Security Architecture Notes and Patterns

Table of Contents

This is the pages version of Stefan Schumacher's Codeberg Repository at https://codeberg.org/0xKaishakunin/Architecture

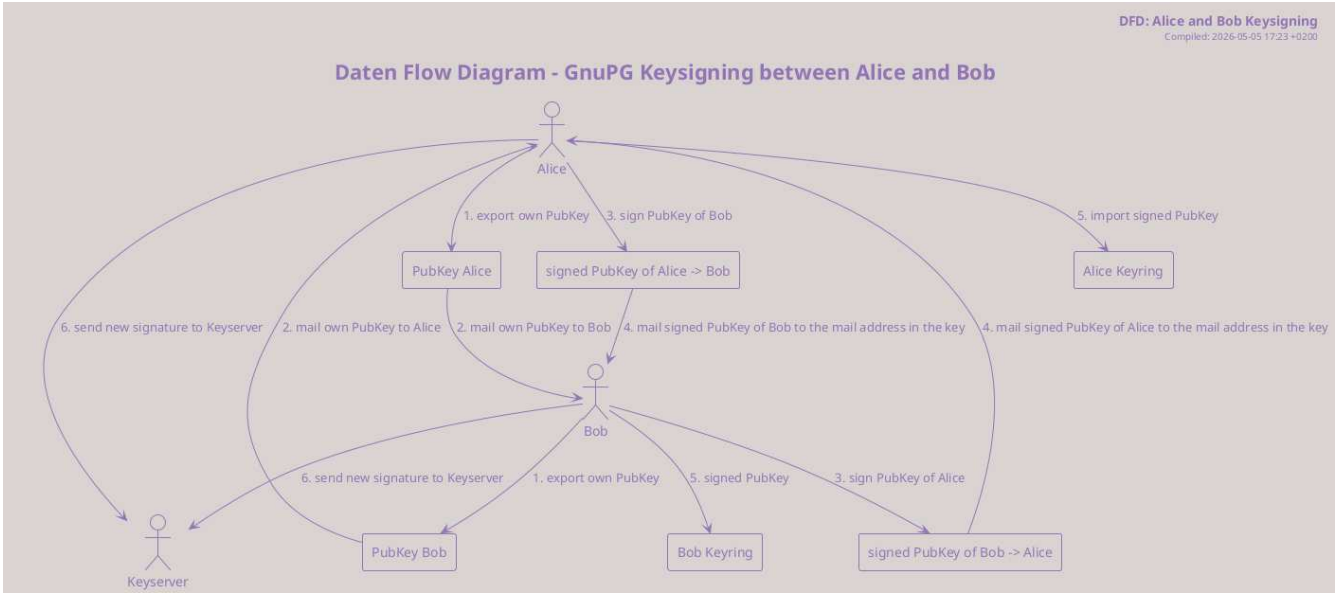
It contains several architecural patterns and exercises with PlantUML with regards to modeling trust and zero trust architecture patterns

1. From GnuPG Keysigning Party to Reproducile Builds according to SLSA4

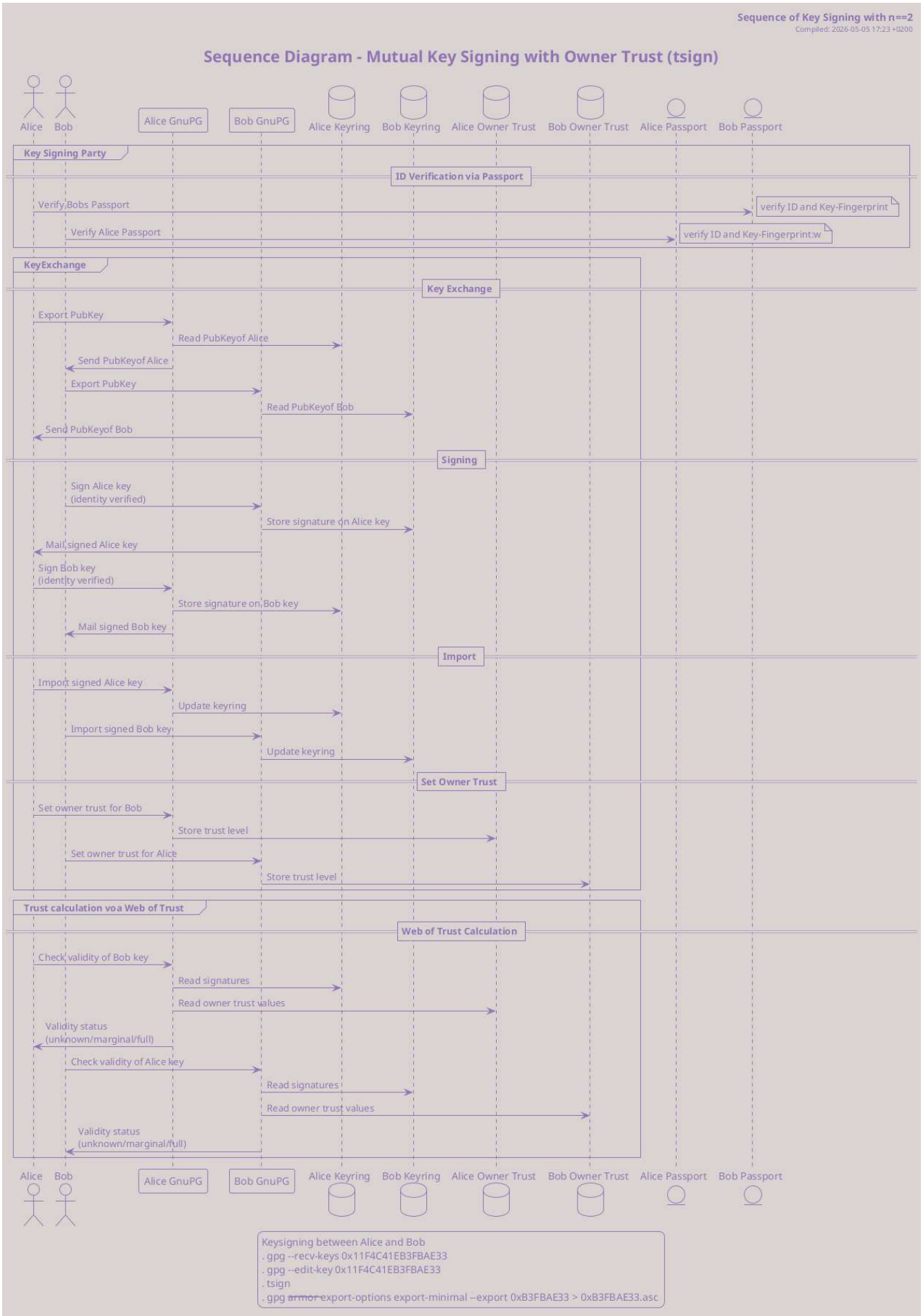
I am trying to model implicit and explicit trust in Zero Trust Architecture diagrams for Threat Modeling.

So I need to bring together the Blue Team/White Hat perspective and the Red Team/Black Hat stuff.

2. DFD: Keysigning Simple



3. SEQ: Keysigning with ID check and WoT


Sequence of Key Signing with n==2
Compiled: 2026-05-05 17:23 +0200
Sequence Diagram - Mutual Key Signing with Owner Trust (tsign)
Alice
Bob
Alice GnuPG
Bob GnuPG
Alice Keyring
Bob Keyring
Alice Owner Trust
Bob Owner Trust
Alice Passport
Bob Passport
Key Signing Party
ID Verification via Passport
Verify Bobs Passport
verify ID and Key-Fingerprint
Verify Alice Passport
verify ID and Key-Fingerprint:w
KeyExchange
Key Exchange
Export PubKey
Read PubKeyof Alice
Send PubKeyof Alice
Export PubKey
Read PubKeyof Bob
Send PubKeyof Bob
Signing
Sign Alice key
(identity verified)
Store signature on Alice key
Mail signed Alice key
Sign Bob key
(identity verified)
Store signature on Bob key
Mail signed Bob key
Import
Import signed Alice key
Update keyring
Import signed Bob key
Update keyring
Set Owner Trust
Set owner trust for Bob
Store trust level
Set owner trust for Alice
Store trust level
Trust calculation voa Web of Trust
Web of Trust Calculation
Check validity of Bob key
Read signatures
Read owner trust values
Validity status
(unknown/marginal/full)
Check validity of Alice key
Read signatures
Read owner trust values
Validity status
(unknown/marginal/full)
Keysigning between Alice and Bob
. gpg --recv-keys 0x11F4C41EB3FBAE33
. gpg --edit-key 0x11F4C41EB3FBAE33
. tsign
. gpg armor export-options export-minimal --export 0xB3FBAE33 > 0xB3FBAE33.asc

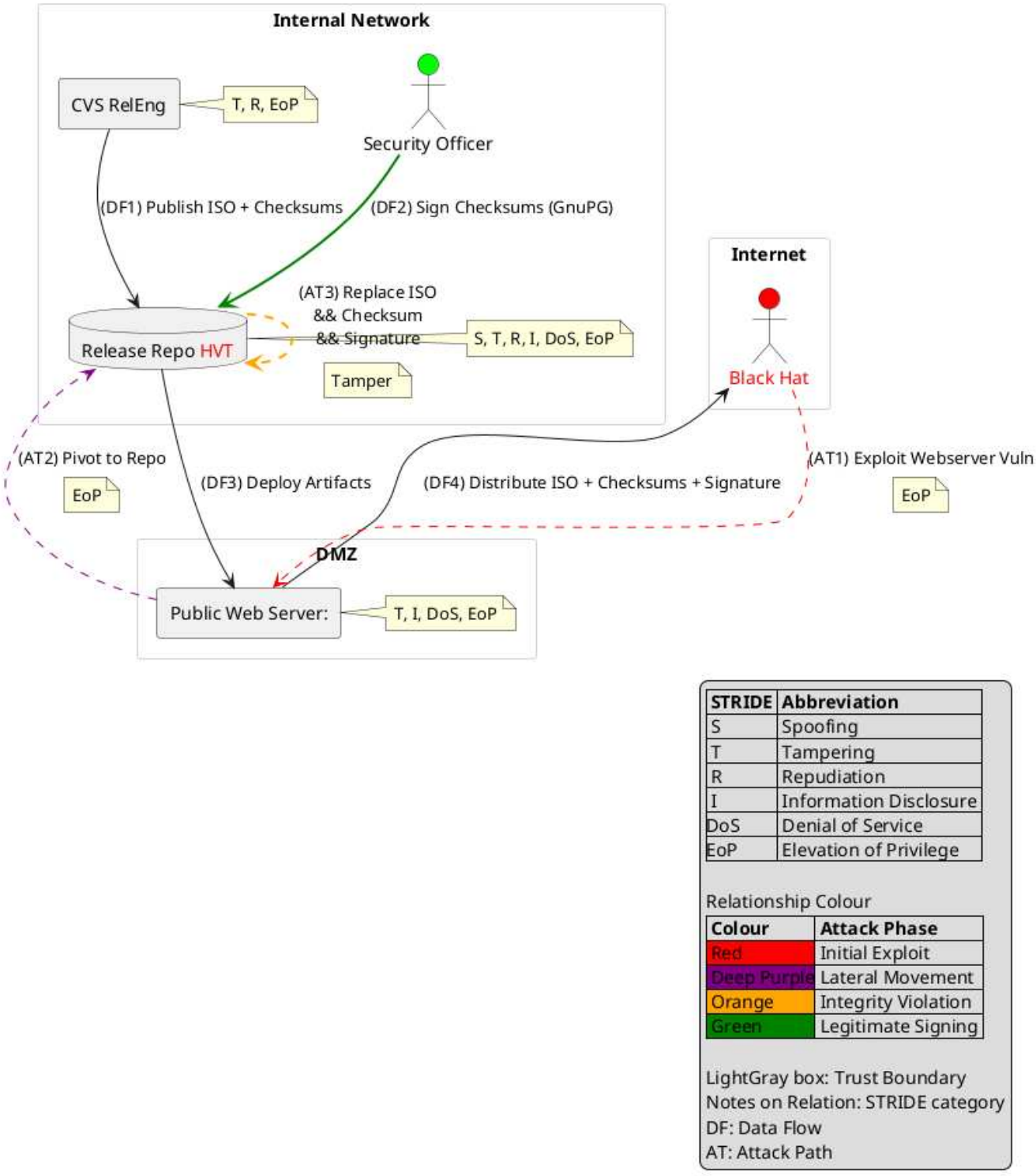
4. NetBSD RelEng Attack STRIDE

The ISO Image is built, signed and uploaded to the WWW server, as well as the Signature file and checksums.

Evil Black Hat hacks the webserver, and swaps the ISO image for a manipulated one with a valid Signature.

The manipulated Signature verifies the fake-integrity of the manipulated ISO image, but not the authenticity.

Attack Flow and Trust Boundaries: manipulated ISO image AND manipulated checksums/signatures (SLSA1)



STRIDE	Abbreviation
S	Spoofing
T	Tampering
R	Repudiation
I	Information Disclosure
DoS	Denial of Service
EoP	Elevation of Privilege

Relationship Colour

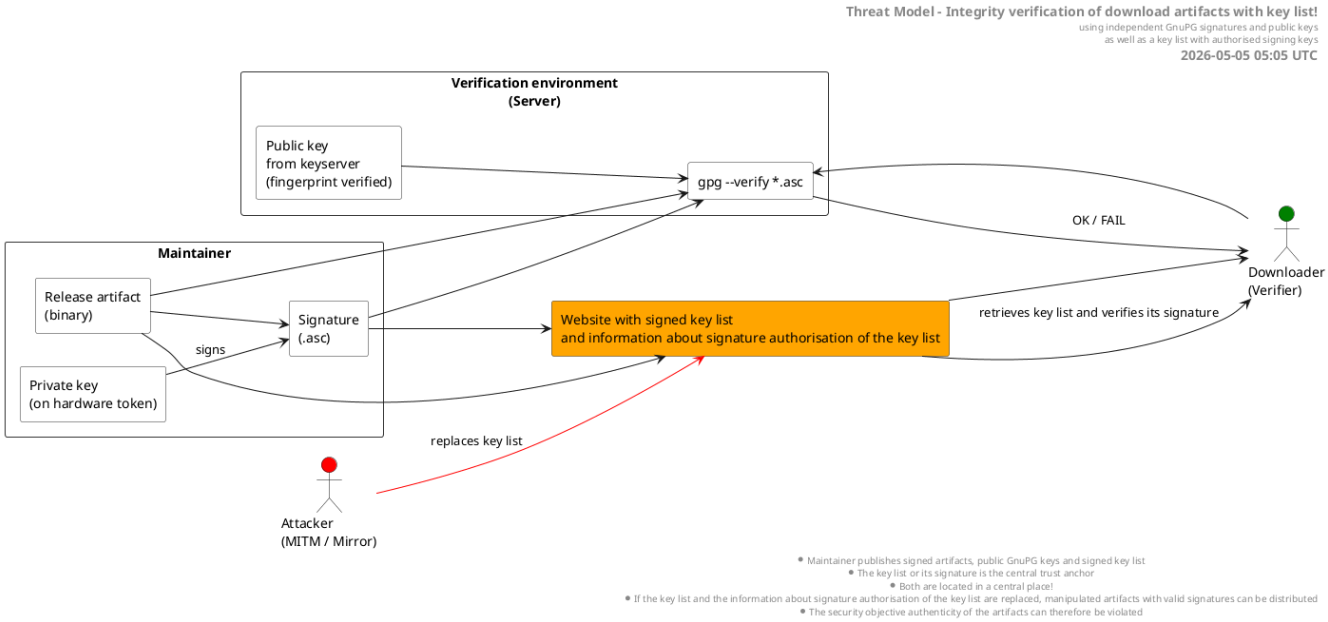
Colour	Attack Phase
Red	Initial Exploit
Deep Purple	Lateral Movement
Orange	Integrity Violation
Green	Legitimate Signing

LightGray box: Trust Boundary
Notes on Relation: STRIDE category
DF: Data Flow
AT: Attack Path

5. Supply-chain Levels for Software Artifacts

5.1. SLSA L1

A simple threat model for SLSA Level 1:



5.2. Reproducible Builds SLSA L4

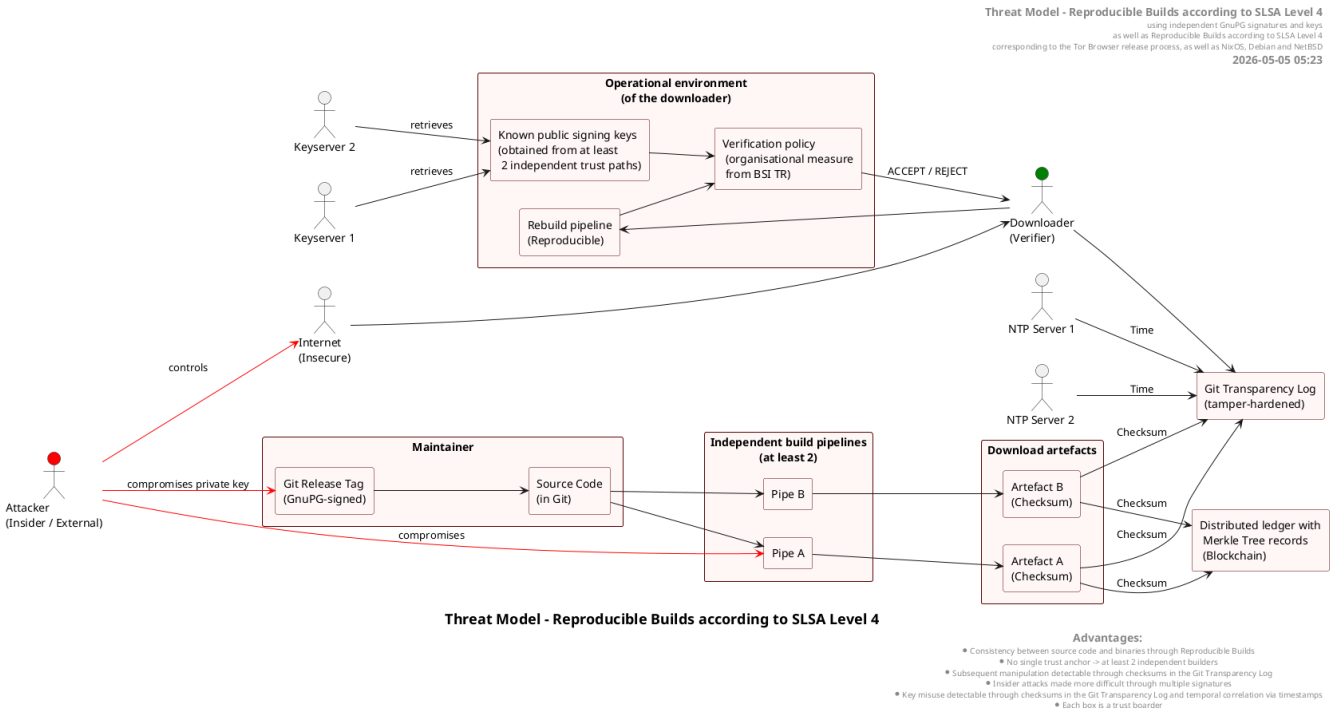
Modeling Trust, Trust Anchors and Boundaries and Attack Vectors for SLSA4:

The whole process draws heavy inspiration from those implemented by NetBSD, Debian, NixOS and the Tor Browser!

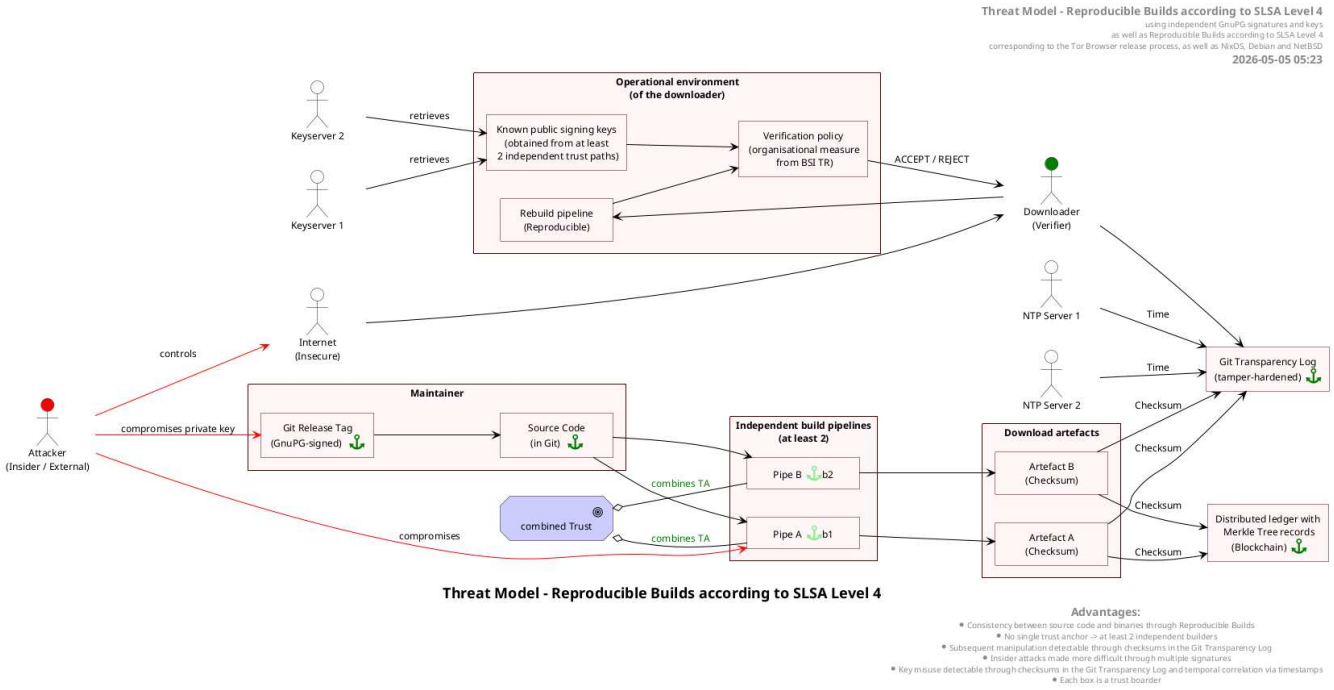
1. general goals
    a. Build process produces identical artefacts (bit-for-bit) from the same source and inputs
    b. independent parties can rebuild and verify outputs match the original → verify freedom from insider threat!
    c. require a deterministic build environments
    d. all build steps, dependencies, and tooling are tightly controlled and audited
2. security goals:
    a. detect tampering in build pipelines or artefacts
    b. prevents hidden backdoors introduced during compilation or packaging by a malicious insider
    c. ensure integrity of supply chain, dependencies and build tools
    d. enables independent verification without trusting the original builder
    e. drastrically reduce insider and supply chain attack surface
3. Zero Trust:
    a. never trust, always verify!
    b. verifiable evidence (rebuild && compare)
    c. eliminates implicit trust in build pipeline
    d. build system considered untrustworthy
    e. combine with signed artefacts and attestation frameworks for full supply chain integrity

5.2.1. The whole implementation

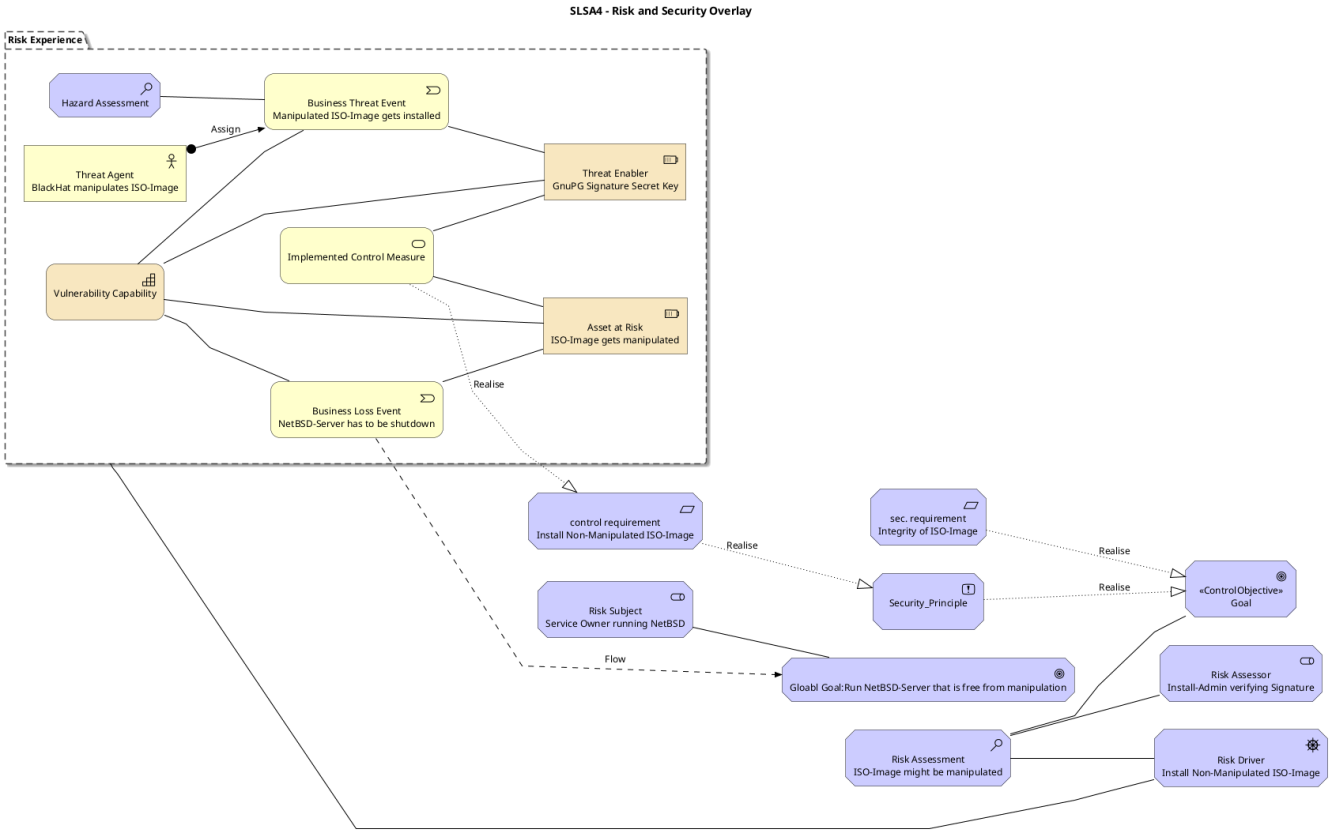
- Trust Boundaries: Rectangles
- Attack Vectors: Red Arrows



- Trust Anchor: Green Anchor



5.3. Motivation/Business Layer: Risk/Security Overlay



6. Links

1. GetPlantUML-CheatSheets.sh
   a. downloads some useful ArchiMate cheat sheets/overviews/examples via `wget`
2. PlantUML-ModelingTrust
   a. Modeling Trust/Risk/Trust Anchors, Chains, and Boarders in Archimate, some TextBook patterns
3. PlantUML-ModelingTrust/GnuPG-WoT-Download/README.html
   a. A GnuPG Key Signing Party and Download Signatures (SLSA1) as well as Reproducible Builds SLSA4 modeled in Archimate