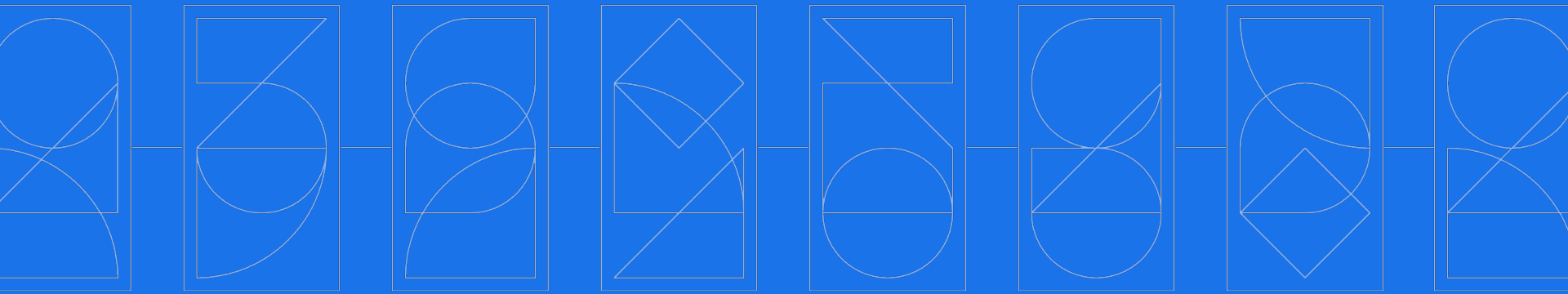
TrustFabric

Clarify your verifiable system design



This presentation contains a **series of questions** for you to answer that will help you clarify your verifiable system design.

The questions will help
you think through
what's important for to log,
who relies on it and
who will verify it.

Each question has an example answer from a **Binary Transparency** scenario to help you.

Example answers for **Certificate Transparency** are in the appendix.

About Binary Transparency Example

PhoneCo are a smartphone manufacturer.

They make software updates and sign them with their private key.

Every phone's updater app checks the update against the known public key before installing it.

Problem

What if someone stole PhoneCo's private signing key?

They could create a malicious update and quietly deliver it to targeted users.

About Binary Transparency Example

Binary Transparency aims to discourage this malicious activity by making it visible.

It introduces a verifiable log with an entry for each software update.

Questions

QUESTION

What information is in each log entry?

Break down what data is stored in an individual log entry. Log entries are permanently recorded so it's worth carefully considering what data is stored.

What information is in each log entry?

Example answer for **Binary Transparency**

The version number and hash of a software update, signed with PhoneCo's private key.

QUESTION

Who or what uses the entries in the log and how?

There is no point in creating a log that no person or thing will ever use or look at.

Who or what uses the entries in the log and how?

Example answer for **Binary Transparency**

Update app

Before installing a new update, the app checks that it exists in the log and only installs it if so.

QUESTION

What types of malicious behaviour are you considering?

There may be lots of different scenarios, that's OK. Write them all out and pick the most important one to start with.

What types of malicious behaviour are you considering?

Example answer for **Binary Transparency**

- A malicious actor stealing the manufacturer's signing key and issuing a malicious software update with a new version number.
- The manufacturer deliberately creating a malicious release e.g. containing malware.

QUESTION

Can you define a "malicious" log entry?

Once you've described a malicious log entry you can start thinking about how they could be verified.

If your log entries contain lots of information, there may be several ways a log entry could be malicious.

Can you define a “malicious” log entry?

Example answer for **Binary Transparency**

A log entry is malicious if:

Hash unrecognised

The hash is for a software update that was made by someone other than the manufacturer (assumed to be malicious).

Duplicated version

The version number is already present in a previous log entry.

Malicious update

The hash refers to a software update that's malicious e.g. contains malware.

QUESTION

Who or what could authoritatively verify if an entry were malicious?

Think about who's ultimately in a position to verify whether an entry is good or malicious.

It may require multiple entities to verify a log entry (based on the different ways it could be malicious). List all the different parties that could verify and how.

Who or what could authoritatively verify if an entry were malicious?

Example answer for **Binary Transparency**

Hash unrecognised

The hash is for a software update that was made by someone other than the manufacturer (assumed to be malicious).

↓

Only the **phone manufacturer** can authoritatively say if a particular hash in a log entry represents a genuine release or was made by someone else.

Duplicated version

The version number is already present in a previous log entry.

↓

Anyone with access to log entries can look for duplicate version numbers. The **phone updater app** verifies an entry is not a duplicate before installing an update.

Malicious update

The hash refers to a software update that's malicious e.g. contains malware.

↓

A **third party security firm** could analyse software updates for malware and other malicious code.

QUESTION

Roughly how many verifiers are required to verify every log entry?

There can be any number of verifiers depending on the claim.

Roughly how many verifiers are required to verify every log entry?

Example answer for **Binary Transparency**

Hash unrecognised

The hash is for a software update that was made by someone other than the manufacturer (assumed to be malicious).

↓

1 verifier - the phone manufacturer.

Duplicated version

The version number is already present in a previous log entry.

↓

1 verifier - anyone that can see the contents of the log.

Malicious update

The hash refers to a software update that's malicious e.g. contains malware.

↓

At least 1 verifier, ideally lots of different security companies.

QUESTION

Are all those authoritative verifiers continually monitoring all entries in the log?

Ideally every authoritative verifier should check every log entry as soon as it's logged, although that's often not possible.

If log entries (or parts of those entries) are never logged, anyone relying on the data in those entries is vulnerable to malicious behaviour.

Are all those authoritative verifiers continually monitoring all entries in the log?

Example answer for **Binary Transparency**

Hash unrecognised

The hash is for a software update that was made by someone other than the manufacturer (assumed to be malicious).

↓

Yes, the security team at the phone manufacturer automatically verify that every log entry hash is present in the build team's release log.

Duplicated version

The version number is already present in a previous log entry.

↓

Yes, in effect, the phone updater app only installs an update after verifying its log entry is not a duplicate.

The security team also continually monitors for duplicate versions.

Malicious update

The hash refers to a software update that's malicious e.g. contains malware.

↓

No. Occasionally a security company audits a software update for malicious code, but it's a manual, time consuming process. Not all versions are inspected.

QUESTION

If all entries are not continually verified, how might they be in the future?

Even if entries aren't fully verified now, it's good to think about whether it would be possible in the future.

If all entries are not continually verified, how might they be in the future?

Example answer for **Binary Transparency**

Hash unrecognised

The hash is for a software update that was made by someone other than the manufacturer (assumed to be malicious).

Duplicated version

The version number is already present in a previous log entry.

Malicious update

The hash refers to a software update that's malicious e.g. contains malware.

↓

Someone (manufacturer? Competitor? Regulator?) could pay one or more security companies to audit every software update.

QUESTION

What happens if the verifiers discover a malicious entry?

What happens if the verifiers discover a malicious entry?

Example answer for **Binary Transparency**

Hash unrecognised

The hash is for a software update that was made by someone other than the manufacturer (assumed to be malicious).

↓

The security team concludes their private key has been compromised in order to sign a malicious update.

Duplicated version

The version number is already present in a previous log entry.

↓

The phone updater app ignores any entries with a duplicate version number.

The security team investigates how the duplicate version log entry occurred.

Malicious update

The hash refers to a software update that's malicious e.g. contains malware.

↓

The third party security company contacts the manufacturer and starts an investigation into how the malicious code got into the update.

QUESTION

What action could be taken as a result of a malicious entry?

If a malicious entry were discovered, there must be some sort of recourse in order to disincentivise malicious behaviour in the first place.

Additionally, sometimes it's also possible to prevent those using the log from relying on the malicious action.

What action could be taken as a result of a malicious entry?

Example answer for **Binary Transparency**

Hash unrecognised

The hash is for a software update that was made by someone other than the manufacturer (assumed to be malicious).

↓

The security team follows an incident response procedure, including rotating their private keys.

They use an out-of-band channel to blacklist the update. The updater app skips the update.

Duplicated version

The version number is already present in a previous log entry.

↓

If the security team's investigation concludes their private key was stolen, they follow an incident response procedure.

Malicious update

The hash refers to a software update that's malicious e.g. contains malware.

↓

The security company publishes an article about the malicious code it found in the software. The manufacturer gets a bad reputation and people stop buying its phones.

QUESTION

Why does the fact that an entry is in the log make it more trustworthy?

Consider why storing data in a log gives more confidence to those that rely on it in your scenario.

Why does the fact that an entry is in the log make it more trustworthy?

Example answer for **Binary Transparency**

Hash unrecognised

The hash is for a software update that was made by someone other than the manufacturer (assumed to be malicious).

↓

As all entries are in the log, the security team can see and verify them. This gives confidence they would spot a bad log entry and prevent installation.

Duplicated version

The version number is already present in a previous log entry.

← ditto

Malicious update

The hash refers to a software update that's malicious e.g. contains malware.

↓

The log means there's a permanent record, so the manufacturer is unlikely to risk making a malicious update and having that discovered.

If an entry is in the log and all entries are actively verified, and there's some recourse in the event of a malicious entry, that gives confidence in the log entry itself.

If either condition is absent, it doesn't give any confidence.

You've finished 🎉

Answering these questions should have helped you clarify the design of your tamper-evident log and verifiable system.

Next try completing a formal analysis using the **Claimant Model** framework.

Appendix

About Certificate Transparency

Certificate Authorities (CAs) issue certificates to domain owners.

CAs should never issue a certificate without the domain owner's consent.

Browsers trust any certificate that's issued by a trusted CA.

This is fundamental to the security of HTTPS.

Problem

What if someone stole a CA's private signing key?

What is a CA made a mistake, or acted maliciously?

An attacker could create a fake certificate and quietly deliver it to targeted users, allowing them to perform a man-in-the-middle attack.

About Certificate Transparency

Certificate Transparency aims to discourage this malicious activity by making it visible.

It introduces a verifiable log with an entry for every certificate issued by a CA.

What information is in each log entry?

Example answer for **Certificate Transparency**

The newly issued certificate, signed by the Certificate Authority.

Who or what uses the entries in the log and how?

Example answer for **Certificate Transparency**

Browsers

When a browser receives a certificate, it checks that it exists* in one of a list of logs that it already knows about.

*Technically it checks a promise from the log that it will be entered, but it amounts to the same thing.

What types of malicious behaviour are you considering?

Example answer for **Certificate Transparency**

- A Certificate Authority deliberately issuing a certificate for a domain without the domain owner's consent.
- A malicious actor stealing a CA's signing key and issuing a bad certificate in the name of the CA.

Can you define a "malicious" log entry?

Example answer for **Certificate Transparency**

A log entry is malicious if the certificate was issued without the consent of the domain owner.

Who or what could authoritatively verify whether an entry were malicious?

Example answer for **Certificate Transparency**

A log entry is malicious if the certificate was issued without the consent of the domain owner.

Only the domain owner can authoritatively confirm whether they asked for a certificate to be issued for their domain. They compare it against the certificates they generate themselves.

Roughly how many verifiers are required to verify every log entry?

Example answer for **Certificate Transparency**

A log entry is malicious if the certificate was issued without the consent of the domain owner.

In a perfect world, millions (every domain owner). In reality, fewer if domain owners delegated to another party e.g. their hosting provider.

Are all those authoritative verifiers continually monitoring all entries in the log?

Example answer for **Certificate Transparency**

A log entry is malicious if the certificate was issued without the consent of the domain owner.

No, in general. Most domain owners don't monitor the logs to check for certificates that they didn't request. Some people use services such as CertSpotter, Google Domains.

If all entries are not continually verified, how might they be in the future?

Example answer for **Certificate Transparency**

A log entry is malicious if the certificate was issued without the consent of the domain owner.

- Companies could start running their own infrastructure to verify CT logs (Facebook and a few others currently do)
- Domain owners could start using services like crt.sh, CertSpotter.
- For less technical domain owners, hosting companies that offer domains and certificates could start checking CT logs on behalf of their clients.

What happens if the verifiers discover a malicious entry?

Example answer for **Certificate Transparency**

A log entry is malicious if the certificate was issued without the consent of the domain owner.

- The domain owner tells the browser security teams that they did not authorize a particular certificate to be issued.
- The domain owner might tell a journalist about the CA misbehaving.

What action could be taken as a result of a malicious entry?

Example answer for **Certificate Transparency**

A log entry is malicious if the certificate was issued without the consent of the domain owner.

The browser security team decides on a case by case basis. They could take actions against the CA:

- In the extreme, remove them from the browser's certificate store.
- Require the CA to undergo a security audit.
- Require the CA to rotate its signing keys and reissue all certificates.

This doesn't help the user who already trusted the malicious certificate, but does make malicious certificates overall less likely.

Further, a journalist could publish a damaging article about the CA which would harm their reputation.

Why does the fact that an entry is in the log make it more trustworthy?

Example answer for **Certificate Transparency**

A log entry is malicious if the certificate was issued without the consent of the domain owner.

A browser has more confidence that a certificate is not malicious since the certificate is available for all to see. The rationale is that a CA is less likely to issue a malicious certificate since they're more likely to get caught (providing someone noticed the bad log entry.)