Conex

Hannes Mehnert, https://hannes.robur.coop

June 11th 2021

What is this about?

- Why is conex still not deployed?
- Next steps for integration of conex into opam
- OCaml Software Foundation supports the development

Brief interlude - what is conex?

- Establishing a trust relation between author of a library and user thereof
- No single point of failure / accumulated trust
- I.e. when compromising any key, impact is the packages delegated to that key
- Delegation to quorum (n out of m) of keys (ocaml core team, MirageOS team)
- Local repository is trusted

What are the views?

- OCaml software user (using opam)
- OCaml software developer (library author)
- Opam repository maintainer (janitor)

User

- Would like to have a secure supply chain
- Inside a company, an overlay repository
- Custom forks, custom trust relations

- Signs releases (dune-release & opam publish integration)
- Needs to enroll their key (sigstore `https://sigstore.dev/`)
- May need to overwrite trust relations locally

Opam repository maintainer

- Applies fixes (adjust lower bounds, ..)
- Shouldn't require the software author to re-sign
- Delegates responsibility of package names to authors
- Quorum signature, otherwise a single maintainer key compromise would be fatal

Current user workflow

- opam repository is a git repository, downloaded as tarball
- "opam update" retrieves a diff between local data and remote data
- this workflow should not change
- validation command `https://opam.ocaml.org/doc/Manual.html#configfield-repository-validation-command`

Security goals of conex

- Avoid "arbitrary installation" (only signed opam packages to be installed - tarballs are signed)
- Avoid "extraneous dependencies attack" (an attacker may not inject more dependencies - metadata (opam file) is signed)
- Avoid "fast forward attacks" (counters will be used, moving it to maximum value by a key compromise should not lead to irrecoverable situation)
- Avoid "indefinite freeze attack" (attacks that avoid updates are detected)
- Avoid "mix and match attack" (only some package updates are provided to the client)
- Reduce "key compromise vulnerability" (no key has superuser privileges)
- Avoid "wrong software installation" (an attacker cannot inject arbitrary data)

Goals for the PKI

- Keys are revokable (compromise, loss of key) and can be rolled over (renewal)
- Both by the key owner and by a quorum / re-release of conex/opam
- Package responsibility can be delegated to authors (with quorum signatures, delegations can be changed)

- Two milestones
- low security (online system, snapshot service, milestone 1)
- high security (authors sign releases, milestone 2)

Snapshot service (milestone 1)

- "Indefinite freeze" and "mix and match" require systems that sign the entire opam repository
- Snapshot service: periodically (every 5 minutes) retrieves the opam repository, validates an update, and signs
- Client can verify (a) signature is valid (b) update is new enough (since it knows every 5 minutes there is a new signature)
- Error handling: snapshot service needs to inform author / maintainer of invalid signature, and re-sign every 5 minutes
- The signed repository needs to be manually checked (avoid misbehaving snapshot service, attacks on repo-snapshot communication)

Relevant data

- Public keys, file hashes and signatures
- Stored in opam-repository
- Q: If extra-sources / extra-files / patches are used, and more (fewer) files are in the "files" subdirectory, what does opam do?
- Q: What does opam do if checksum mismatches?
- Q: Can we upgrade to SHA-256 for these checksums?

Milestone 2 - full end-to-end signatures and verification

- Tooling for repository maintainers (for quorum signatures)
- Tooling for authors (to sign their commits)
- Tooling for teams (to enroll and remove new team members)
- Trust overlay for one opam repository
- Multiple opam repositories (how is trust managed, how are packages selected)
- CI tooling for opam-repository that verifies signatures and delegations

- Conex compiles, snapshot service under development
- Once that is deployed, we need testers
- Details need to be sorted (how to do quorum signatures, how to enroll keys, sigstore integration)
- Authoritative information for "maintains a package Y" (the opam metadata is not the reality)

What to discuss?

- Workflows (maintainers, package authors)
- Opam checksums of extra files
- Data format (at the moment, "keys/" and additional files in the packages)
- Quorum signatures
- Key enrollment
- Multiple opam repositories and trust delegations
- CI to cryptographically sign results (reproducible builds, supply chain)