Certificate reputation

Dorottya Papp
Motivation

- Verification on a digital certificate does not reveal important factors
  - Is it a fake certificate? (Hash collision)
  - Was it mistakenly issued? (Comodo scandal)
  - Was it maliciously issued? (Rogue CA)
- How do CAs operate?
- Is the current CA system satisfactory?
What does the big picture look like?
SSL Observatory, ICSI Certificate Notary

ANALYSIS AND DATA PROVIDING
EFF SSL Observatory

- **Goal:**
  - Document CA behavior
  - Search vulnerabilities connected to certificates

- Collected IPv4 certificates with TLS Handshake

- MySQL database for certificates
  - Also made public (for a time)
SSL Observatory

- Windows or Firefox → 1 482 CA certificates
  - = 650+ organizations
- Certificate for 192.168.1.2 …
- Certificate for localhost
- Certificate that is not a CA but can sign
- Result: colored map of CAs
ICSI Certificate Notary

- Passive collection from live upstream data
ICSI Certificate Notary

- Usage: DNS queries
- Tree of Trust
REPLACING THE CA SYSTEM
EFF Sovereign Keys

• Proposal to fix structural inconsistencies in establishing encrypted connections

• Proving control of a domain:
  ◦ control a CA-signed certificate or
  ◦ has to use a DNSSEC-signed key

• Creation of Sovereign Keys
  ◦ writing to a semi-centralized, verifiably append-only data structure → original claim cannot be altered
  ◦ Master copies are kept on timeline servers
  ◦ Additional copies on mirrors for scalability
EFF Sovereign Keys

- Shortcomings
  - DoS against mirrors: store TB of junk indefinitely (a.example.com, b.example.com, …)
  - Attackers may add malicious mirrors faster, then users could notice them being bad
  - Rogue CA problem: if timeline servers are willing to pause additions to their timelines for some time (or run their clocks slow), and collaborate with a Certificate Authority or a party in the DNSSEC hierarchy, they may be able to pretend to have registered new sovereign keys before the actual registrants
EFF Sovereign Keys

• Short coming
  ◦ Monotonicity at all timeline servers → synchronized clocks all the time (what about leap second handling?)
  ◦ Time measurement of a timeline server can not be verified or contested by registrants
Perspectives

- **Goal:**
  - Clients should be able to choose who they trust
  - Improve Trust-on-first-use (Tofu) authentication

- **Infrastructure:**
  - Public notaries: monitor and build public history of SSL certificates
  - Notary Authorities: determine legitimate notary servers and publish them
Perspectives

Client connects to service

1. Client

2. Service responds with public key

3. If offered key is untrusted, client contacts notary

4. Notary returns records of key history to the client

5. Client interprets relevant key data and accepts/rejects key

Network Service

Notary
Perspectives

- Shortcomings:
  - Leaking browsing history
  - Notary lag: certificates change between probings → invalid result
Convergence

- Improves the design of Perspectives
- Additional goal – trust agility:
  - Trust decision can be revised at any time
- Notary lag
  - Users supply the certificate, notary contacts the website
- Privacy problem
  - Local caching → notary is contacted only when the certificate is unknown
  - Notary bounce: trusted notary acts as a proxy
Convergence

- Trust threshold on user side
  - Majority/minority of notaries agree?

- Shortcomings
  - Citibank problem: many certificates, each request is answered with a different certificate
  - Captive portals → implementation upon the DNS level
Convergence

- REST API for notaries → extensive design
Another approach: monitor certificates

GOOGLE CERTIFICATE
TRANSPARENCY
Goals

- Open framework for monitoring and auditing SSL certificates in nearly real-time
- Detect
  - Mistakenly issued certificates
  - Maliciously acquired certificates
- Identify rogue CAs who issue certificates maliciously
Architecture

Current TLS/SSL System

TLS/SSL System with Certificate Transparency (X.509v3 Extension)

- Certificate Authority
- Log Server
- example.com
- Client (browser)

1. CA submission (Precertificate)
2. Log response (SCT)
3. Cert issuance (SSL cert w/SCT)

- Existing TLS/SSL system
- Supplemental CT components
- One-time operations
- Synchronous operations
- Order of operation
Components

- **Certificate logs**
  - Maintain cryptographically assured, publicly auditable and append-only records
  - Records contain certificate chains
  - When a chain is submitted, a signed timestamp is returned → evidence

- **Monitors**
  - Publicly run servers
  - Periodically fetch data from all log servers
  - Watch for suspicious certificates
Components

- Auditors
  - Lightweight software components
  - Verify log behavior and cryptographic consistency
  - Verification of a particular certificate
  - Take partial information about a log and verify this information with other partial information they have
- Implementation
  - Integral component of the TLS client
  - Standalone service
  - Secondary function of a monitor
Workflow – Integrated into TLS

Monitors watch logs for suspicious certs and verify that all logged certs are visible.

Certificate owners query monitors to verify that nobody has logged illegitimate certs for their domain.

Auditors verify that logs are behaving properly; they can also verify that a particular cert has been logged.

Monitors and auditors exchange information about logs to help detect forked or branched logs.
QUESTIONS?