Application Security: Threats and Architecture

Steven M. Bellovin
smb@cs.columbia.edu
http://www.cs.columbia.edu/~smb

We're from the Security Area, and We're Here to Help You

- We annoy a lot of people
- We keep demanding more security mechanisms
- We keep demanding more security analysis
- We keep changing what we want
- Is there a reason for this, or is the Security Area a home for professional nuisances?

The World Has Changed

Old	*New*
Teenage joy-hackers	Hacking for profit
Password-guessing	Distributed password-guessing
Password “sniffing”	Programmable bots with “sniffers”
Exploit bugs	Protocol-level attacks
Simple scanner	Tailored worms and viruses

Why has this happened? “Follow the money”.

The requirements have changed

because the threats have changed.

What are Today's Problems?

- Eavesdropping
- Monkey-in-the-middle
- ARP-spoofing
- "Evil twin" access points
- Routing attacks

All of these are seen in the wild. (See Christian Huitema's APPS Area slides (`http://www.huitema.net/talks/ietf63-security.ppt`) for an excellent precis of the situation.)

Patterns of Thought

- Serial number 1 of any new device is delivered to your enemy.
- You hand your packets to your enemy for delivery.
- Your enemy is just as smart as you are. If we haven't seen a given class of attack yet, it's because it hasn't been necessary; simpler attacks have worked well enough. (Besides, how do you know if you'll actually notice it?)

Things that Don’t Work Well

- Plaintext passwords (we outlawed them a long time ago)
- Plaintext challenge/response based on passwords
- Crypto without bilateral authentication: to whom are you talking?

Is This the Party to Whom I am Speaking?

- Who is at the other end of a TCP connection?
- Who is at the other end of a TLS-over-TCP connection?
- Is it the party you meant? Think about `paypal.com`, `whitehouse.com`, or `nasa.com`

Who is the Right Party?

- With two-party protocols, you often have some idea of the other party's identity and credentials
- Problems can arise if you don't know the other side — that's why signed email won't have much effect on spam — or if you're relying on untrustworthy third parties (some commercial CAs)
- Multi-party protocols make this much worse

Multi-Party Protocols

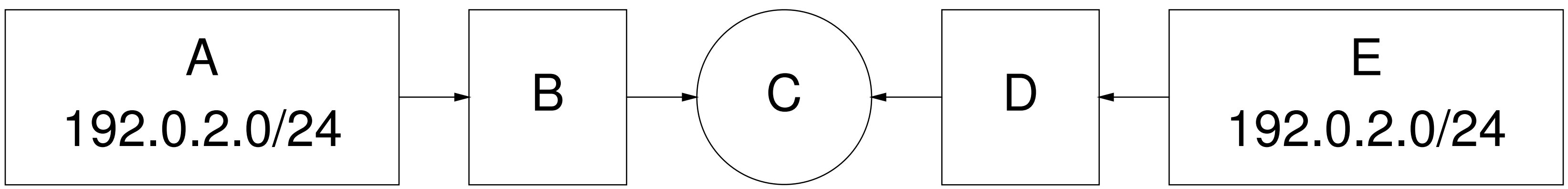
- More and more of our protocols are multi-party: BGP, SIP, AAA, p2p, etc.
- The client may not have a direct relationship with the ultimate server, and vice-versa
- How can either party verify the other's credentials?
- More seriously, how can either party verify the other's *authority*?
- Note: such connectivity often instantiates *business agreements*, the terms of which are often not easily reducible to protocol syntax and semantics

The Routing Problem

Autonomous system A advertises 192.0.2.0/24 to BGP peer B.

B tells C that the path to 192.0.2.0/24 is {B,A}.

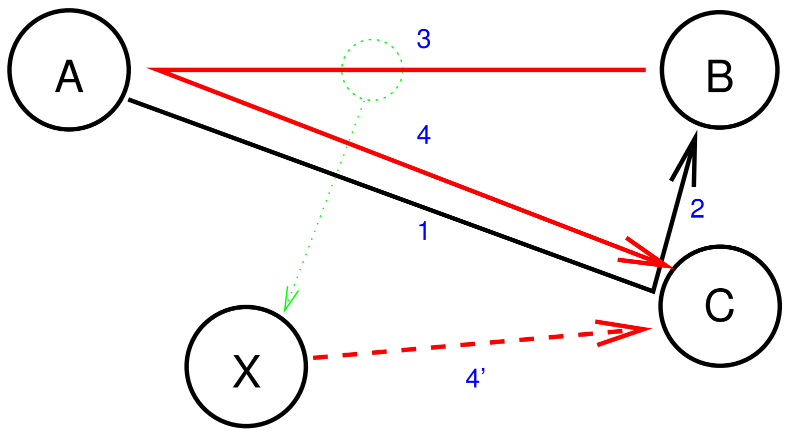
Similarly, E advertises the same prefix to D, which tells C that the path to 192.0.2.0/24 is {D,E}.



Which should C believe? Either? Both? Neither?

C has contracts with B and D, which specifies what prefixes they may originate. C has no contract with — or knowledge of — A or E.

SIP Call Transfer



1. A tries to call C
2. The call is redirected to B
3. B agrees to transfer the call to C
4. A contacts C

Can X steal those credentials and call C? How does C know that messages 4 or 4′ are *authorized*?

Transitive Trust

- Sometimes trust is transitive

☞ In that case, cryptographic tokens can be used to convey authority

- Sometimes, trust is done by reference to external authority: should RIRs give out certificates for IP address blocks?
- If this isn't possible — consider a SIP proxy chain

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$$

Can A trust D to forward the call setup to the real E? Does A have any idea of D's existence, role, or trustworthiness? Does A even know that D is in the path?

Cryptography Depends on Authorization

- In the first SIP example, message 3 cannot be reliably encrypted unless either A or C has authentication credentials for the other.
- Are you encrypting your message to the *right* party?
- An encrypted channel to a bad guy only provides protection from intrusion detection systems...
- Trusted — and trustable — authorities are essential for protocol security.
- You can be your own authority if you wish to hand out credentials to everyone you talk to.
- But can you trust yourself?

Secure Application Protocol Design

- Identify the different parties
- Identify the trust relationships between them
- Who has to trust whom?
- How is identity established? How is authorization established?
- Bilateral communication can be handled by mutual agreement and (offline) credential exchange
- Multi-party communication is *much* more difficult
- You can't build a secure protocol without this analysis

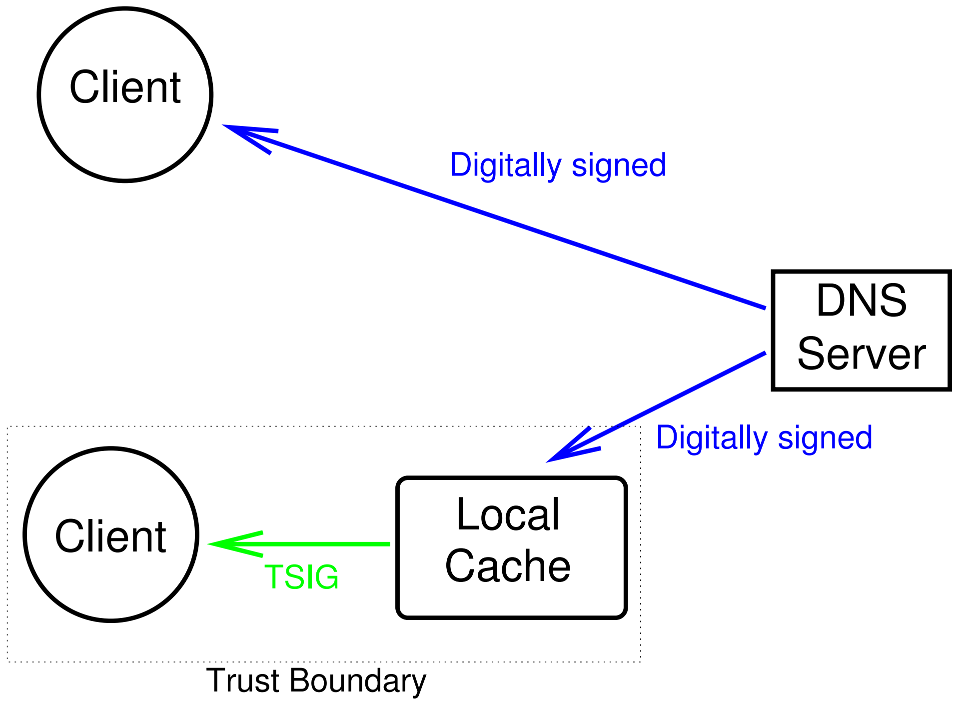
Security from the Beginning

- It's easy to bolt on crypto on a single path
- It's hard to add it later on a multi-hop path
- It's *very* hard to change the trust model later. (Example: "redirects" are easier to analyze than proxies.)
- Moral: do the analysis *very* early on, and get help early

Selecting Cryptographic Primitives

- Do you need confidentiality+authentication or just authentication? (Note: confidentiality without authentication is generally dangerous)
- For two-party communication, symmetric cryptography is often sufficient (but try to avoid passwords)
- When multiple parties need to see a single message, you almost always need public key cryptography
- Often, hybrid schemes can be used
- If standard IETF cryptographic protocols cannot be used, contact the Security Area.
- Even the Security Area isn't competent to design cryptographic primitives such as hash functions and encryption algorithms

Hybrid DNSsec Paths



DNSsec uses digital signatures because it is multi-party. But a trusted local cache can do the expensive verification, and use TSIG to reliably tell a local party the results.

Properties of Cryptographic Primitives

- Encryption is much more expensive than hashing
- Public key crypto is much more expensive than symmetric crypto
- Public key often scales better to large environments — the (highly secure) credential issuer need not be online at all times, and old client credentials are not endangered if that machine is compromised
- Revoking public key credentials is hard work
- Symmetric techniques can work well if all parties are online simultaneously
- The choice is often difficult, and frequently depends on estimates of likely scale and deployment patterns

Final Thoughts

- The enemy is getting a lot better
- We *must* use cryptography to secure our protocols (though that won't protect us against buggy code)
- Proper cryptographic desgin depends on four things:
  - Cryptographic primitives (RSA, AES, SHA-1, etc)
  - Cryptographic protocols (Security Area)
  - Threat model (Security Area and protocol designers)
  - Trust patterns
- Only the protocol designers understand the trust model
- Everyone has to work together on the threat model — but it's constantly getting worse