Why Provable Transparency is Useful Against Surveillance

Ben Laurie, Google (benl@google,com), 11 Jan 2014.

Background

Certificate Transparency (RFC 6962) is an example of a general mechanism which permits a public, verifiable, append-only log of stuff.

In the case of CT, the stuff is X.509v3 certificates, but it works equally well for anything that can be expressed as bits.

Verifiable append-only logs can also be used to create verifiable maps[1].

Verifiable logs and maps can be used as tools to prevent, or at least make substantially more difficult, certain types of surveillance.

Examples

HTTPS Interception

Certificate Transparency guards against the issuance of HTTPS certificates to parties other than their intended recipients without this becoming noticable. This makes it very hard to intercept traffic sent over HTTPS through either middleboxes or DNS hijacking - or any other mechanism which leaves the URL to the intended website intact.

End-to-end message encryption

Email, instant messaging and the like have mechanisms permitting the encryption of invidivual messages using the public key of the recipient.

The problem is, there are no good, widely usable, methods to map an email address or IM identity to a correct public key.

Verifiable maps offer the possibility of maintaining such mappings in a way that does not require a trusted third party.

Note that this is not a complete solution - unlike HTTPS certificates, where there is already an ecosystem existing around verifying who should have certificates and revoking them when inappropriately issued, there is no such thing in place for, e.g. PGP keys. But, at least, if you believe that the intended recipient of your message is checking the map, you can be reasonably sure that if there are no disputed entries in the map, then you have the correct key.

---

[1] The log contains all changes to the map (e.g. all additions and deletions). Replaying the changes gives the current state of the map. Optimisations are available for clients prepared to rely on public audit to retrieve particular entries rather than having to retrieve the entire log.

It is less clear what should be done in the case of disputes - perhaps find a more direct way to contact the recipient? Certainly an area for further thought and research.

Binary Transparency

One criticism of many software solutions to privacy and anonymity is that the vendor can replace the binary for any particular user with a customised version containing malware. This is particularly true for Javascript based solution, due to their typically dynamic nature, but applies to any binary.

If platforms declined to install binaries (including executable web content) unless the binary was demonstrably in a public log, this would make such attacks much more difficult: the attacker would still be able to serve custom binaries, but would have to make them publicly visible, vastly increasing the probability that he would be found out.

DNSSEC

There is a good deal of enthusiasm for DANE, moving away from the CA model for TLS certificates. But DNSSEC is not a magic bullet - high value targets typically have domains in every TLD, and this leaves them at the mercy of the whole ecosystem of registries and registrars (and the APTs that might control them). Transparency can be applied to DNSSEC keys similarly to TLS keys. In this case, anti-spam seems possible by tying the logs to the name hierarchy - each TLD designates a log, for itself and possibly for its children, but if children misbehave, the parent can state that the child's log is elsewhere.

Conclusion

Whilst transparency (i.e. verifiable logs and maps) does not directly prevent attacks it does convert many attacks that can currenty be carried out covertly to overt attacks, increasing the probability that the attack is discovered, the attacker held accountable and the attack counteracted.