When the going gets tough, Get TUF going!

Lily Guo - lily@docker.com
Riyaz Faizullabhoy - riyaz@docker.com / @riyazdf
Motivation

What is TUF?

Using TUF

Hermetic Builds
Where does software come from?
$>curl | sudo bash
$> apt-get install
• authenticity
$> apt-get install

rm -rf /*
• authenticity
• integrity
$> apt-get install really-old-foo
$>#not after 2007
$>apt-get install really-old-foo
• authenticity
• integrity
• freshness
$> $pkg-manager install foo
• authenticity (TLS)
• integrity (TLS)
• freshness
• authenticity (TLS - transport only)
• integrity (TLS - transport only)
• freshness
foo
Replay Attacks?

```bash
$> apt-get install old-insecure-foo
```
Survivable Key Compromise?
Trust Thresholding?
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
• ease of use
Get TUF
(The Update Framework)
• Diplomat: Using Delegations to Protect Community Repositories
• Survivable Key Compromise in Software Update Systems
• A Look in the Mirror: Attacks on Package Managers
• Package Management Security
TUF repository
TUF repository

packages
root
timestamp
snapshot
targets
delegation
Root Metadata

Root:

Timestamp:

Snapshot:

Targets:

Expiry: ...
Root Metadata

Root:
Timestamp:
Snapshot:
Targets:
Expires:
Offline for security

- Backup in bank vault
- Use signing hardware
Targets Metadata

java:

openssl:

...
Targets Metadata

Keys: {
    Alice: A
    Bob: B
}

java: [Alice]
openssl: [Bob]

Expiry: ...
Delegation Metadata

A

java-8-jre: { hashes }
java-7-jre: { hashes }
...
Expiry: ...

B

openssl-1.0.1t: { hashes }
openssl-1.0.2h: { hashes }
...
Expiry: ...
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
Snapshot Metadata

Root: \{ hashes \}
Targets: \{ hashes \}
Alice: \{ hashes \}
Bob: \{ hashes \}
...

Expiry: ...
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
Timestamp Metadata

Snapshot: \{ hashes \}

... Expiry: 24 hours from now...
Timestamp Metadata

Snapshot: \{ hashes \}

...
The diagram illustrates the relationships between different software packages. The primary components are:

- java
  - java-8-jdk
  - java-7-jdk
  - java-8-jre
  - java-7-jre

- jdk

- jre

- apt

- openssl
  - openssl-1.0.1t
  - openssl-1.0.2h

The diagram shows dependencies and cyclic dependencies among these packages.
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
Metadata Lifetime

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
Keeping Freshness

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
Snapshot Expired!

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
Sign a new Timestamp to point the Snapshot

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
Want to publish something?

Timestamp

Snapshot

Targets/Delegations

Root

Lifetime
Sign the hash into a new Targets or Delegation file

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
Sign a new Snapshot that references this Targets file

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
Sign a new Timestamp that references the new Snapshot
Situation normal

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
Oh no, I think my Snapshot key was compromised!
Compromise is “when” not “if”
Root Metadata

Root: 
Timestamp: 
Snapshot: 
Targets:
Before recovery

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
Create and sign the new Snapshot key into Root
Sign a new Snapshot with the new key
Sign new Timestamp to reference new Snapshot

- Timestamp
- Snapshot
- Targets/Delegations
- Root

Lifetime
• authenticity
• integrity
• freshness
• thresholding
• survivable key compromise
• ease of use

GPG

TUF

coming soon!
How can we start using TUF?
Notary is a Docker project that allows anyone to have trust over arbitrary collections of data [https://docker.com](https://docker.com) — Edit

<table>
<thead>
<tr>
<th>Branch</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Commit</th>
<th>Author</th>
<th>Message</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>#900</td>
<td>endophage</td>
<td>Merge pull request #900 from cyli/fix-server-errolldversion</td>
<td>6 days ago</td>
</tr>
<tr>
<td>Godeps</td>
<td></td>
<td>Update protobuf</td>
<td>6 days ago</td>
</tr>
<tr>
<td>buildscripts</td>
<td></td>
<td>Change coverage range for badges, and clean up the cross dir when mak...</td>
<td>11 days ago</td>
</tr>
<tr>
<td>client</td>
<td></td>
<td>added tests for the IsValid setting through VerifySignature</td>
<td>7 hours ago</td>
</tr>
<tr>
<td>cmd</td>
<td></td>
<td>Factor out cert generation to helper</td>
<td>6 days ago</td>
</tr>
<tr>
<td>cryptoservice</td>
<td></td>
<td>spelling</td>
<td>12 days ago</td>
</tr>
<tr>
<td>docs</td>
<td></td>
<td>fix access typos, close 833</td>
<td>6 days ago</td>
</tr>
<tr>
<td>fixtures</td>
<td></td>
<td>Remove the HTTP endpoint for Notary Signer, as it's not used by anyth...</td>
<td>17 days ago</td>
</tr>
<tr>
<td>migrations</td>
<td></td>
<td>Update migration syntax, add integration test for rethink</td>
<td>4 months ago</td>
</tr>
<tr>
<td>misc</td>
<td></td>
<td>Sleep for 1s after push, add comment about certs</td>
<td>4 months ago</td>
</tr>
<tr>
<td>notarymysql/docker-entrypoint-initdb.d</td>
<td></td>
<td>lots of minor improvements to setup.</td>
<td>6 months ago</td>
</tr>
<tr>
<td>passphrase</td>
<td></td>
<td>Refactor passphrase to ask for snapshot and targets passphrases separ...</td>
<td>21 days ago</td>
</tr>
</tbody>
</table>
Demo

- ease of use?
- authenticity
- integrity
- freshness
- thresholding
- survivable key compromise
- ease of use
github.com/docker/notary
IN WE TRUST
alpine
$> export DOCKER_CONTENT_TRUST=1
$> $pkg-manager install openssl
Design Goals:

- root of trust in package manager maintainers
  - with thresholding

- freshness guarantees

- signed index of all packages

- package targets signed by package maintainers
  - with thresholding
package-manager
maintainer(s)
package-manager
maintainer(s)
freshness
signs
index
package-manager
maintainer(s)
freshness
signs index
maintainer keys
Future work: hermetic builds
FROM golang:1.7.1

RUN apt-get update && apt-get install -y \
curl \
clang \
libstdc++-dev \
libsqlite3-dev \
patch \
tar \
xz-utils \
python \
python-pip \
--no-install-recommends \
&& rm -rf /var/lib/apt/lists/*

RUN useradd -ms /bin/bash notary \
    && pip install codecov \
    && go get golang.org/x/tools/cmd/cover
Learn More

• Read the spec:
  • github.com/theupdateframework/tuf/ (docs/tuf-spec.txt)

• Look at Notary:
  • github.com/docker/notary

• Read the Docker Content Trust docs:
  • docs.docker.com/engine/security/trust/content_trust/
Booth D38 @ LinuxCon + ContainerCon

**Thurs Oct 6th**
Orchestrating Linux Containers while Tolerating Failures - Drew Erny
Unikernels: When you Should and When you Shouldn’t - Amir Chaudhry
Berlin Docker Meetup

**Friday Oct 7th**
Tutorial: Comparing Container Orchestration Tools - Neependra Khare
Tutorial: Orchestrate Containers in Production at Scale with Docker Swarm - Jerome Petazzoni
THANK YOU
Root Metadata

Root: 
Timestamp: 
Snapshot: 
Targets: 

Expiry: ...

Appendix: root key rotations
Appendix: root key rotations
Appendix: root key rotations
Appendix: root key rotations
Appendix: DCT pull flow
Appendix: DCT pull flow

1. User initiates the pull process with `docker pull alpine:latest`.
2. Docker CLI requests the manifest/layer using `GET /v2/docker.io/library/alpine/_trust/tuf/*.json`.
3. The Notary Server verifies the data and looks up the hash: `lookup 'alpine' = 'ea0d1389...'`.
4. The user then pulls the specific image `docker pull alpine@ea0d1389...`.
5. Finally, the image is tagged: `docker tag alpine@ea0d1389 alpine:latest`.

The process uses a merkle tree for verification.