Authentication Architectures

Steven M. Bellovin
smb@research.att.com
http://www.research.att.com/~smb

Why Authentication?

- Relatively straight-forward concepts
- Often amenable to centralized architectural solution
- A lot more complexity under the hood

A Long Time Ago...

- Many years ago, I was asked to help devise security mechanisms for some OSS.
- The primary risk being discussed was unauthorized access via craft terminals.

Approach 1: Passwords

- Passwords can be -- and are -- easily shared.
- Passwords can be -- and are -- easily guessed.
- Passwords can be -- and are -- written down on yellow stickies.
- Passwords were the approach the system used -- and they didn't work.

Approach 2: Smart Cards

- Smart cards are expensive.
- Readers are even more expensive.
- Readers are too heavy, too bulky, and draw too much power for hand-held terminals.
- What if people forget their cards one day?

Approach 3: Hand-held Tokens

- Tokens are expensive.
- Tokens are inconvenient.
- The craft personnel won't like them.
- What if people forget their tokens one day?

Approach 4: Proxy Logins

Let users log in to their hand-held terminals; let the terminals act as proxies when talking to other systems.

- There isn't enough extra memory in the terminals.
- How do we administer the logins on these terminals?
- The rest of the system architecture isn't built for that sort of thing.

Approach 5

- Use passwords -- still.
- Rely on procedural safeguards to prevent the old problems.
- Will that work? Probably not...

Lessons Learned

- Security isn't free; in fact, it can be expensive.
- Security isn't always convenient.
- The embedded base made matters worse.
- I had no security pixie dust available.

But -- you have to start some time. And what is the cost of insecurity?

From the Baltimore Sun,13 June 1989

Callers trying to dial a probation office in Delray Beach, Fla, yesterday heard sex talk from a panting woman named Tina instead.

Southern Bell officials said that a computer hacker reprogrammed their equipment over the weekend, routing overflow calls intended for the local probation office to a New York-based phone sex line.

Fast Forward: Security for the NNS

- Tokens are now accepted security measures.
- Back-up maintenance architecture designed for dumb, dial-up terminals.
- Nary a Web browser in sight, originally.
  - Of course, that changed rapidly.

Web Browsers and OSSes

- Browsers are now *the* universal graphical interface.
- Any modern system design needs to accommodate such an interface.
- But what does that do to security?

Browsers Versus Tokens

- Web browsers are stateless; each interaction must be authenticated separately.
- Basic web authentication sends the same string for each request.
- But tokens generally give a *different* answer each time, to prevent replays...

How Should Browsers Do Authentication?

- Users logs in once to authentication server; server sends out "cookie".
- Or user has "client-side certificate".
- Both mechanisms can be compatible with tokens.
- But both *can* be implemented with passwords, with most of their problems.

Authentication Cookies

- Often replayable.
- Sometimes stored on disk; vulnerable to theft due to browser bugs.
- Simple; requires little infrastructure.
- But -- some designs provide *too much* authority. And some designs don't scale well for large, distributed applications.

Certificates

- Certificates can cryptographically associate a public key with an identity.
- Must be installed on each machine used by each individual.
- Fundamentally superior -- secret never shared with any other component.
- Can be implemented in tokens.

But -- do they require too much infrastructure?

The Myth of the PKI

Do certificates require a corporate-wide PKI?

- Certificates only need to be authorized by their own domain of discourse.
- Corporate-wide secure email may need a corporate solution.
- Craft logins to an OSS only require certificates for that OSS; no corporate-scale work is required (or desired).

“I Left my Token Home”

- Local management can cope with this *better* than a central PKI can.
- New token (or certificate) can be issued quickly.
  - But must update authorization information as well.
- Revocation is an issue for any public-key technology; quick access to authorization tables can help.

Upcoming Issues

- Single sign-on
- Role-based authentication
- Cross-certification
- Trustworthy platforms

Single Sign-on

- Vast improvement in convenience.
- Potentially helps security -- no need to constantly supply passwords.
- But can hurt security -- if A trusts B and B trusts C, does A trust C? Should it?
  - Don't extend trust transitively; always go back to user's sign-on station!
  - Easier to control with certificates.

Role-Based Authentication

- How can one user do another's job?
  - Sharing passwords (or certificates or tokens) is a bad idea.
- Could use short-lived certificates.
- Authorize multiple certificates for shared jobs -- same privileges, but different identity.
  - Implies that each user has many certificates.
    - Another reason to avoid a central, identity-based PKI.

Cross-Certification

- How does a user of one OSS perform actions on another?
  - A corporate PKI doesn't solve this; *authorization* counts.
- OSS's certify each other; each OSS vouches for the identity and authorization of its own users.

Trustworthy Platforms

- Can we trust Windows?
  - Tokens authenticate the user, not the user's actions.
- Given that we can't trust Windows, how do we build secure systems?
  - See *Trust in Cyberspace* (http://www.nap.edu/readingroom/records/0309065585.html) for more discussion of this.
- Note: monocultures can hurt.

The Role of Architecture

- Defines a goal
- Shows a direction for migration of existing platforms and design of new ones
- Justifies short-term expense to achieve long-term improvement
- Promotes interoperability

No Panaceas

- Applications, requirements, and environments do differ; one size does not fit all.
- Technology changes; yesterday's answers won't fit tomorrow's problems.
- Architecture cannot be static.