SUSE

Zero-Trust-Security in production and delivery

How to implement it via a contract on the blockchain

The Problem

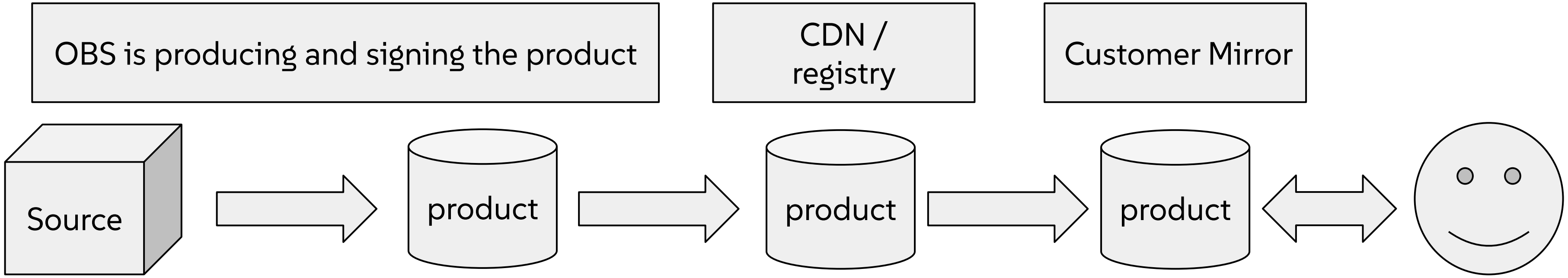
Many many problems actually...

We have many single point of failures leading to a compromised system.

- Every OBS admin can inject not wanted binaries
- Every content provider can block updates. Including the used cloud provider. (same is true for any mirror or container registry)
- Even older content could be provided, since it is still valid signed (known vulnerabilities can be used to attack the consumer afterwards)
- There is no usable way for a customer to verify that he is on a current state when the attack happens in the delivery chain.
- An already reported grave security issue may not reach the customer and he can not easily check on his system the absence.
- Centralized services like sigstore are just moving the problem, but are not solving it.
- A targeted attack to single customer, where only this user gets manipulated content is unlikely to be noticed by anyone.

The Current Setup

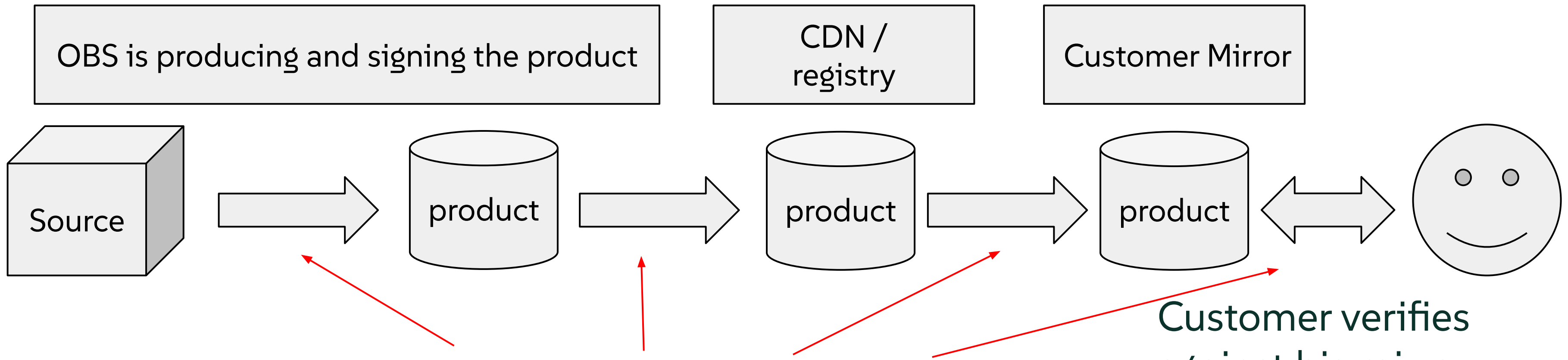
A cascade with many single critical places in a row …



Customer verifies against his mirror only

The Current Setup

A cascade with many single critical places in a row …

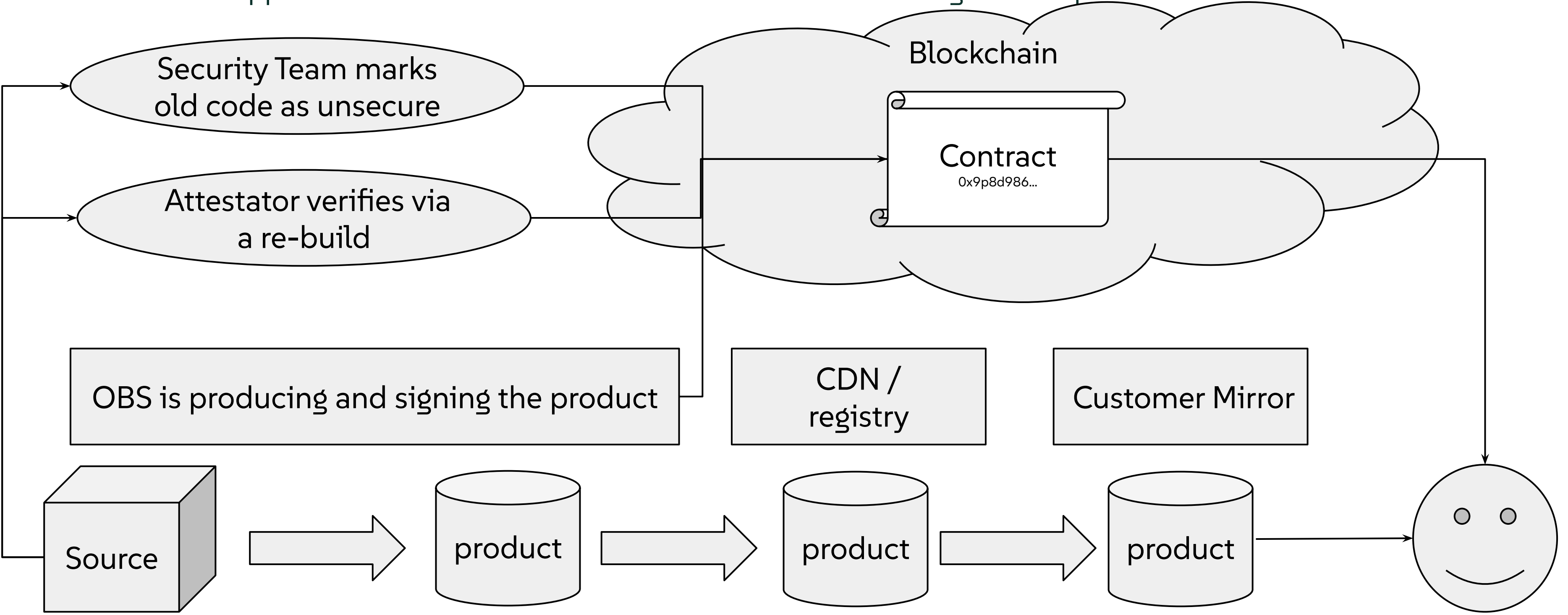


Each step provides multiple risks for manipulation! Attackers might be system administrators, cloud hoster, network providers, SSL CA authorities, …

Customer verifies against his mirror only

The Solution

Verification happens via the blockchain as second channel without a single attack point…



Blockchain
Contract
0x9p8d986…
Security Team marks old code as unsecure
Attestator verifies via a re-build
OBS is producing and signing the product
CDN / registry
Customer Mirror
Source
product
product
product

The First Implementation

EVM contract as (supposed to be) agreed between the parties

- Defines the different roles
- Offers to store product registrations on the blockchain
- Allows to register an executed rebuild with binary identical result
- Allows to register a grave security issue for a build
- Can proof current state of provided distribution
- Contract can only be changed as agreed after deployment

Lean CLI tool to check the local zypper repository cache against the contract:

- No need to buy any crypto currency or to run a blockchain node for the user!

Demo

Using deployed contract on Ethereum Holesky test network

```
# zypper ar https://download.opensuse.org/repositories/home:/adrianSuSE:/suse-distro-blockchain/openSUSE_Factory
# zypper ar https://download.opensuse.org/repositories/home:/adrianSuSE:/suse-distro-blockchain-example/openSUSE_Factory sdb-example
# zypper ref
# zypper in suse-distro-check

# suse-distro-check sdb-example
Reaching out to https://ethereum-holesky-rpc.publicnode.com
Used chain ID: 17000, @block: 2798553, contract: 0x6135d6ec831bD648852Ea10a3f162d353286D4a5
Reading /var/cache/zypp/raw/suse-distro-blockchain-example/repodata/repomd.xml
Selected product:          example-1
Used source SHA-256:       8a645f5782b507202c75ee7fbeaf7bb21d34dd5c2eda4118bb76a31a39226e30
Build Type:                rpm-md
No critical security issues reported
Same rebuild not (yet) attestated
The contract proofed your repository cache as current state :)

(exit code 0, being happy :)
```

Small things missing...

Check Tool:

- Integrate in “zypper ref” via plugin
- Integrate into podman
- Find a generic way for images/appliances

Polish up admin tool:

- Register Builds
- Add attestations
- Set security issue flag

Out of scope for now:

- Implementing distributed proofs and signatures to reach the referenced source code

Resources

- Git repository https://github.com/adrianschroeter/suse-distro-blockchain
- OBS project https://build.opensuse.org/project/show/home:adrianSuSE:suse-distro-blockchain
- Deployed Contract on Ethereum Holesky test network:

  https://holesky.beaconcha.in/address/0x6135d6ec831bd648852ea10a3f162d353286d4a5