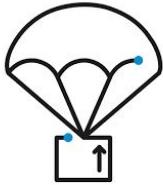
Figure 1: Security measures mentioned by at least 5% of each group. While most experts said they keep their system updated and use two-factor authentication to stay safe online, non-experts emphasized using antivirus software and using strong passwords.



Repository compromise

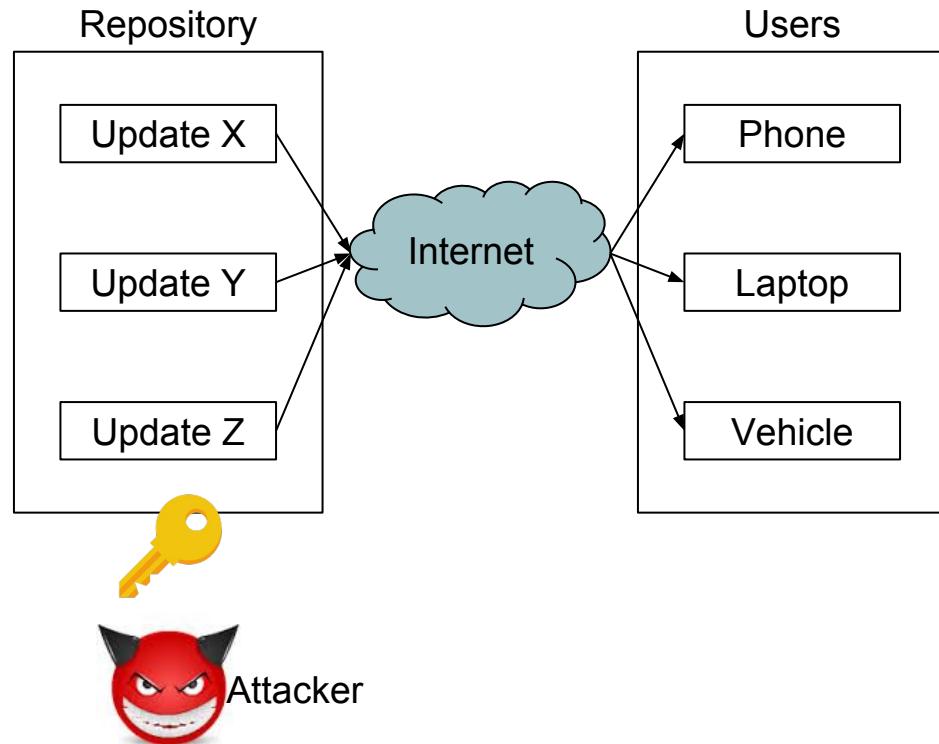
- Examples:
 - Microsoft Windows Update (2012): **Flame** malware targeted Iran nuclear efforts
 - **South Korea** cyberattack (2013): >\$750M USD in economic damage
 - **NotPetya** (2017): infected multinational corporations
- Compromise **millions** of devices

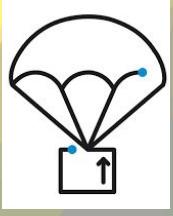




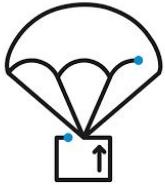
Goal: compromise-resilience

- Only question of when, not if
- Cannot prevent compromise
- But must severely limit impact



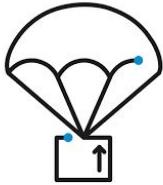


The Update Framework (TUF)



What is on a repository?

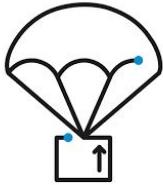
- Repository contains packages + metadata



What is on a repository?

- Repository contains packages + metadata
- *Package*
 - Smallest unit of update
 - Software application or library

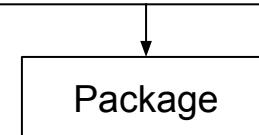
Package

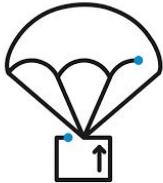


What is on a repository?

- Repository contains packages + metadata
- *Package*
 - Smallest unit of update
 - Software application or library
- *Metadata*
 - Cryptographic hashes, file sizes, version numbers, etc.
 - About packages, or other metadata files

```
{  
  "signatures": [  
    {  
      "keyid": "ce3e02e72980b09ca6f5efa68197130b381921e5d0675e2e0c8f3c47e0626bba",  
      "method": "ed25519",  
      "sig": "9095bf34b0cbf9790465c0956810cb3729bc96beed8ee7e42d98997b1e8ec0a6780e575565"  
    }  
  ],  
  "signed": {  
    "_type": "Targets",  
    "expires": "2030-01-01T00:00:00Z",  
    "targets": {  
      "/project/file3.txt": {  
        "hashes": {  
          "sha256": "141f740f53781d1ca54b8a50af22cbf74e44c21a998fa2a8a05aaac2c002886b"  
        },  
        "length": 28  
      }  
    },  
    "version": 1  
  }  
}
```





The Update Framework (TUF): secure software updates

- **Authenticity and integrity**—even if repository compromised
- Design principles
 - Separation of duties
 - Threshold signatures
 - Explicit & implicit revocation of keys
 - Minimizing risk using offline keys
 - Selective delegation of trust
 - Diversity of hashing + signing algorithms
- (CCS 2010)
- <https://theupdateframework.com>

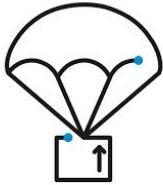
Survivable Key Compromise in Software Update Systems

Justin Samuel^{*}
UC Berkeley
Berkeley, California, USA
jsamuel@berkeley.edu

Nick Mathewson
The Tor Project
nickm@alum.mit.edu

Justin Cappos
University of Washington
Seattle, Washington, USA
justinc@cs.washington.edu

Roger Dingledine
The Tor Project
arma@mit.edu

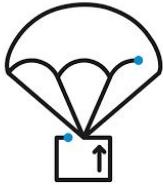


Separation of duties

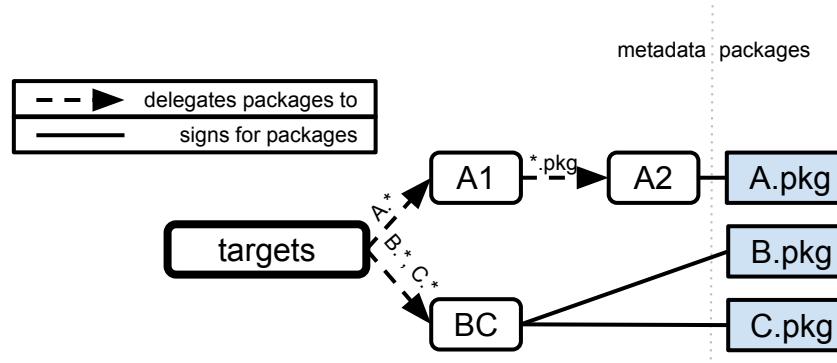
Design principles:

1. **Separation of duties**

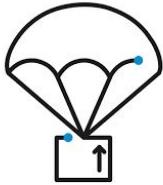
(i.e., don't put all your eggs in one basket).



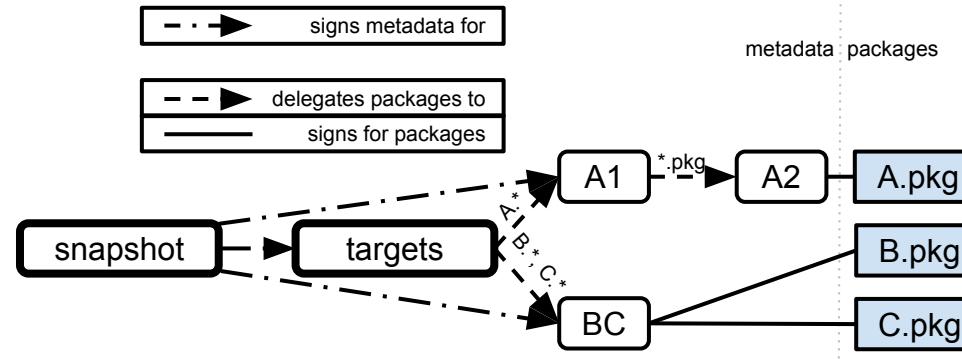
The targets role



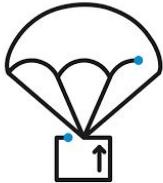
Role	Purpose
targets	Indicates metadata such as the cryptographic hashes and file sizes of packages. May delegate this responsibility to other, custom-made roles.



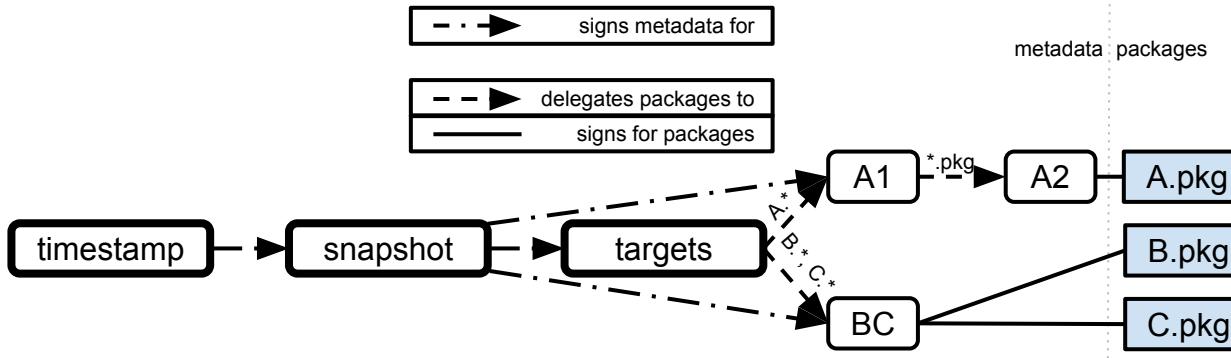
The snapshot role



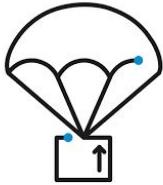
Role	Purpose
targets	Indicates metadata such as the cryptographic hashes and file sizes of packages. May delegate this responsibility to other, custom-made roles.
snapshot	Indicates which packages have been released at the same time by the repository.



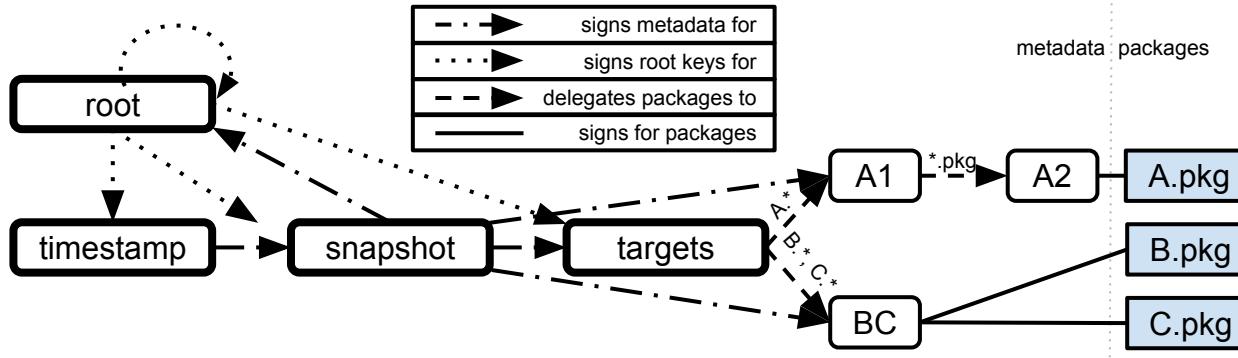
The timestamp role



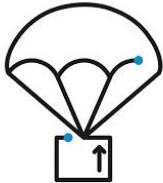
Role	Purpose
targets	Indicates metadata such as the cryptographic hashes and file sizes of packages. May delegate this responsibility to other, custom-made roles.
snapshot	Indicates which packages have been released at the same time by the repository.
timestamp	Indicates whether there is any new metadata or package on the repository.



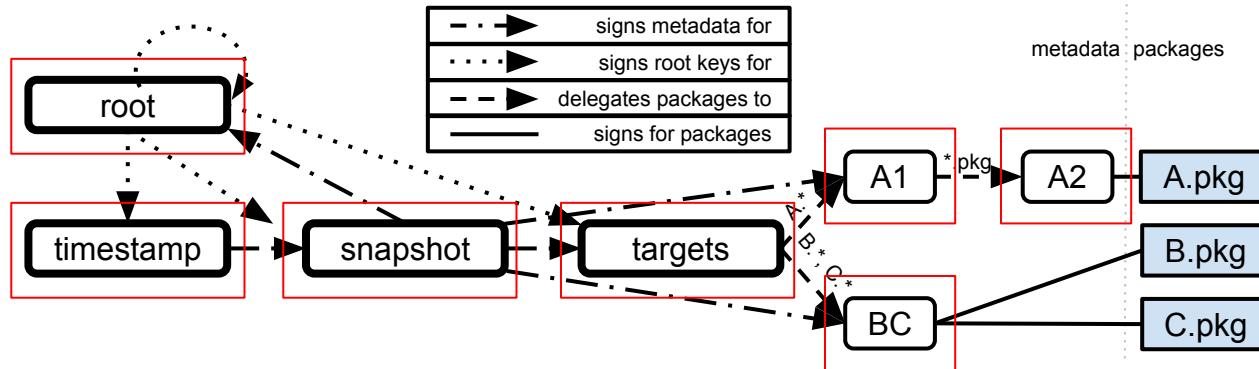
The root role



Role	Purpose
targets	Indicates metadata such as the cryptographic hashes and file sizes of packages. May delegate this responsibility to other, custom-made roles.
snapshot	Indicates which packages have been released at the same time by the repository.
timestamp	Indicates whether there is any new metadata or package on the repository.
root	Serves as the certificate authority for the repository. Distributes and revokes the public keys used to verify the root, timestamp, snapshot, and targets role metadata.



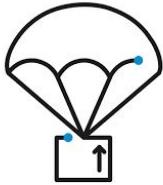
Separation of duties



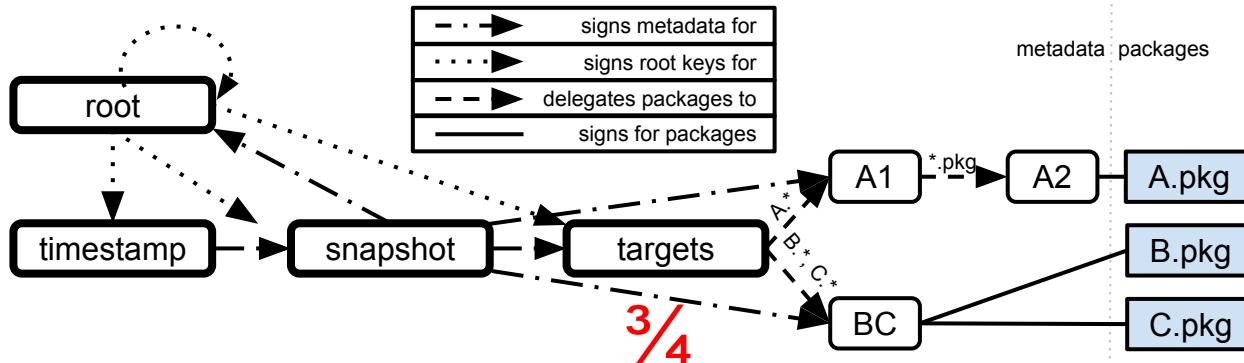
Design principles:

1. Separation of duties

(i.e., don't put all your eggs in one basket).

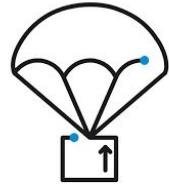


Threshold signatures

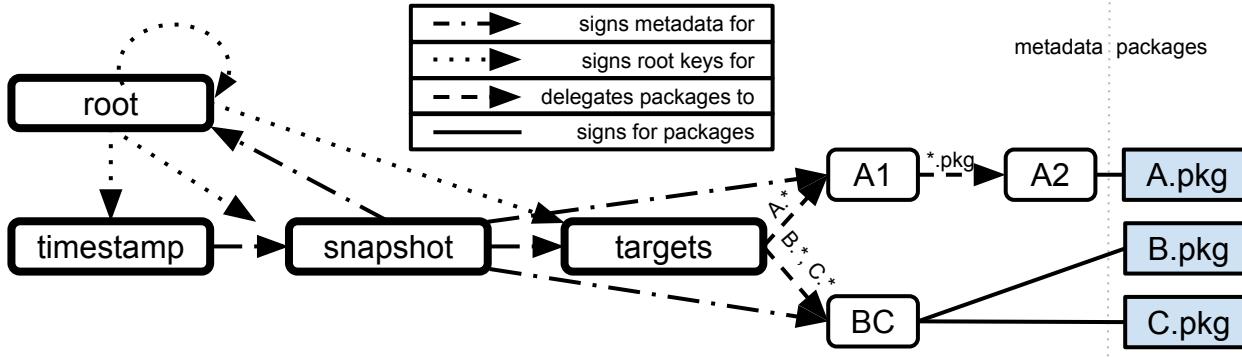


Design principles:

1. Separation of duties.
2. **Threshold signatures**
(i.e., like the two-man rule to launch nuclear missiles).

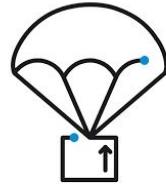


Explicit & implicit revocation of keys

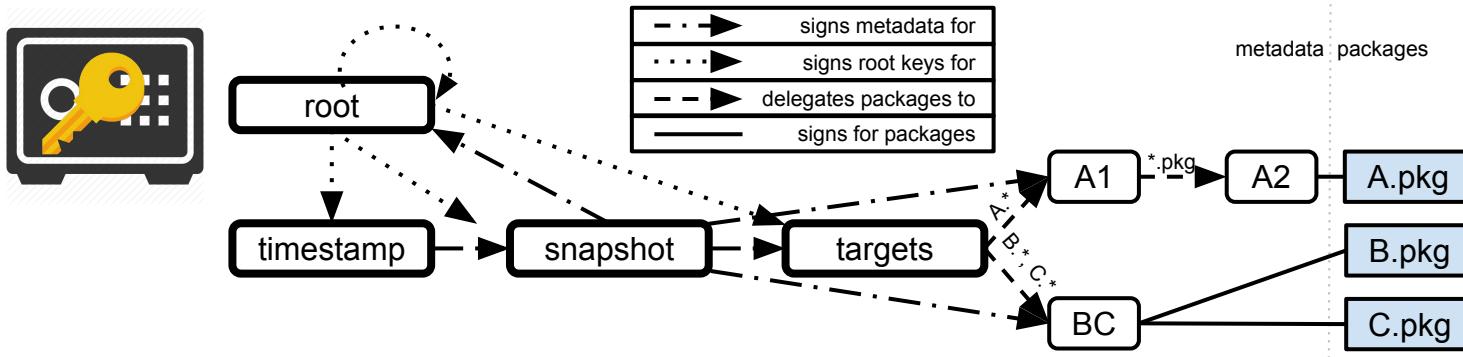


Design principles:

1. Separation of duties.
2. Threshold signatures.
3. **Explicit and implicit revocation of keys.**

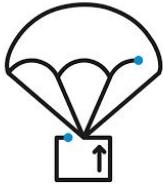


Minimizing risk with offline keys



Design principles:

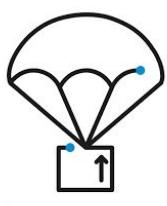
1. Separation of duties.
2. Threshold signatures.
3. Explicit and implicit revocation of keys.
4. **Minimized risk through use of offline keys**
(i.e., don't put keys to the kingdom under the carpet).



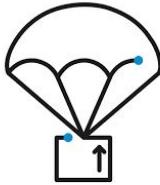
Diversity of cryptographic algorithms

- **Hedge your bets**
- Can't break TUF unless you **break them all**
- No need to depend on just SHA-2 or SHA-3, RSA or Ed25519
- Can even try **post-quantum crypto** at the same time





How TUF Has (and Does) Evolve



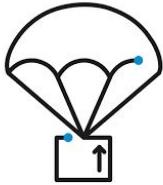
TUF Standardization Process (TAPs)

- TAP 3 -- multi-role signatures over unequal quorums
- TAP 4 -- multi-repository consensus
- TAP 5 -- split repository location across URLs [draft]
- TAP 6 -- version numbers in root metadata
- ~~TAP 7 -- TUF conformance testing [rejected]~~
- TAP 8 -- Key rotation / self revocation [draft]
- TAP 9 -- Mandated metadata signing scheme
- TAP 10 -- Remove native compression support

Future TAPs

- Clearer versioning support
- Wireline formats
- Partially signed threshold metadata
- Supply chain security integration

Discuss with us, then submit (TAP 1/2)



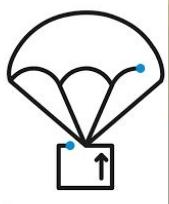
TUF Standardization Process (TAPs)

- TAP 3 -- multi-role signatures over unequal quorums
- TAP 4 -- multi-repository consensus
- TAP 5 -- split repository location across URLs [draft]
- TAP 6 -- version numbers in root metadata
- ~~TAP 7 -- TUF conformance testing [rejected]~~
- TAP 8 -- Key rotation / self revocation [draft]
- TAP 9 -- Mandated metadata signing scheme
- TAP 10 -- Remove native compression support

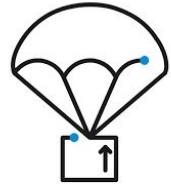
Future TAPs

- Clearer versioning support
- Wireline formats
- Partially signed threshold metadata
- Supply chain security integration

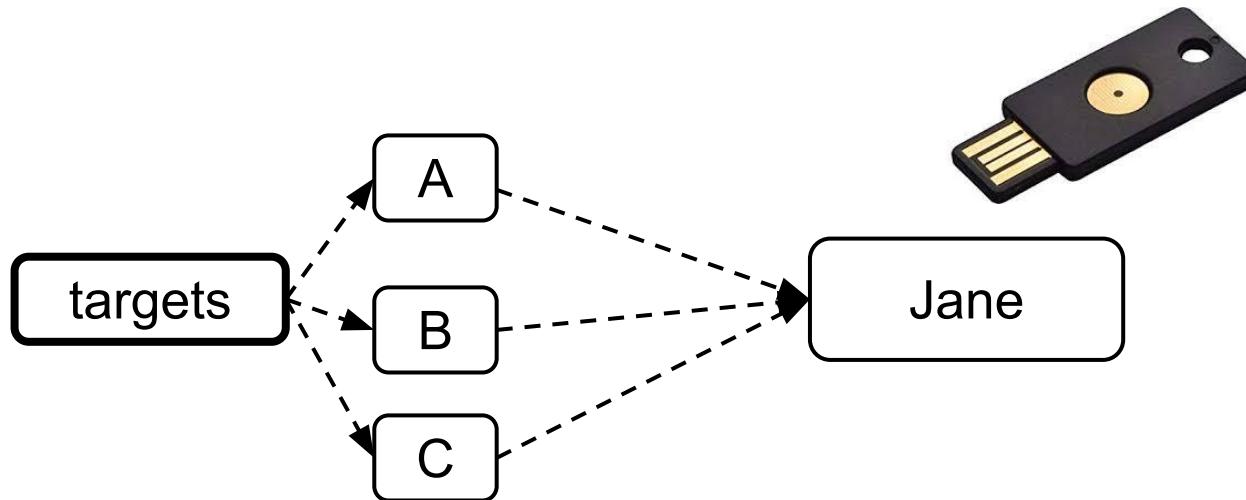
Discuss with us, then submit (TAP 1/2)



TAP 8: Key Rotation and Self Revocation

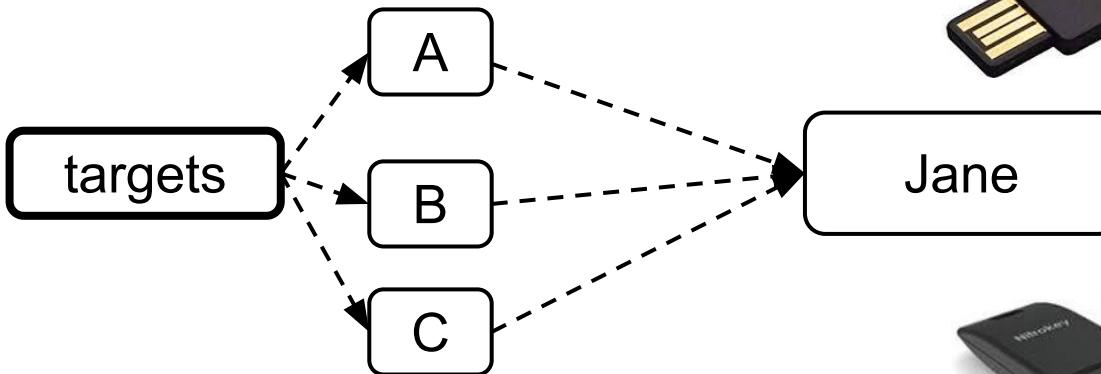


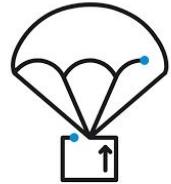
TAP 8: Key rotation / self revocation



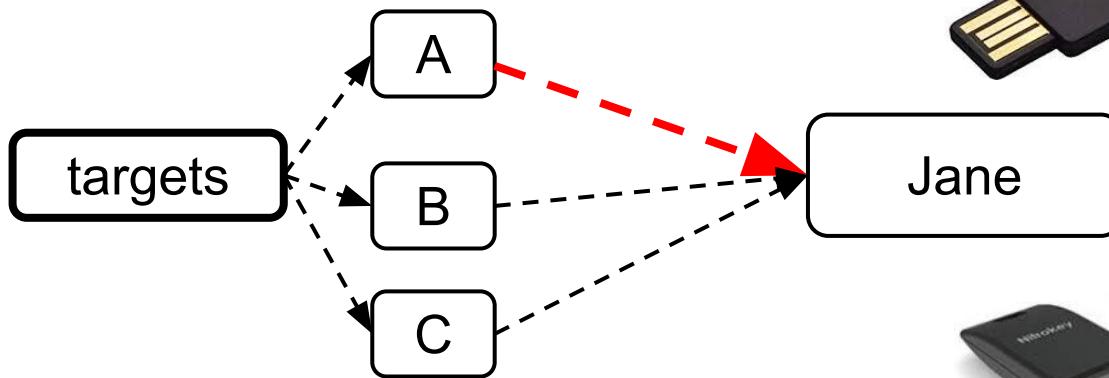


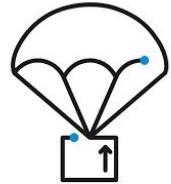
TAP 8: Key rotation / self revocation



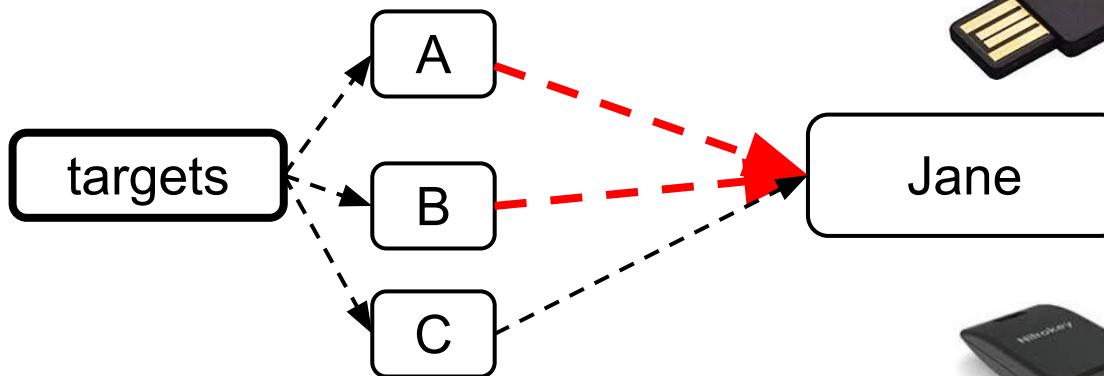


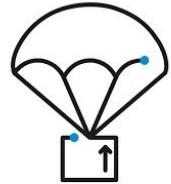
TAP 8: Key rotation / self revocation



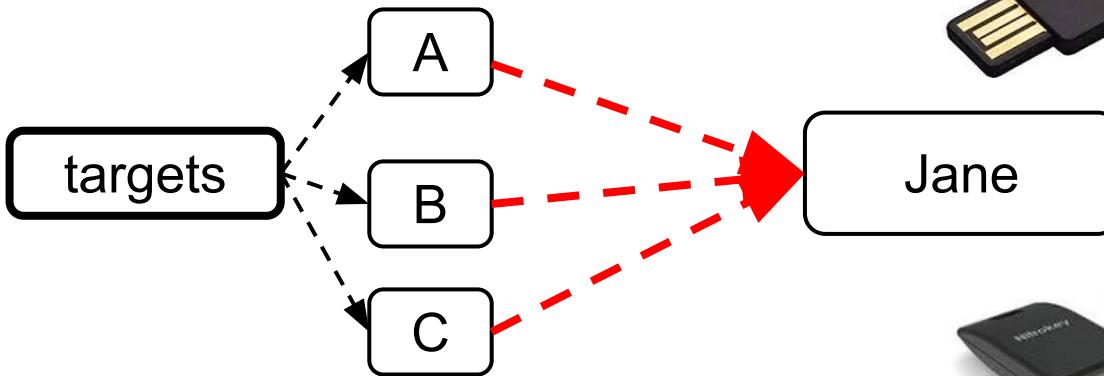


TAP 8: Key rotation / self revocation



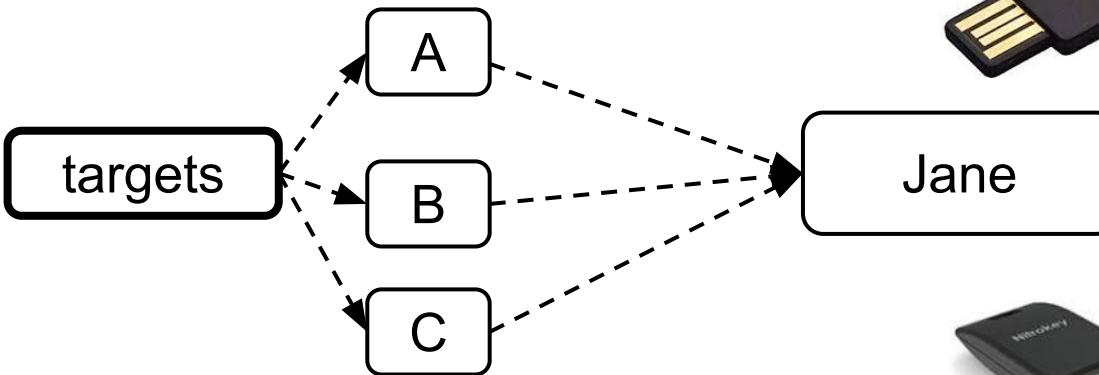


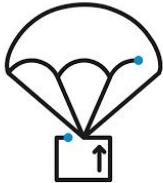
TAP 8: Key rotation / self revocation





TAP 8: Key rotation / self revocation

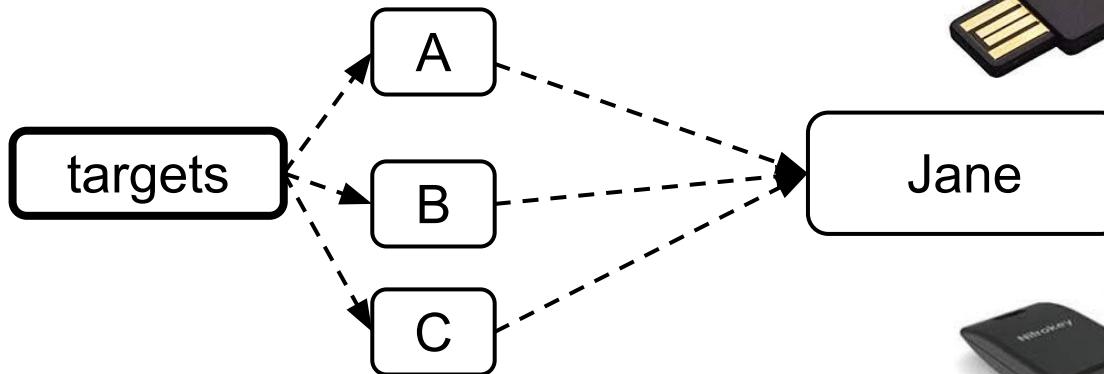


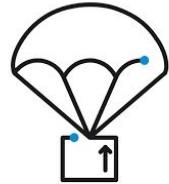


TAP 8: Key rotation / self revocation

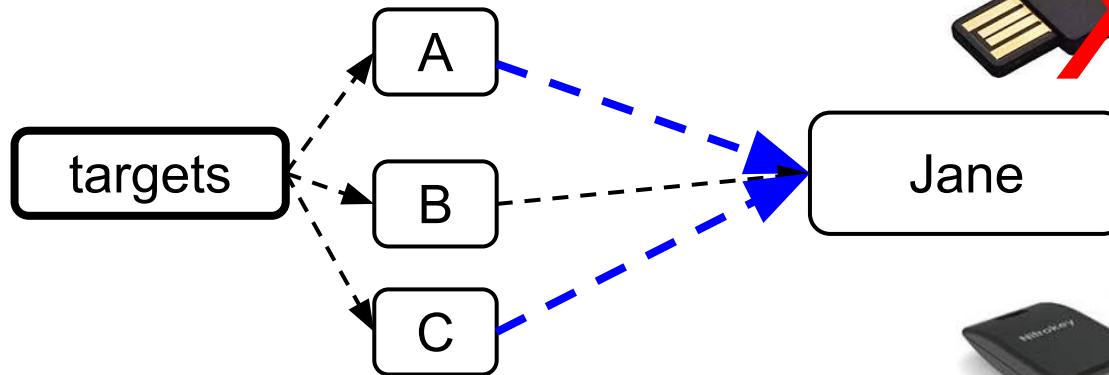


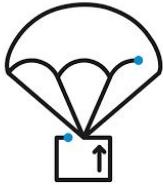
Jane



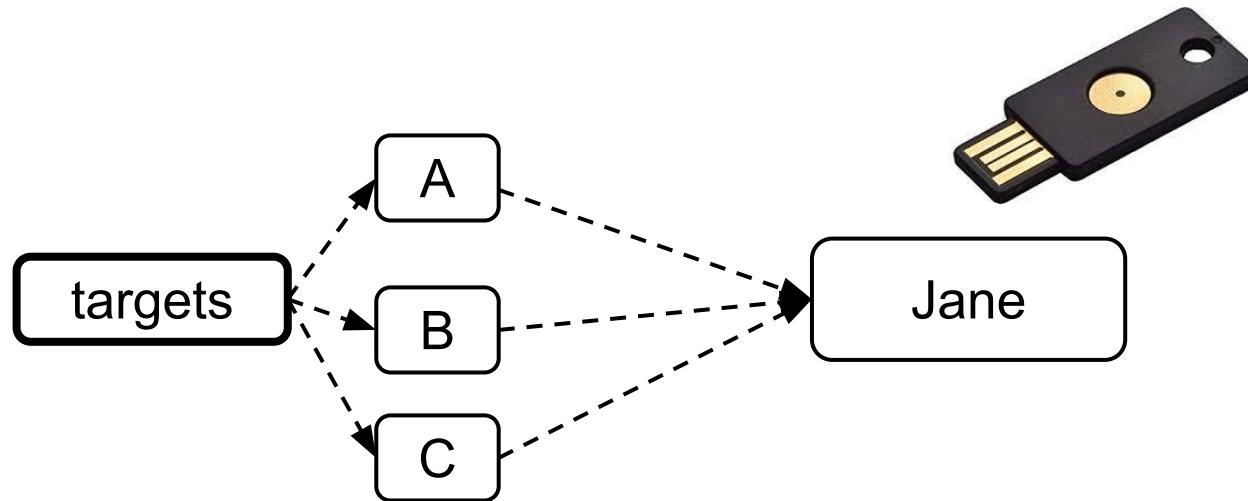


TAP 8: Key rotation / self revocation

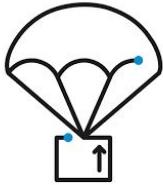




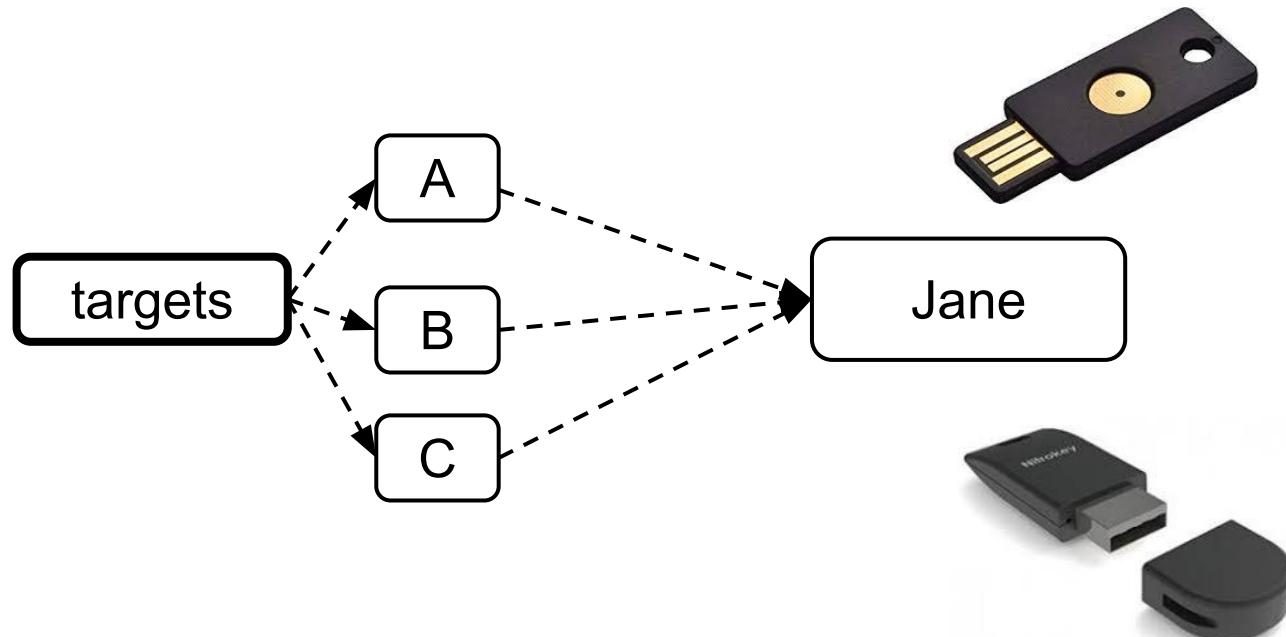
TAP 8: Key rotation / self revocation



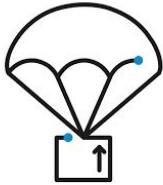
Solution: self rotation / revocation



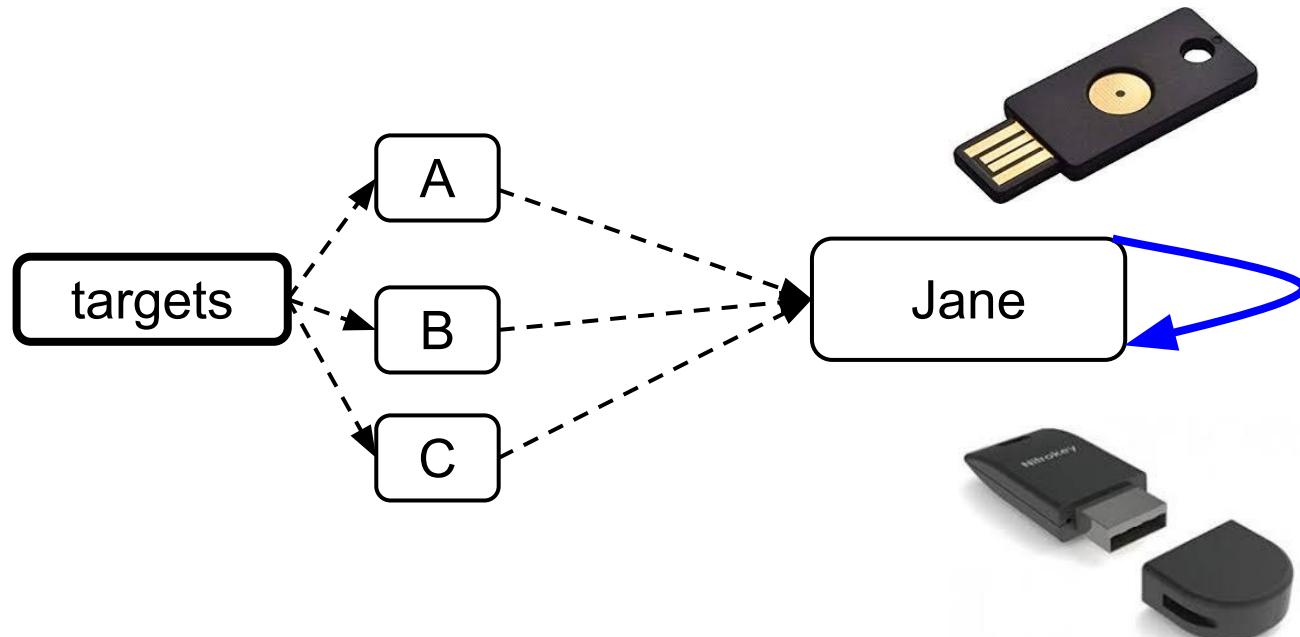
TAP 8: Key rotation / self revocation



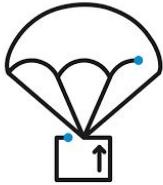
Solution: self rotation / revocation



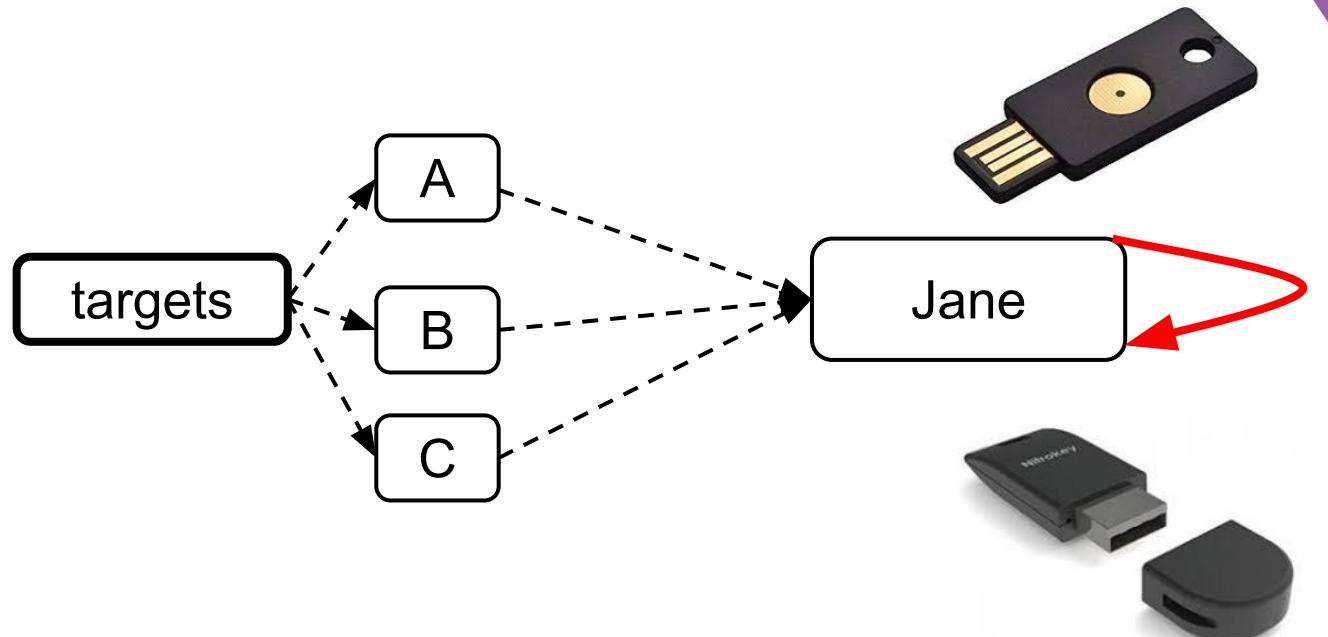
TAP 8: Key rotation / self revocation



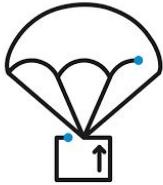
Solution: self rotation / revocation



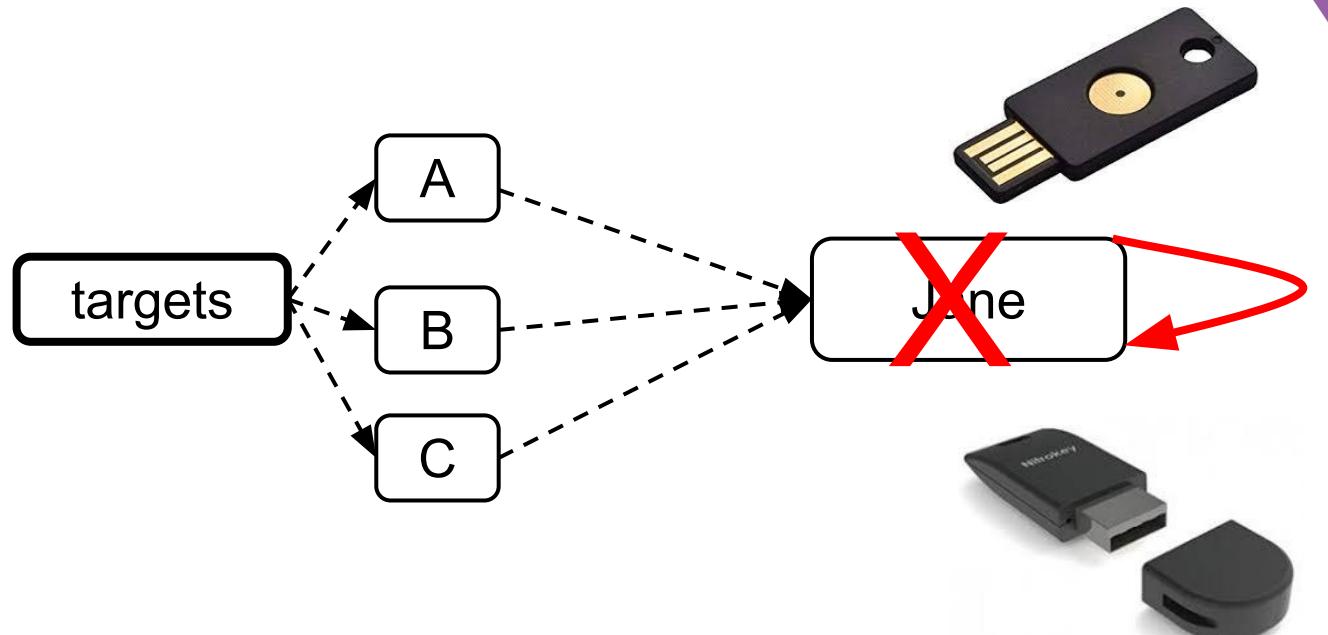
TAP 8: Key rotation / self revocation



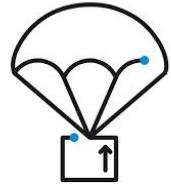
Solution: self rotation / revocation



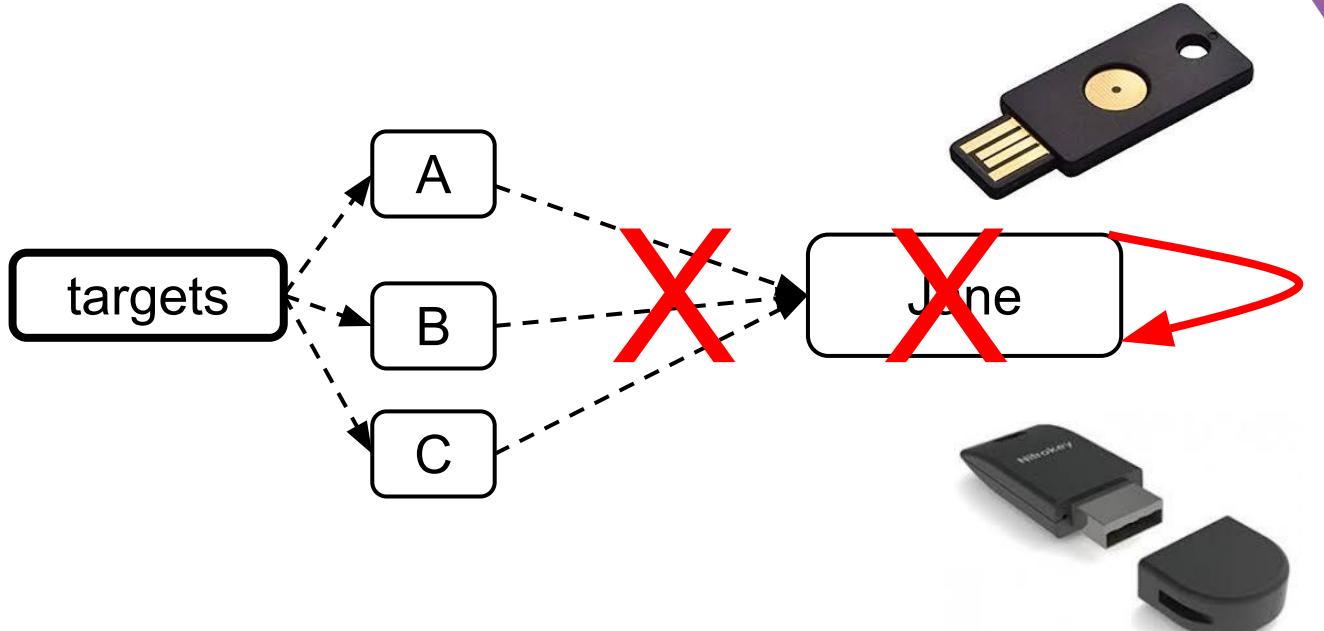
TAP 8: Key rotation / self revocation



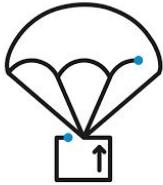
Solution: self rotation / revocation



TAP 8: Key rotation / self revocation



Solution: self rotation / revocation



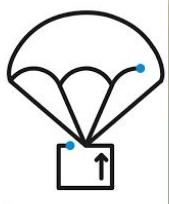
TAP 8: Key rotation / self revocation



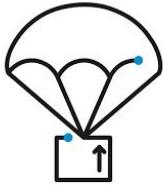
Self-managing project use case
Also very cloud-native relevant
Immediately rotate / revoke



- Hannes Mehnert, Justin Cappos, Marina Moore



TAP 5: Split repository location across URLs



TAP 5: Split repository location across URLs

- Problem: How do you partially trust a repo?
 - What if you need A, but the repo contains other packages?

A.pkg

B.pkg

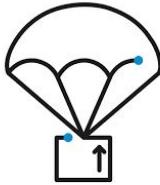
C.pkg

?

?

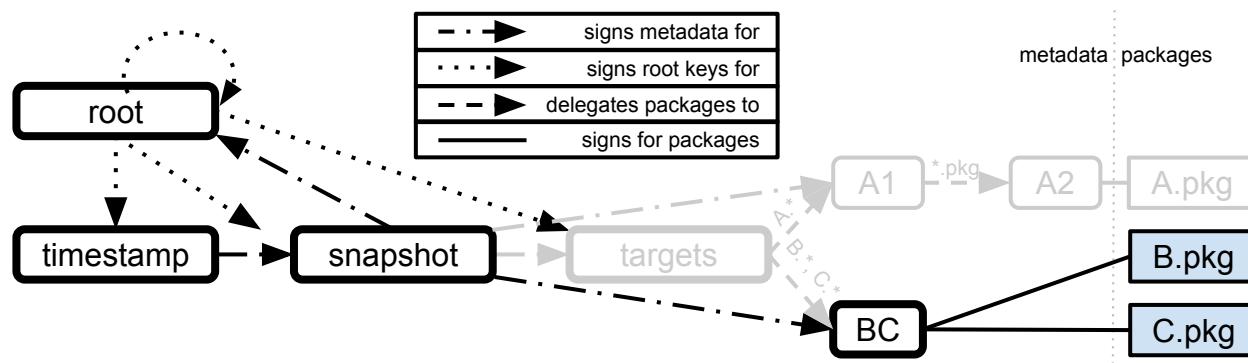
?

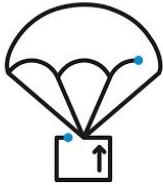
?



TAP 5: Restricting trust to a single project (example)

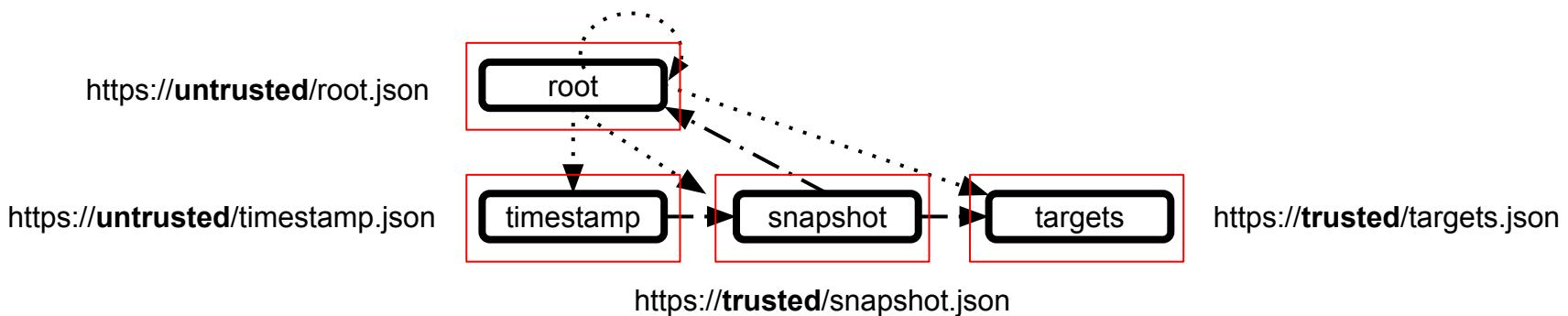
- Cloud-native use case
- Can **control what enterprise users see** on a repository
- Example: trust **only** this image on Quay

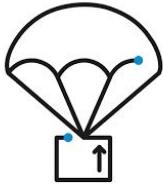




TAP 5: Trusting a mirror only for online metadata (example)

- Alternative Cloud-native use case
- Running Docker Hub in **adversarial** environments
- Potentially hostile server trusted **only** for **timeliness** and **consistency** of images

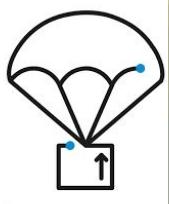




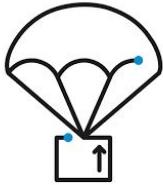
TAP 5: Split repository location across URLs

- Came out of discussions with **CoreOS**
 - Evan Cordell, Jake Moshenko





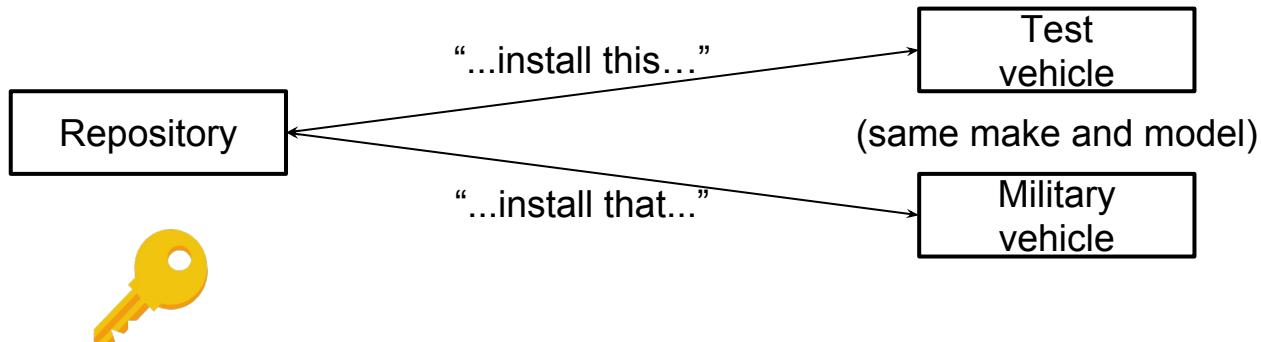
TAP 4: Multi-repository consensus

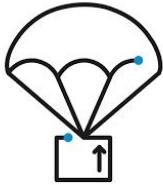


TAP 4: Multi-repository consensus

Scenario: Repository controls what updates are applied

Question: Should the repository sign this info with a key on the repo or a key kept offline?

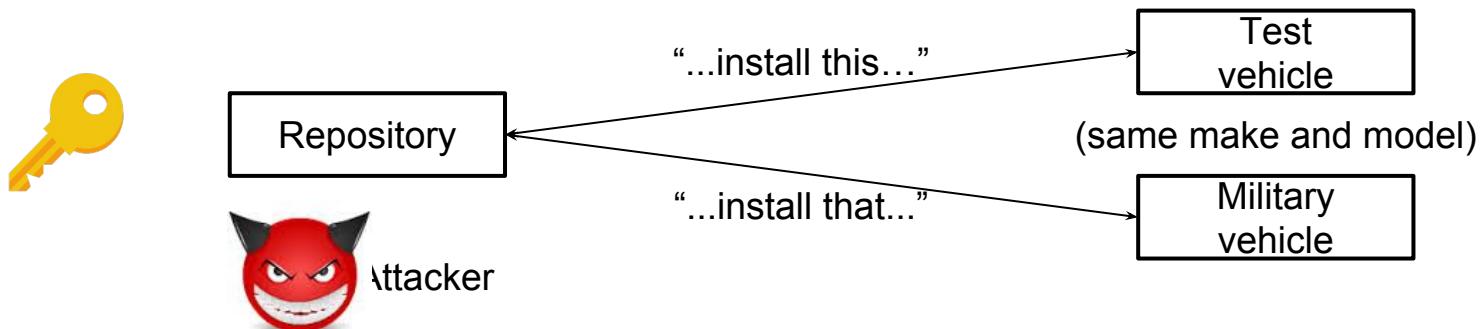


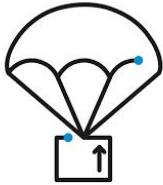


TAP 4: Multi-repository consensus

Online key: Flexible but insecure

- Use **online** keys to sign all metadata
- **Pro: on-demand customization**
 - Easy to install different updates on vehicles of same make and model
 - Can instantly blacklist only buggy updates
- **Con: no compromise-resilience**
 - Attackers cannot tamper with metadata without being detected

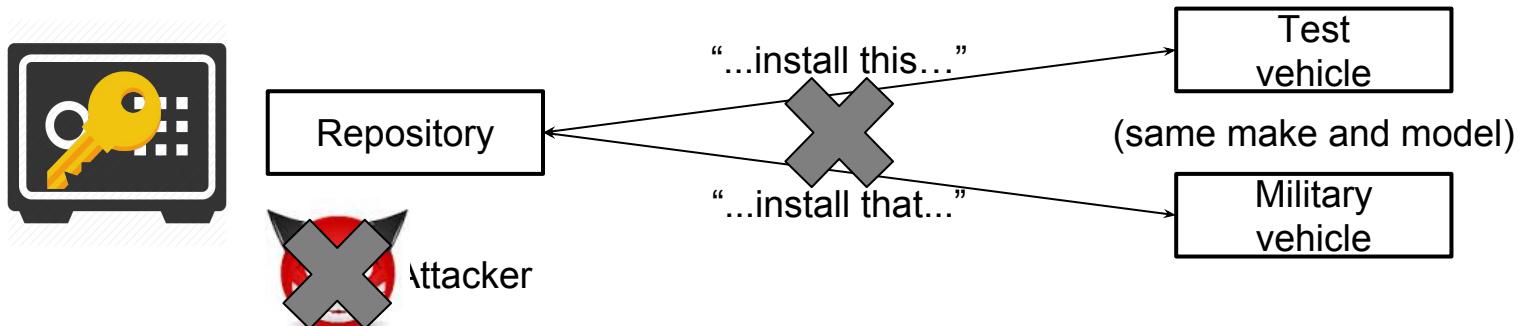


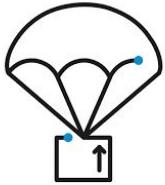


TAP 4: Multi-repository consensus

Offline key: Secure but inflexible

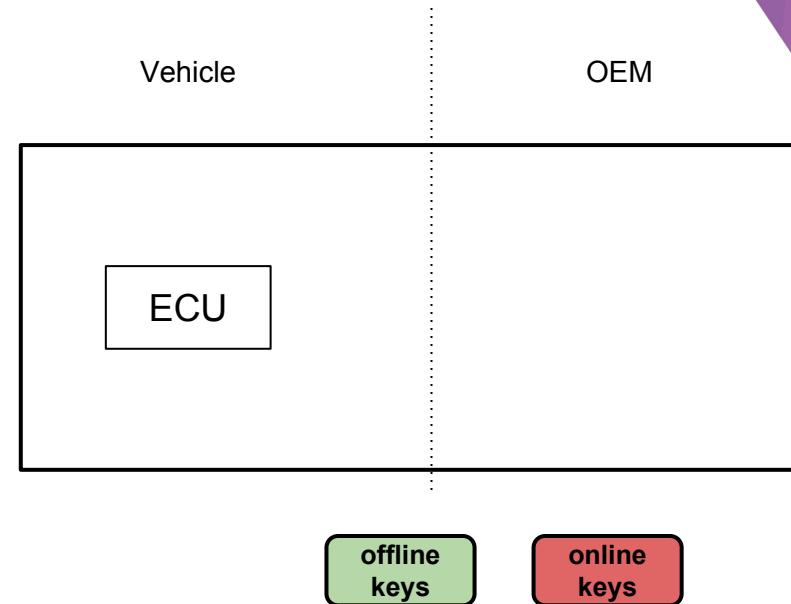
- Use **offline** keys to sign all metadata
- **Pro: compromise-resilient**
 - Attackers cannot tamper with metadata without being detected
- **Con: no on-demand customization**
 - Difficult to install different updates on vehicles of same make and model
 - Cannot instantly blacklist only buggy updates

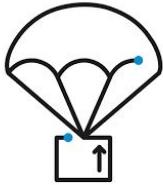




TAP 4: Multi-repository consensus

Solution: Use two repositories

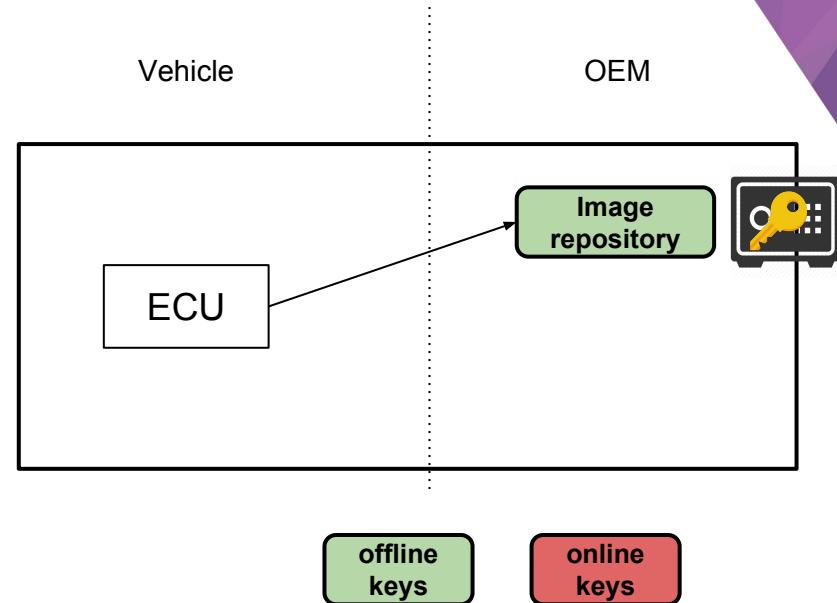


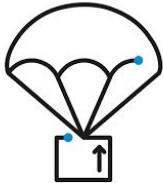


TAP 4: Multi-repository consensus

Solution: Use two repositories

- Image repository
 - Uses **offline keys**
 - Provides signed metadata about all available updates for **all ECUs on all vehicles**

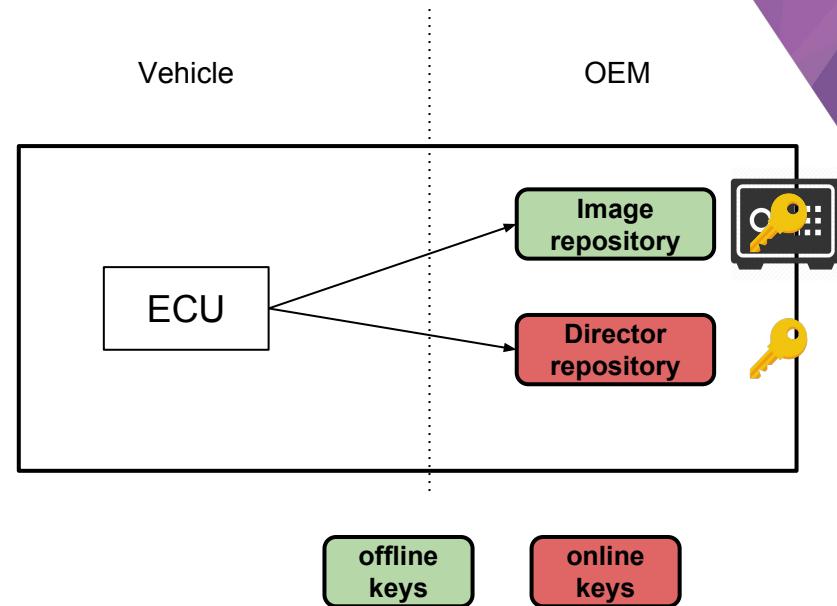




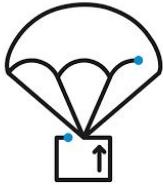
TAP 4: Multi-repository consensus

Solution: Use two repositories

- Image repository
 - Uses offline keys
 - Provides signed metadata about all available updates for all ECUs on all vehicles
- Director repository
 - Uses **online** keys
 - Signs metadata about **which updates** should be installed on **which ECUs** on a vehicle



Cloud native relevance: Nation state attackers



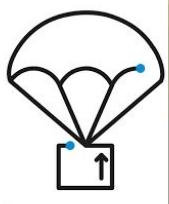
TAP 4: Multi-repository consensus

Strong involvement from automakers [Uptane]

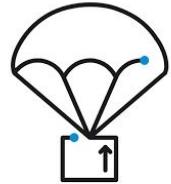
- Work closely with vendors, OEMs, etc.
- Many top suppliers / vendors adopted Uptane in future cars!
 - ~12-35% of cars on US roads
- Automotive Grade Linux
- IEEE / ISTO standardization
 - Vibrant community
 - Dozens of institutions



Cloud Native help from CoreOS (Evan Cordell and Jake Moshenko)

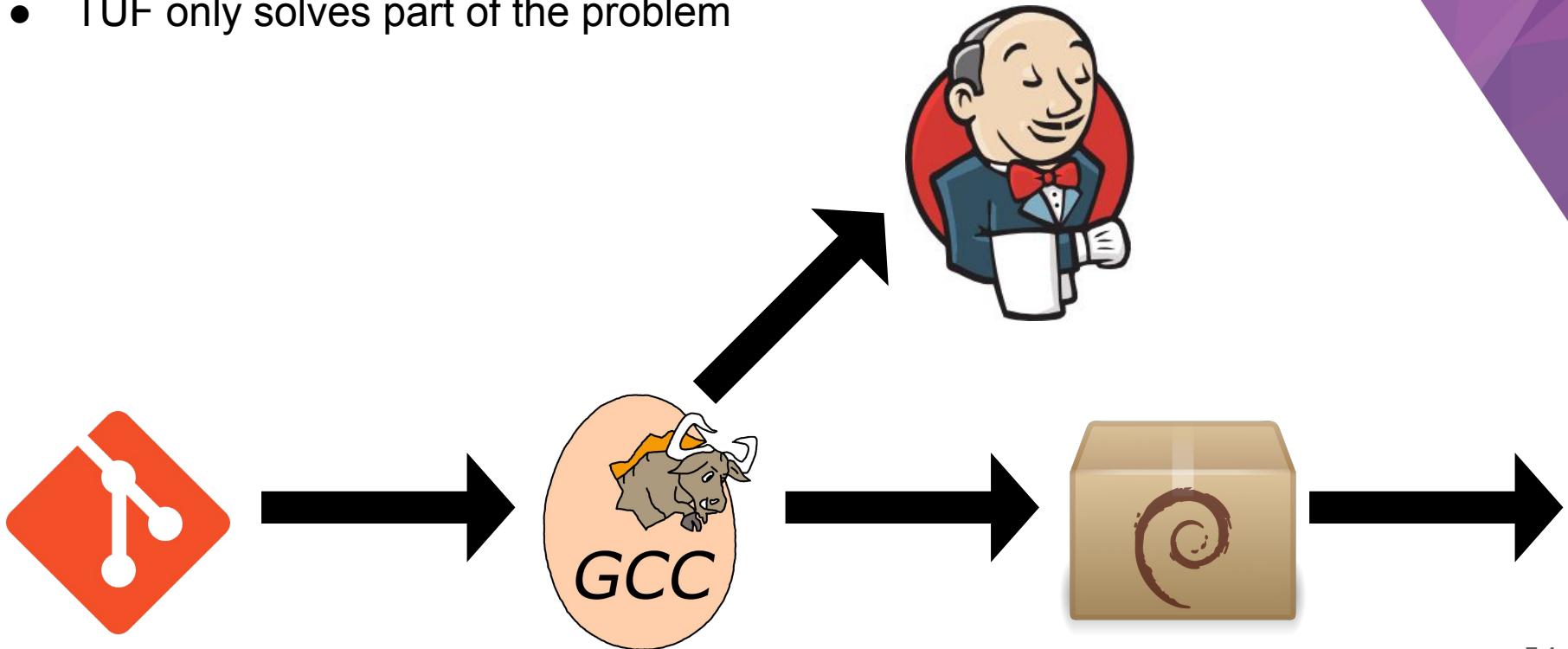


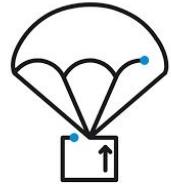
Supply Chain Security with TUF and in-toto



Supply chain security with in-toto

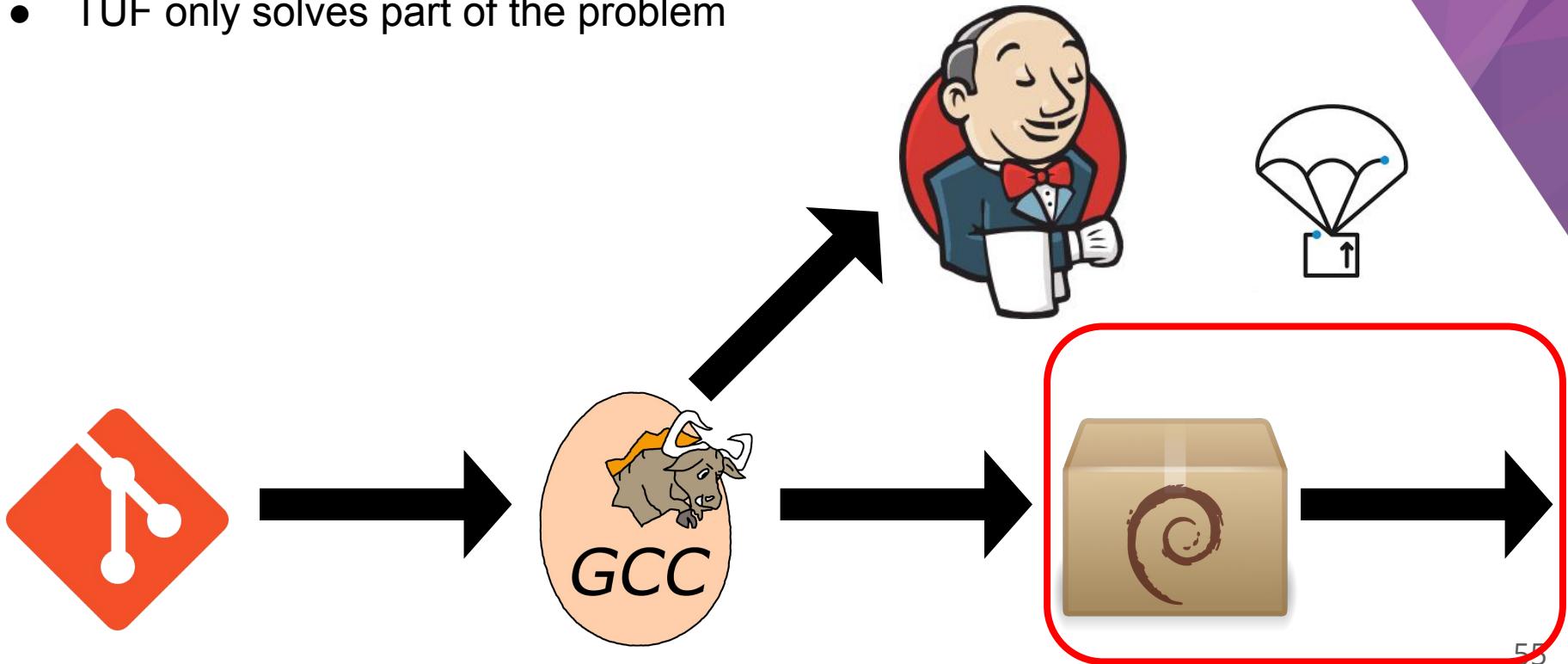
- TUF only solves part of the problem

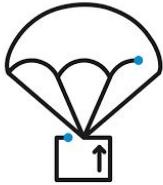




Supply chain security with in-toto

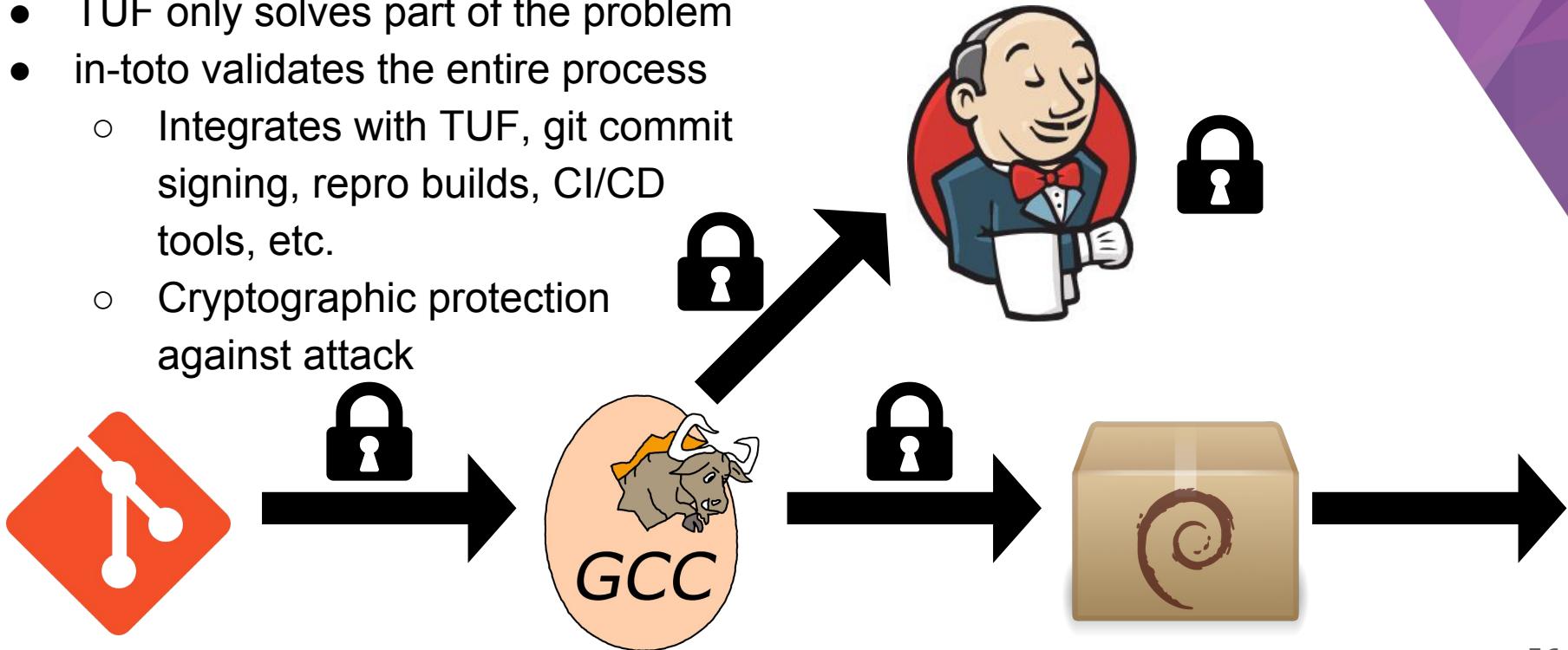
- TUF only solves part of the problem

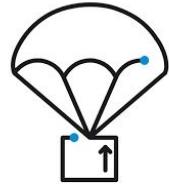




Supply chain security with in-toto

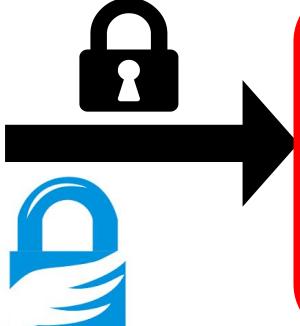
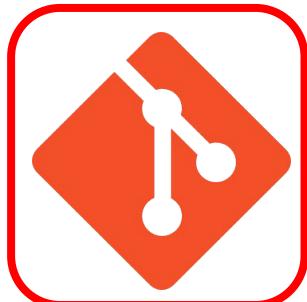
- TUF only solves part of the problem
- in-toto validates the entire process
 - Integrates with TUF, git commit signing, repro builds, CI/CD tools, etc.
 - Cryptographic protection against attack



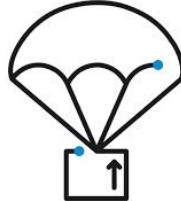


Supply chain security with in-toto

- TUF only solves part of the problem
- in-toto validates the entire process
 - Integrates with TUF, git commit signing, repro builds, CI/CD tools, etc.
 - Cryptographic protection against attack



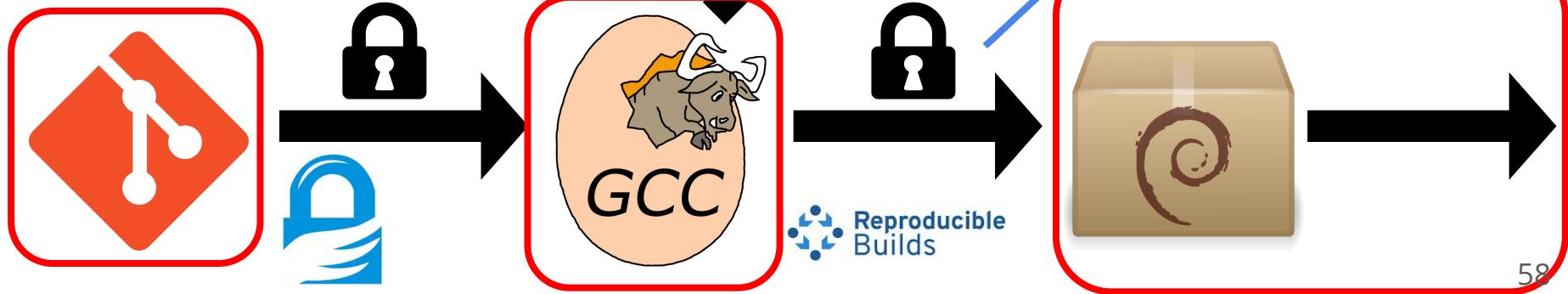
 Reproducible Builds





Supply chain security with in-toto

- TUF only solves part of the problem
- in-toto validates the entire process
 - Integrates with TUF, git commit signing, repro builds, CI/CD tools, etc.
 - Cryptographic protection against attack





Supply chain security with in-toto



Santiago
Torres-Arias



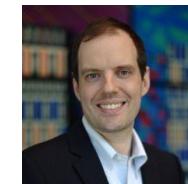
Hammad
Afzail



Lukas
Puehringer



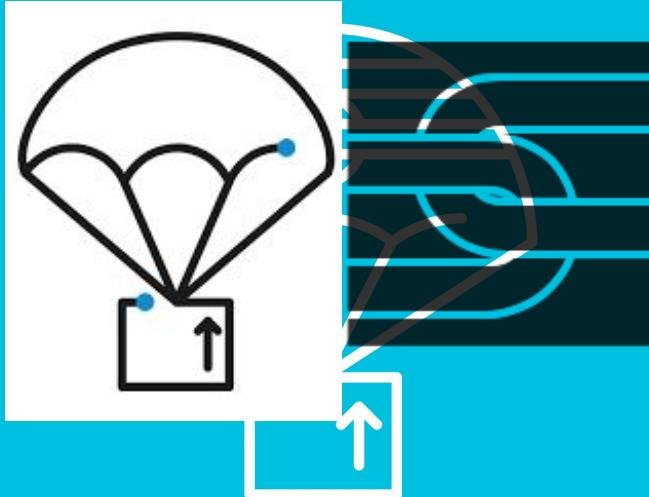
Reza
Curtmola



Justin
Cappos



Why TUF + in-toto





DATADOG

TUF and in-toto in practice:

Datadog Agent Integrations



Datadog, Agent, and Agent integrations

- **3 pillars of Datadog monitoring**
 - Infrastructure metrics
 - App performance
 - Logs
- **Agent**
 - Collects events and metrics
- **Agent integrations**
 - Add-ons / plug-ins
 - > 100 and counting

The screenshot shows the Datadog Integrations page. At the top, there's a navigation bar with links for 'CUSTOMERS', 'ABOUT', 'BLOG', 'LOGIN', and a purple 'GET STARTED FREE' button. Below the navigation is a section titled 'Integrations' with the subtext 'More than 200 built-in integrations. See across all your systems, apps, and services.' A grid of integration categories is displayed in a 4x5 grid:

All	API	AWS	AZURE	CACHING	CHAOS ENGINEERING	CLOUD	COLLABORATION
CONFIGURATION & DEPLOYMENT	CONTAINERS	COST MANAGEMENT	DATA STORE	DIRECT CONNECT			
EXCEPTIONS	GOOGLE CLOUD	HEALTH	ISSUE TRACKING	LANGUAGES	LOG COLLECTION	MESSAGING	
MONITORING	NETWORK	NOTIFICATION	ORCHESTRATION	OS & SYSTEM	PROCESSING	PROVISIONING	
SEARCH	SECURITY	SOURCE CONTROL	WEB				

Below the categories is a search bar with a magnifying glass icon and the placeholder text 'Search for an integration...'. A grid of 10 integration icons is shown, arranged in two rows of five:

Amazon ECS	Amazon EKS	CONSUL	CRI	cri-o
docker	AWS Fargate	etcd	Google Container Engine	kubernetes

Decoupling integrations from Agent release cycle

- **Agent**
 - 6-week release cycle
- **Agent integrations**
 - Latest versions bundled with the Agent every 6 weeks
 - But we also want to publish new versions *independently* of the Agent
 - So customers can beta-test immediately



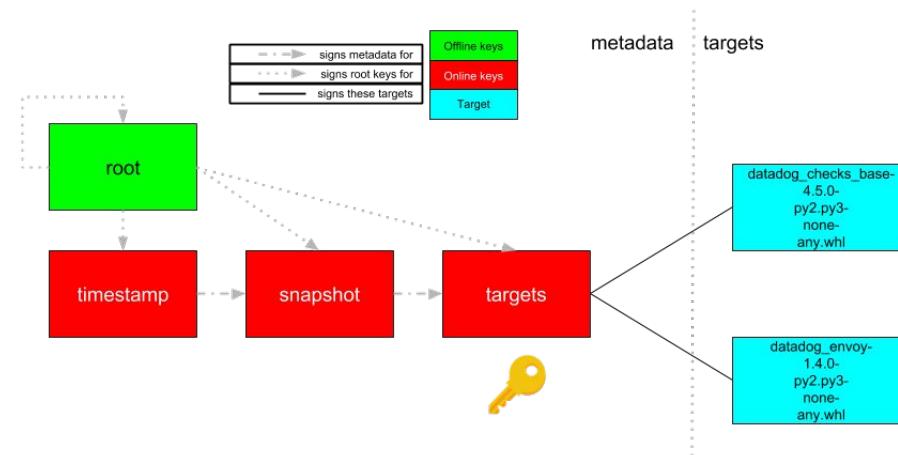
State-of-the-art: CI/CD

- **CI/CD**

- Continuous integration / continuous deployment

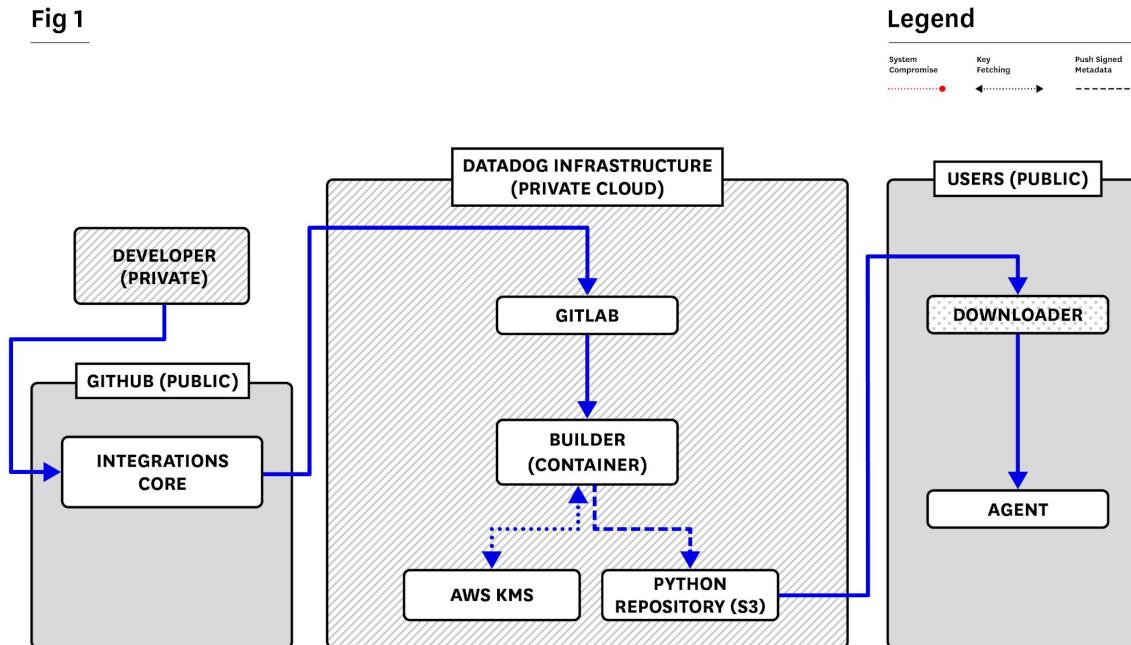
- **Pros**

- Faster deployments
- Clean build environments
- More secure handling of code-signing keys



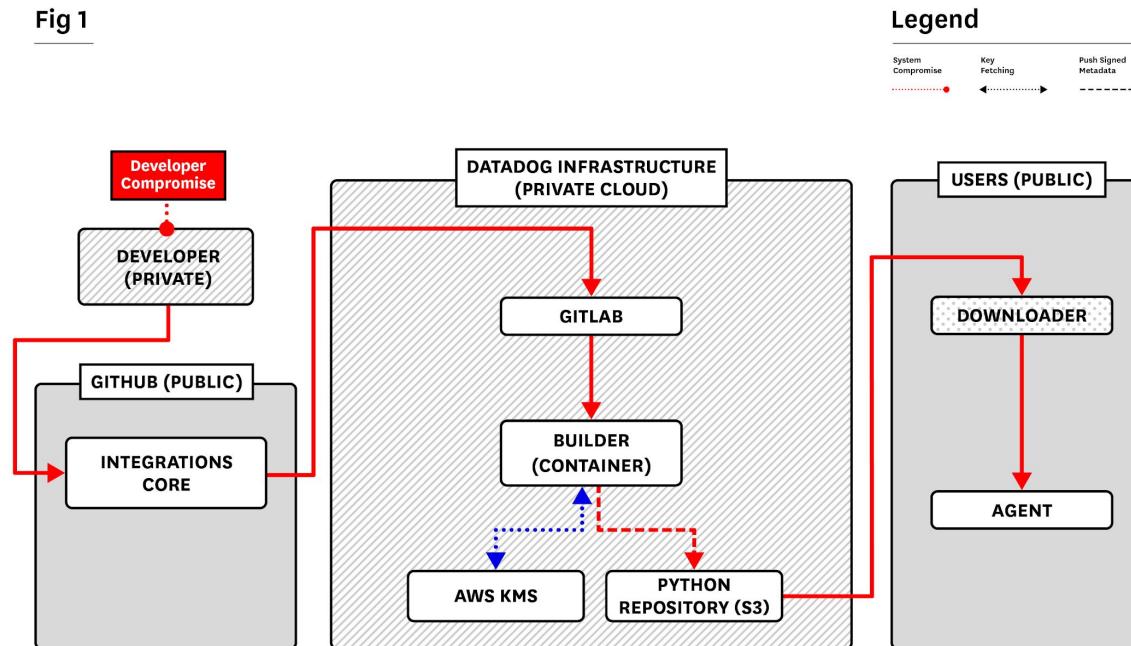
State-of-the-art: what can go wrong?

Fig 1



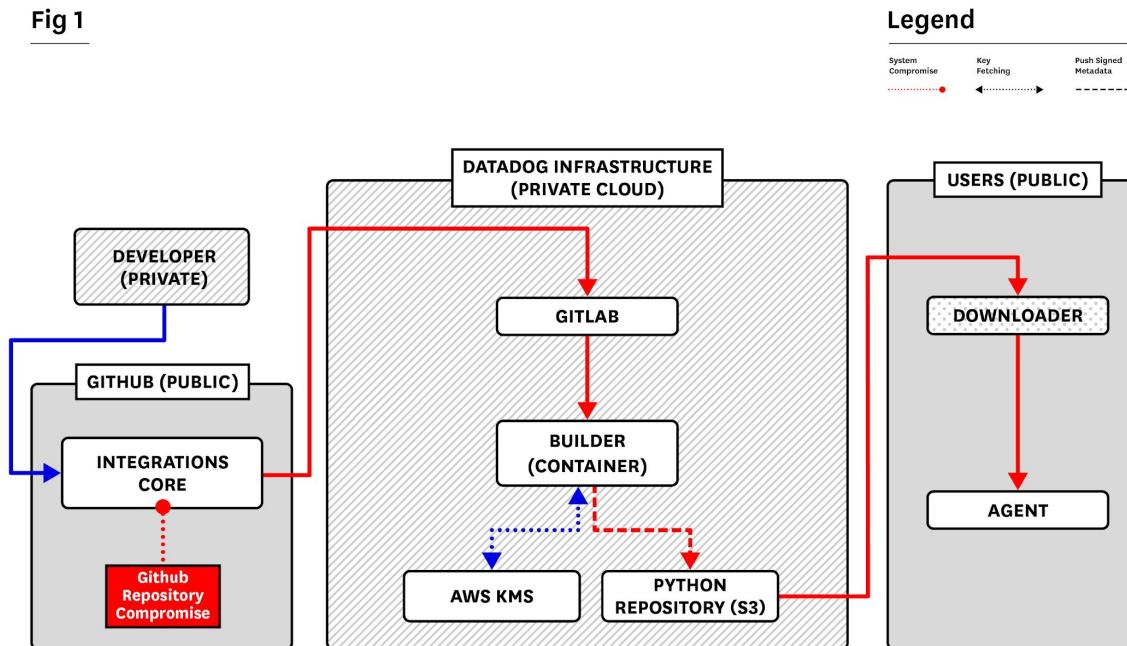
State-of-the-art: developer key compromise

Fig 1



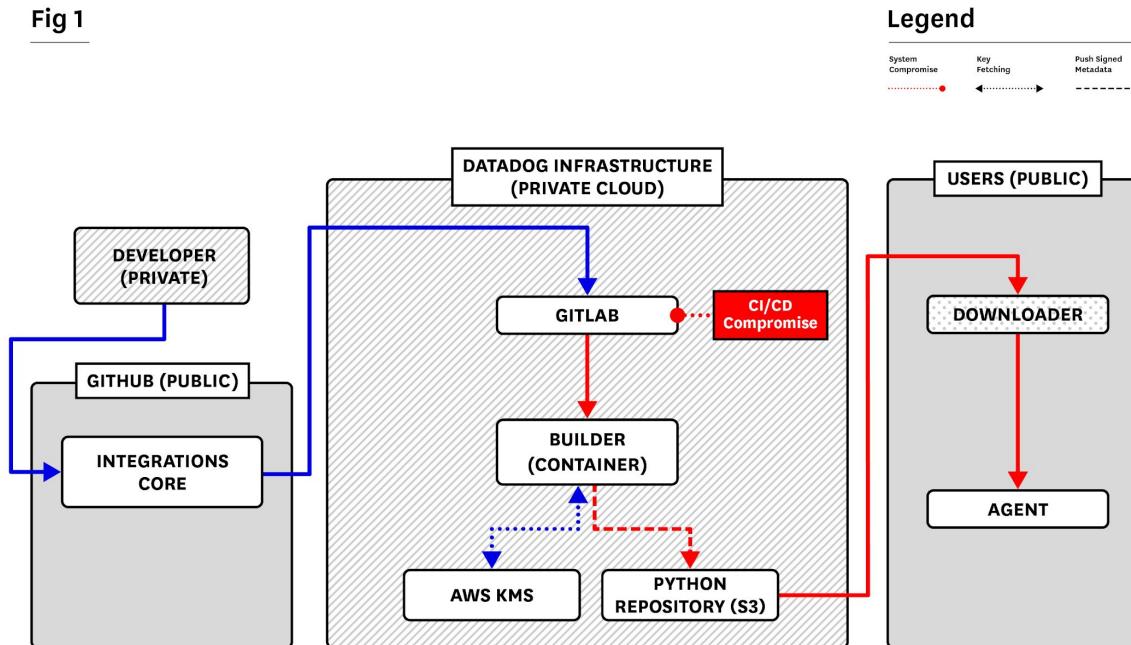
State-of-the-art: VCS repository compromise

Fig 1



State-of-the-art: CI/CD system compromise

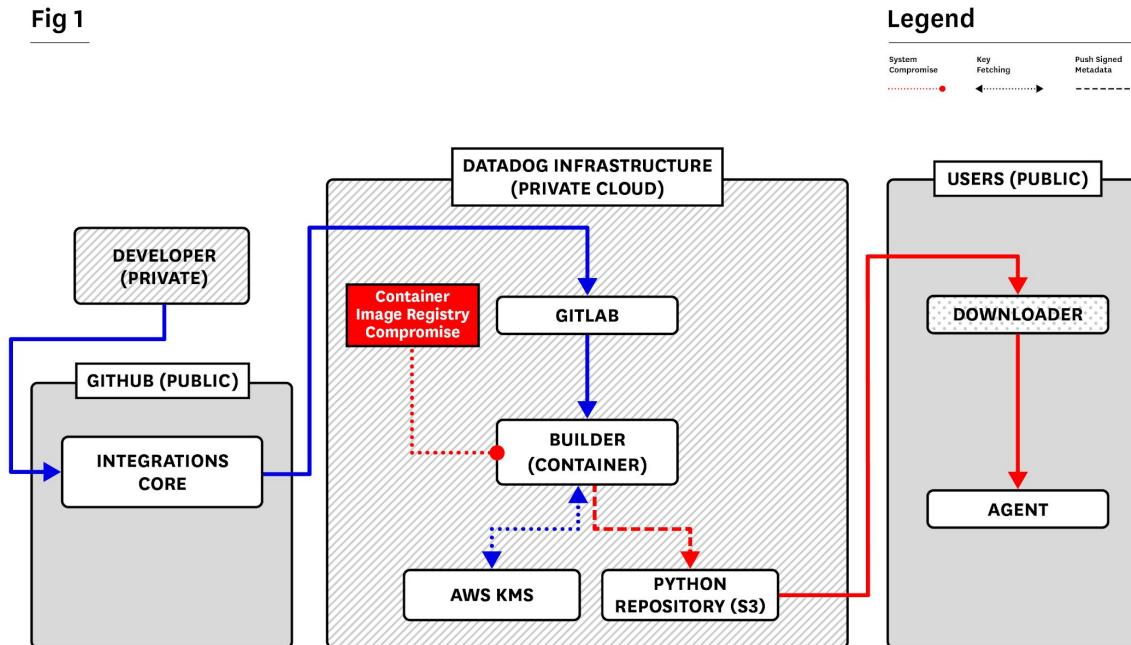
Fig 1





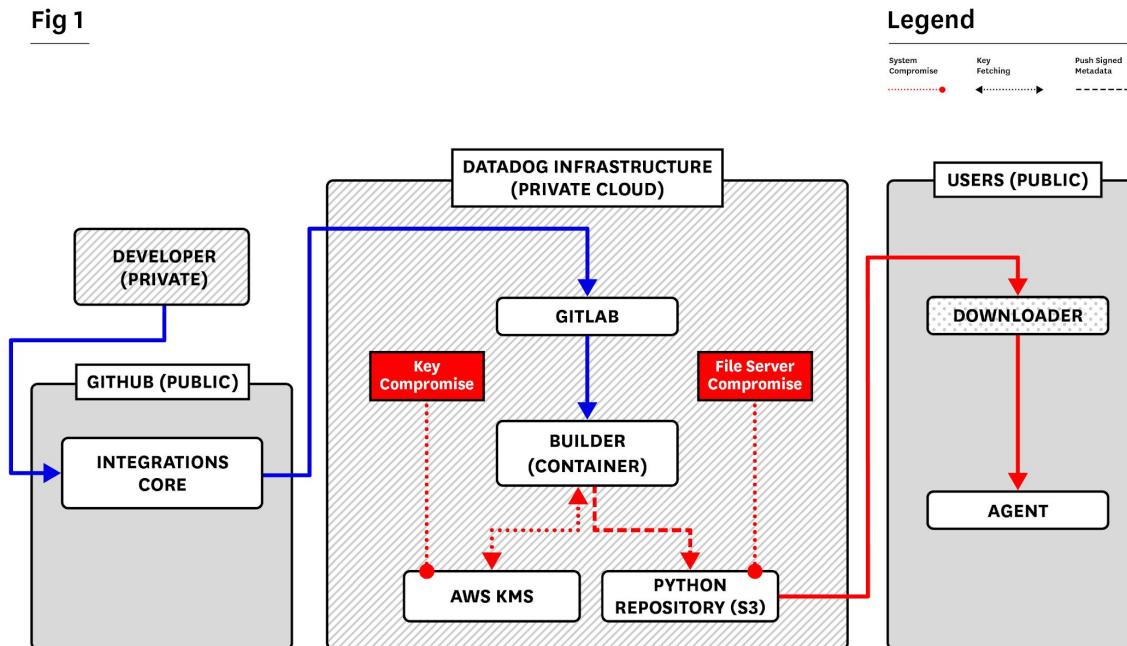
State-of-the-art: container image registry compromise

Fig 1



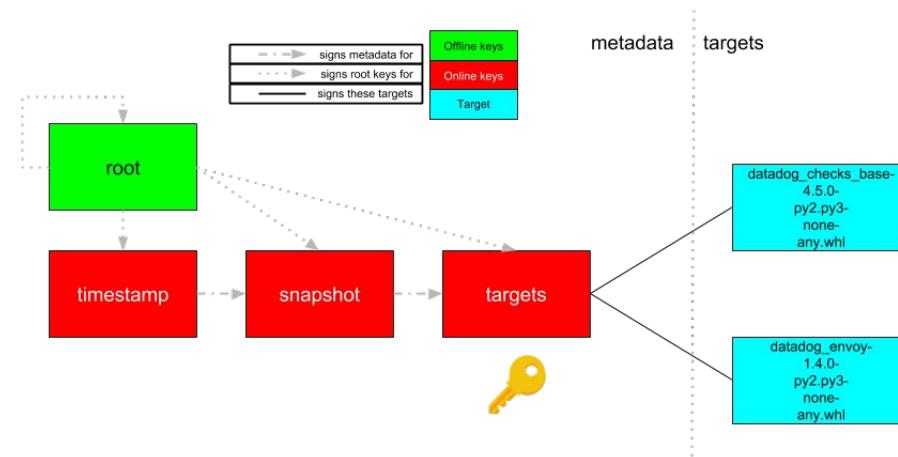
State-of-the-art: key + file server compromise

Fig 1



State-of-the-art: no compromise-resilience

- CI/CD
 - Continuous integration / continuous deployment
- Pros
 - Faster deployments
 - Clean build environments
 - More secure handling of code-signing keys
- Cons
 - **No compromise-resilience**



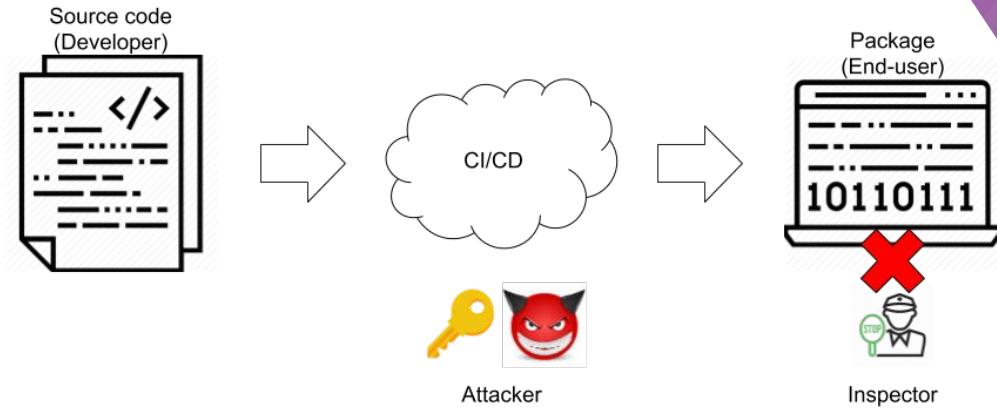
Key idea: tamper-evident CI/CD

- **Tamper-evident**

- $x \Leftrightarrow$ source code
- $f \Leftrightarrow$ authentic CI/CD pipeline
- $y \Leftrightarrow$ package
- Does $y = f(x)$?

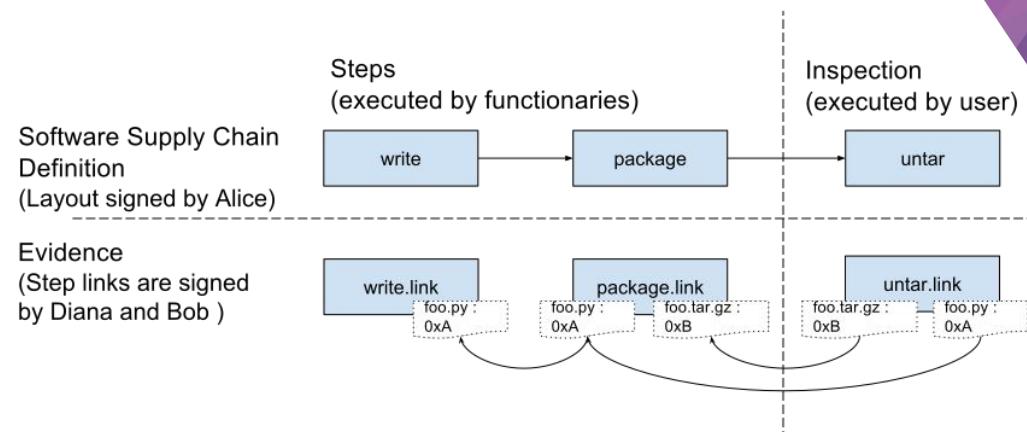
- **Compromise-resilience**

- End-users download x, f , and y
- If $y \neq f(x)$, then reject y



in-toto: software supply chain integrity

- Pipeline = series of **steps**
 - Every step produces signed link / attestation: “I got this input, and produced that output.”
- **Inspection**
 - Verify whether each step followed pipeline
- Provides **E2E verification** of entire supply chain
- <https://in-toto.io>



Datadog Agent integrations software supply chain

1. tag

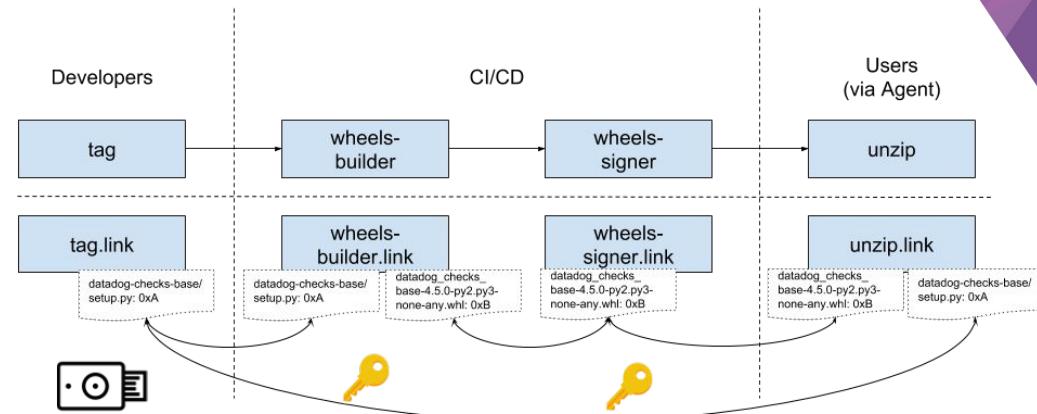
- Developer outputs source code

2. wheels-builder

- Container must receive same source code as in “tag”
- (Container builds wheels)
- Container outputs wheels

3. wheels-signer

- Container must receive same wheels as in “wheels-builder”



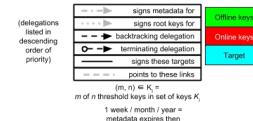
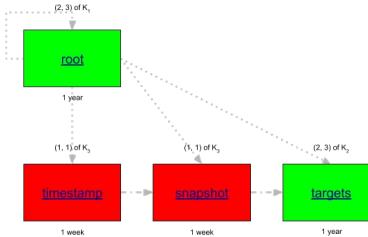


TUF + in-toto = tamper-evident CI/CD

- Offline keys (administrators)
- Semi-offline keys (developers)
- Online keys (CI/CD)

TUF + in-toto = tamper-evident CI/CD

- **Offline keys (administrators)**
 - **TUF root of trust**
- Semi-offline keys (developers)
- Online keys (CI/CD)

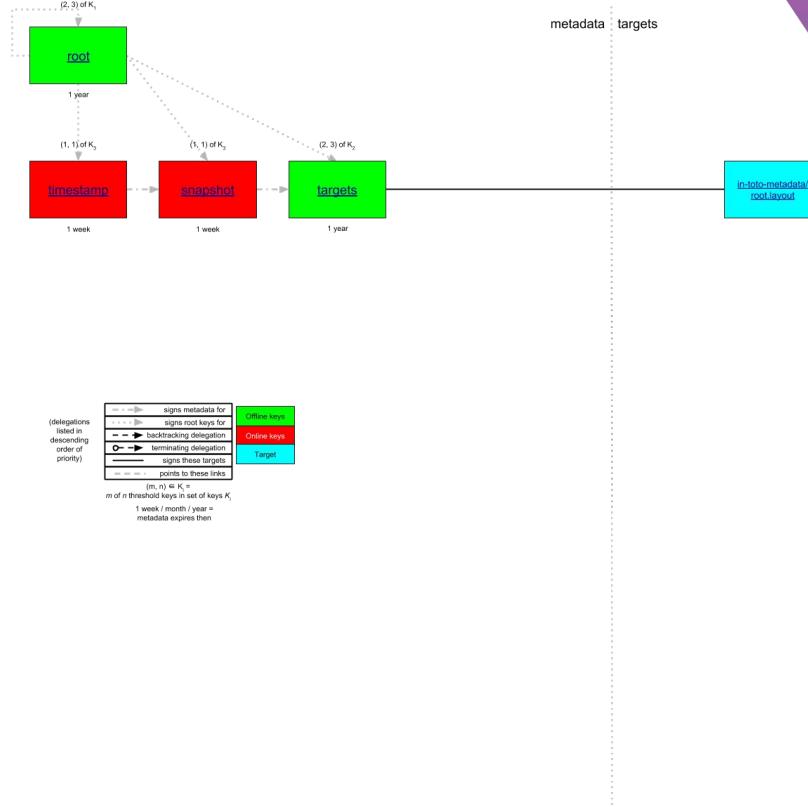


TUF + in-toto = tamper-evident CI/CD

- **Offline keys (administrators)**
 - TUF root of trust
 - **in-toto software supply chain**

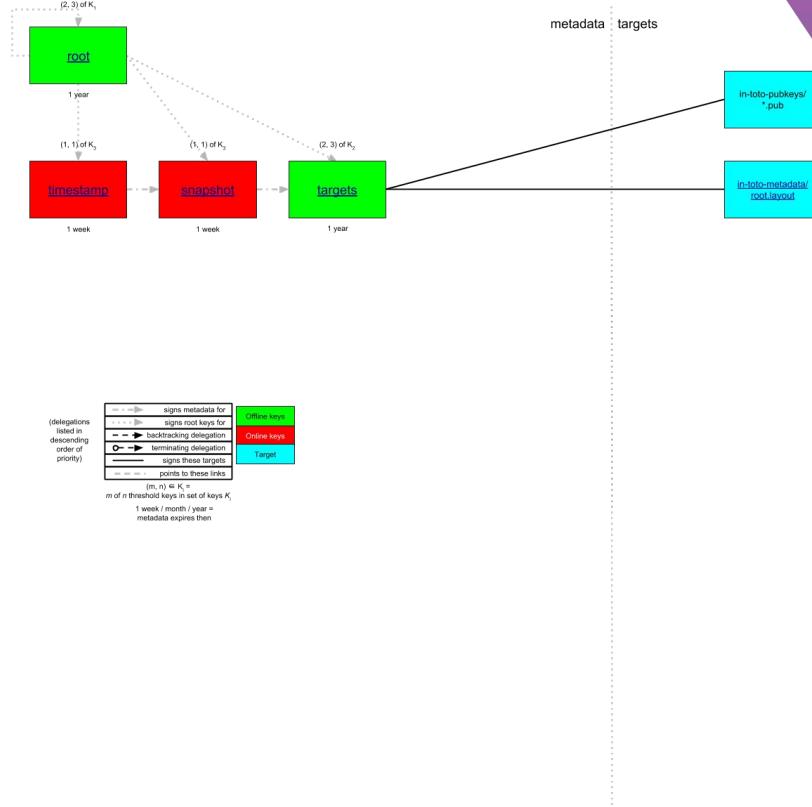
- **Semi-offline keys (developers)**
 - **Python source code**

- **Online keys (CI/CD)**



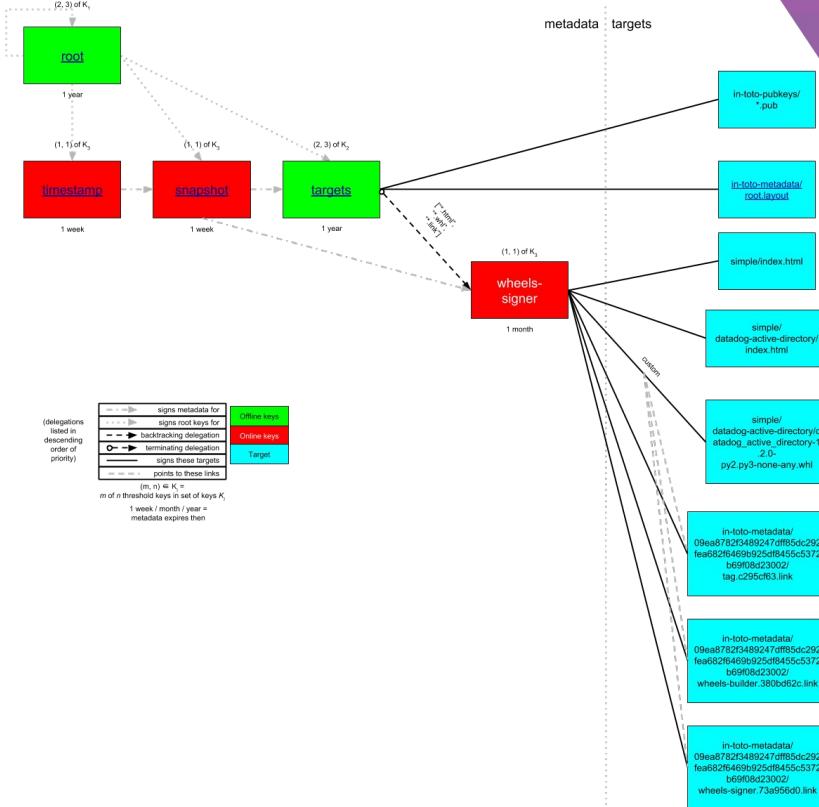
TUF + in-toto = tamper-evident CI/CD

- **Offline keys (administrators)**
 - TUF root of trust
 - in-toto software supply chain
 - **Public keys for in-toto software supply chain**
- Semi-offline keys (developers)
 - Python source code
- Online keys (CI/CD)



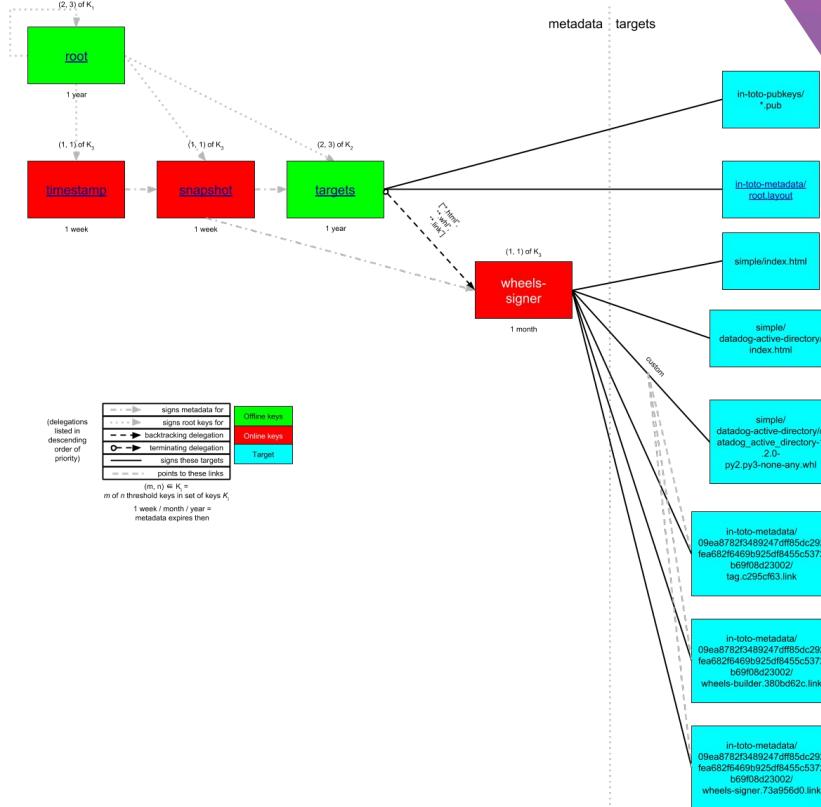
TUF + in-toto = tamper-evident CI/CD

- Offline keys (administrators)
 - TUF root of trust
 - in-toto software supply chain
 - Public keys for in-toto software supply chain
- Semi-offline keys (developers)
 - Python source code
- Online keys (CI/CD)
 - in-toto links
 - Packages
 - (universal Python wheels)**



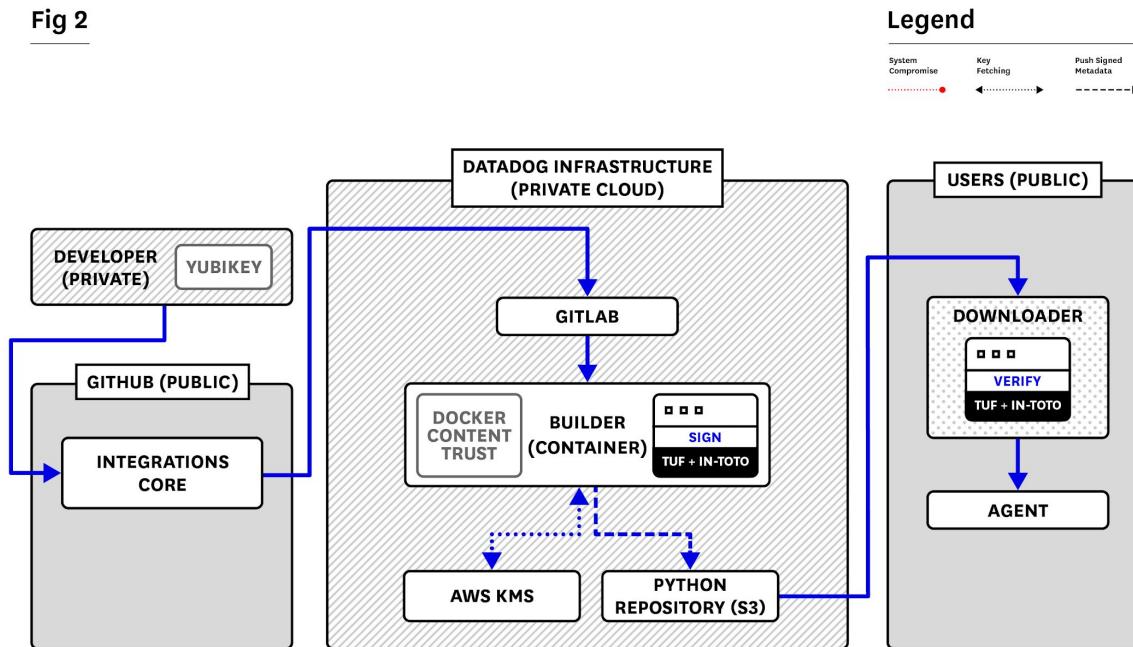
TUF + in-toto = tamper-evident CI/CD

- **Offline keys (administrators)**
 - TUF root of trust
 - in-toto software supply chain
 - Public keys for in-toto software supply chain
- **Semi-offline keys (developers)**
 - Python source code
- **Online keys (CI/CD)**
 - in-toto links
 - Packages (universal Python wheels)



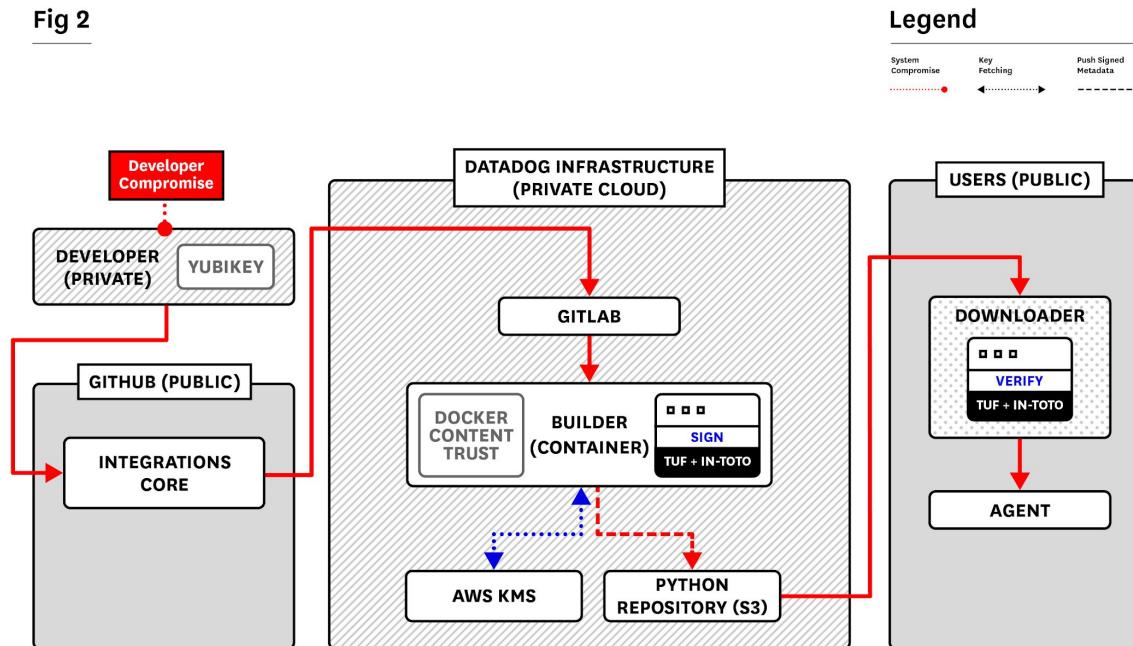
TUF + in-toto: what can go wrong?

Fig 2



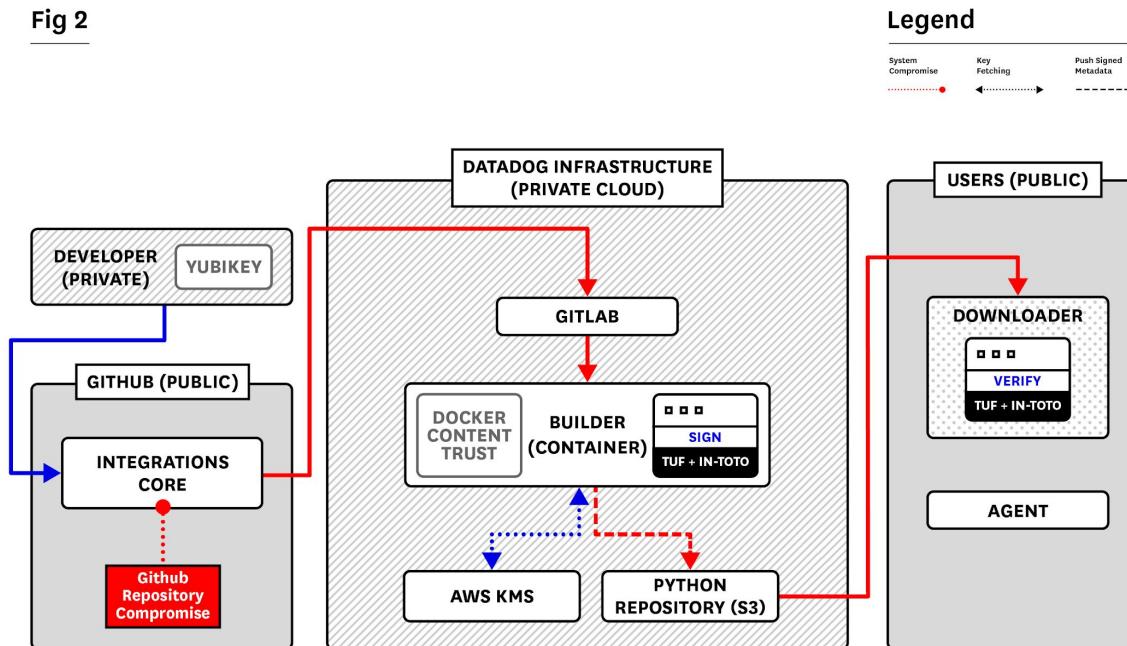
TUF + in-toto: developer key compromise

Fig 2



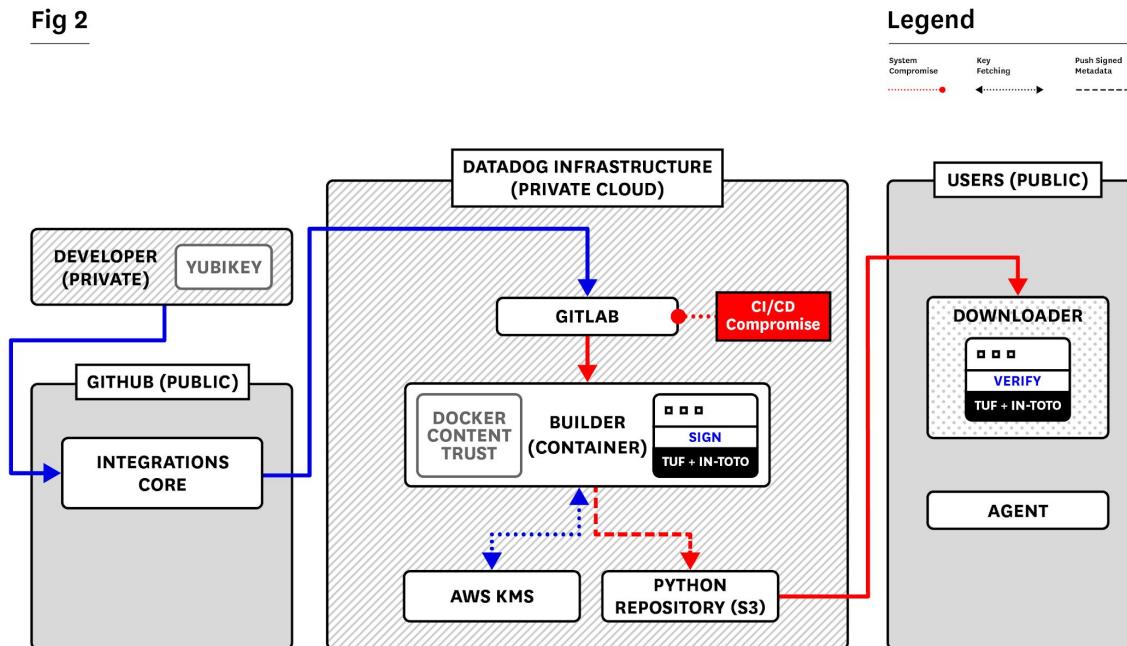
TUF + in-toto: VCS repository compromise

Fig 2



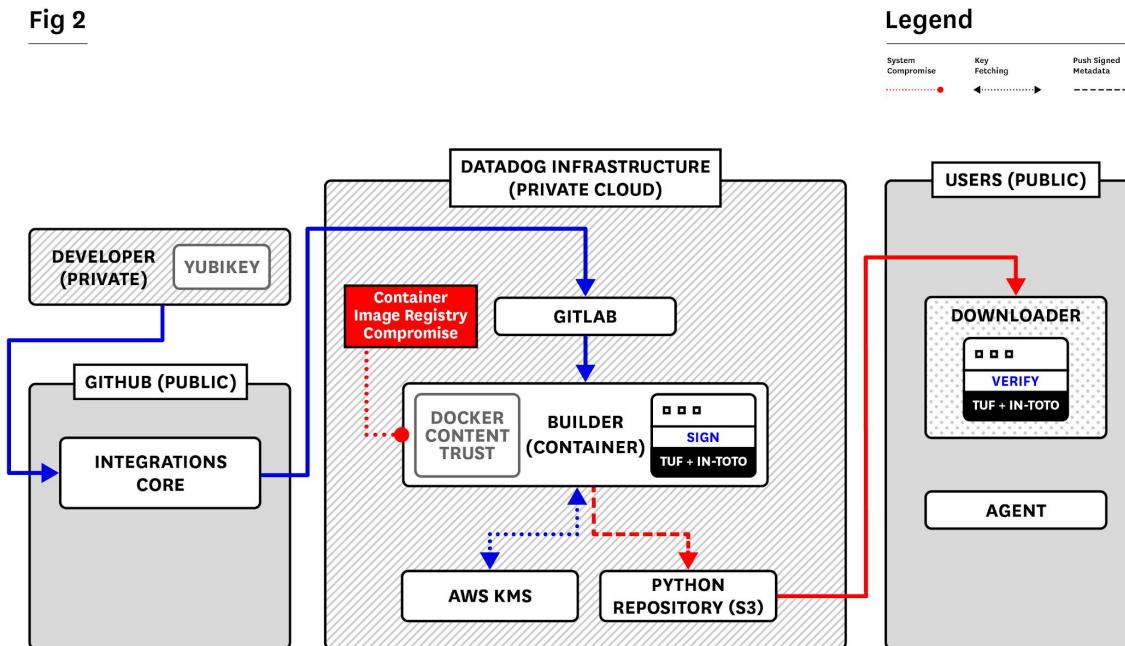
TUF + in-toto: CI/CD system compromise

Fig 2



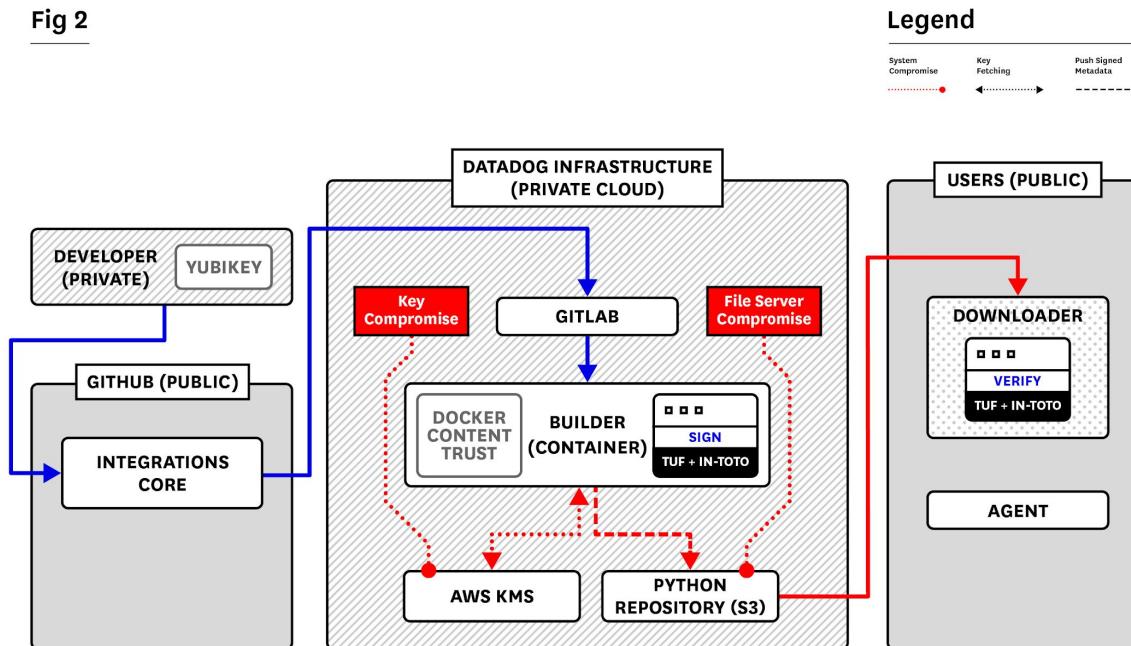
TUF + in-toto: container image registry compromise

Fig 2



TUF + in-toto: key + file server compromise

Fig 2



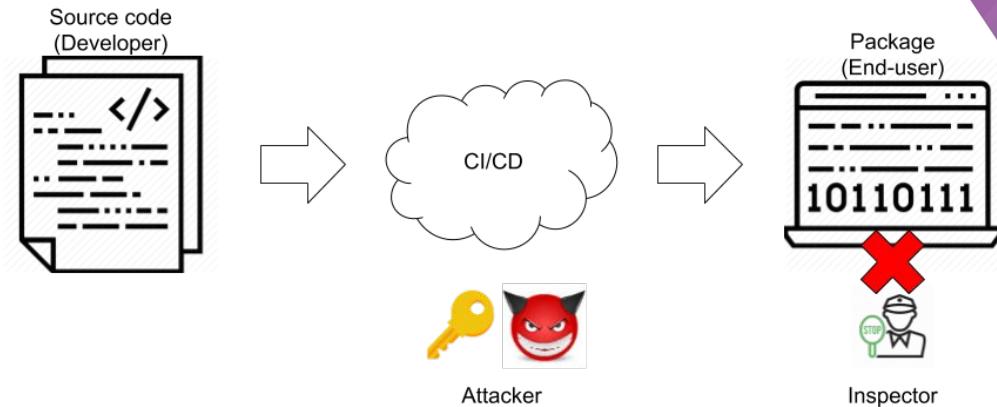


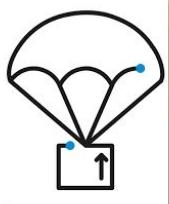
Live demo of production



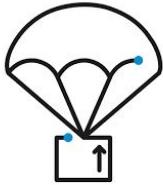
Takeaway: TUF + in-toto = tamper-evident CI/CD

- Tamper-evident
 - $x \Leftrightarrow$ source code
 - $f \Leftrightarrow$ authentic CI/CD pipeline
 - $y \Leftrightarrow$ package
 - Does $y = f(x)$?
- Compromise-resilience
 - End-users download x, f , and y
 - If $y \neq f(x)$, then reject y
- Industry-first
 - Datadog Agent 6.8.0



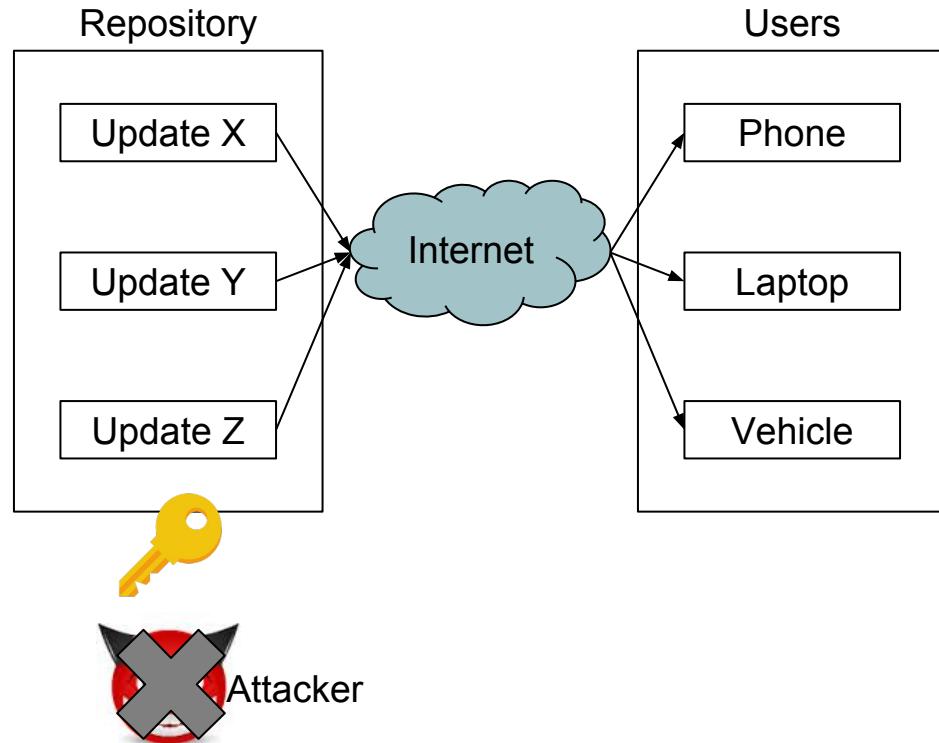


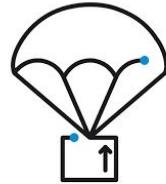
Conclusions



Takeaway: TUF = compromise-resilience

- Only question of when, not if
- Cannot prevent compromise
- But must severely limit impact
- Use TUF





TUF: selected integrations & deployments



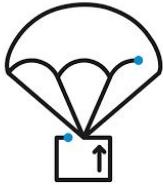
Flynn



Advanced
Telematic
SYSTEMS

Uptane





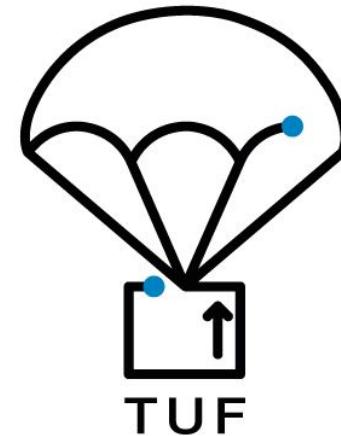
Acknowledgements

- **Datadog**
 - Andrew Becherer, Douglas DePerry, Agent-Integrations, Agent-Core
- **NYU**
 - Sebastien Awwad, Justin Cappos, Lois Anne DeLong, Vladimir Diaz, Lukas Puhringer, Santiago Torres-Arias
- **Docker**
 - Nathan McCauley, Diogo Monica, David Lawrence, Justin Cormack
- **CoreOS**
 - Evan Cordell, Jacob Moshenko
- **Uptane**
 - Uptane Alliance



Q & A

- Thanks for your time!
- TUF: <https://theupdateframework.com>
- in-toto: <https://in-toto.io/>
- Email: trishank@datadog.com [DataDog, TUF]
- Email: jcappos@nyu.edu [TUF, in-toto]
- Yubikey: <https://github.com/DataDog/yubikey>



NYU

TANDON SCHOOL
OF ENGINEERING