System Transparency Logging

No Author Given

No Institute Given

1 Overview

Suppose that you downloaded an executable binary. You are ready to install it, but realize that there might be a catch. How do you know it is the same executable binary that everyone else downloaded, and not, say, a modified one which includes a backdoor? The short answer is that you cannot easily know that.[1] In this talk, we present a partial solution that helps *detect* such attacks.

The basic idea is to use a transparency log. It is like Certificate Transparency, expect that we are logging cryptographically signed checksums as opposed to X.509 certificates. We focus on checksums that represent system artifacts. A system artifact could be an executable binary, a Debian package, or an Ansible playbook (to mention a few examples). Our goals can be summarized as follows:

1. Ensure that everyone observes the same signed checksums.
2. Facilitate detection of compromised signing keys.

2 Threat model

We consider an attacker that gained control of a software provider's secret signing keys and release infrastructure. The attacker also runs the log, and up to a threshold of cosigning witnesses. Witness cosigning is part of our gossip-audit model. Without a gossip-audit model the log must be a trusted third-party.

The attacker's goal is to make a client accept a malicious artifact without anyone noticing. Assuming that the client enforces transparency logging, the attacker's best bet is no longer to craft a malicious artifact that is only served to a handful of clients. This follows from the fact that there must be a public trace of each artifact before a client accepts it as valid, making it *discoverable*.

3 Current status

Our transparency log design is implemented as a Trillian personality. Trillian is essentially a general-purpose backend with a database and an append-only Merkle tree. The personality part is an application-specific front-end. Further details can be found in the System Transparency Front-End (STFE) repository.

[1] A reproducible build is helpful, but it still begs the question whether you see the same source code as everyone else. Our work coexists well with reproducible builds.