Application Security: Threats and Architecture

Steven M. Bellovin

`smb@cs.columbia.edu`

`http://www.cs.columbia.edu/~smb`

Dept. of Computer Science, Columbia University

Types of Problems

- Simple bugs
- User actions
- System architecture

Types of Problems

Simple Bugs Easy to patch, but more keep showing up

User Actions Little good work done on human interface

System Architecture Very difficult to patch later; systems need to be designed properly from the start

The Old World

Old

Teenage joy-hackers

Password-guessing

Password “sniffing”

Exploit bugs

Simple scanner

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
- Fake certificates
- Host attacks

All of these are seen in the wild. Most of these are done for money.

Common Attitudes

- No one is interested in me
- They don't know my software
- Nothing can go wrong... go wrong... go wrong...

Better Patterns of Thought

- I'm target Number 1
- Serial number 1 of any new device is delivered to my enemy.
- I hand my packets to my enemy for delivery.
- My enemy is just as smart as I am. 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
- Plaintext challenge/response based on passwords
- Crypto without proper authentication: to whom are you talking?

Plaintext Passwords

- Password sniffing has been going on since 1993.
- Then, it was on the Internet backbone; now, it's localized
- The easiest defense is crypto

Password-Based Challenge/Response

- Intercept such a packet
- Guess at passwords; see if the response matches
- Such code has existed for quite a while
- Strong password authentication protocols exist, but they're not used much

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 `paypa1.com`, `whitehouse.com`, or `nasa.com`

One Word

Authorization

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

One of the Causes of Phishing!

- Users can’t tell the genuine sites from the fakes
- Virtually no one knows what a certificate is
- Of those who do, virtually no one knows what certficate *should* be sent
- Supply a certifcate with the initial deposit receipt? This is an *authorization* certificate, not an identity certificate

Human Factors

- Our systems are not designed for ordinary users
- What user knowledge do we rely on for security?
- How many users have such knowledge?
- What can we do about the lack thereof?

Example: Certficates

- The entire SSL system was based on an academic model of cryptography
- Academically, it's very sound — but it relies on a *trust anchor*
- Again — users do not know what such a thing is
- We haven't tried to compensate
- The result has been phishing

We Can’t Hand Out Certificates

- There’s no easy way to give the user a bank’s certificate
- We can yell at Microsoft, or we can design around the problem.
  - SecurID card or other token
  - One-time password lists
  - Client-side certificates
  - Something only the real user will recognize

We've Taught Bad Habits

- How many companies use both `www.mymegacorp.com` and `www.mymegacorp-Online.com` or the like?
- Too many — and by doing so we've accustomed users to such things
- `online.mymegacorp.com` or `www.mymegacorp.com/online` are both better

Fake Certificates

- Newest attack: a fake, self-signed certificate
- “Instructive” text: “When you see the message about the certificate authority, click ‘yes’. By relying on our own certificate authority, we’ve improved our security”
- Users are accustomed to clicking “yes”

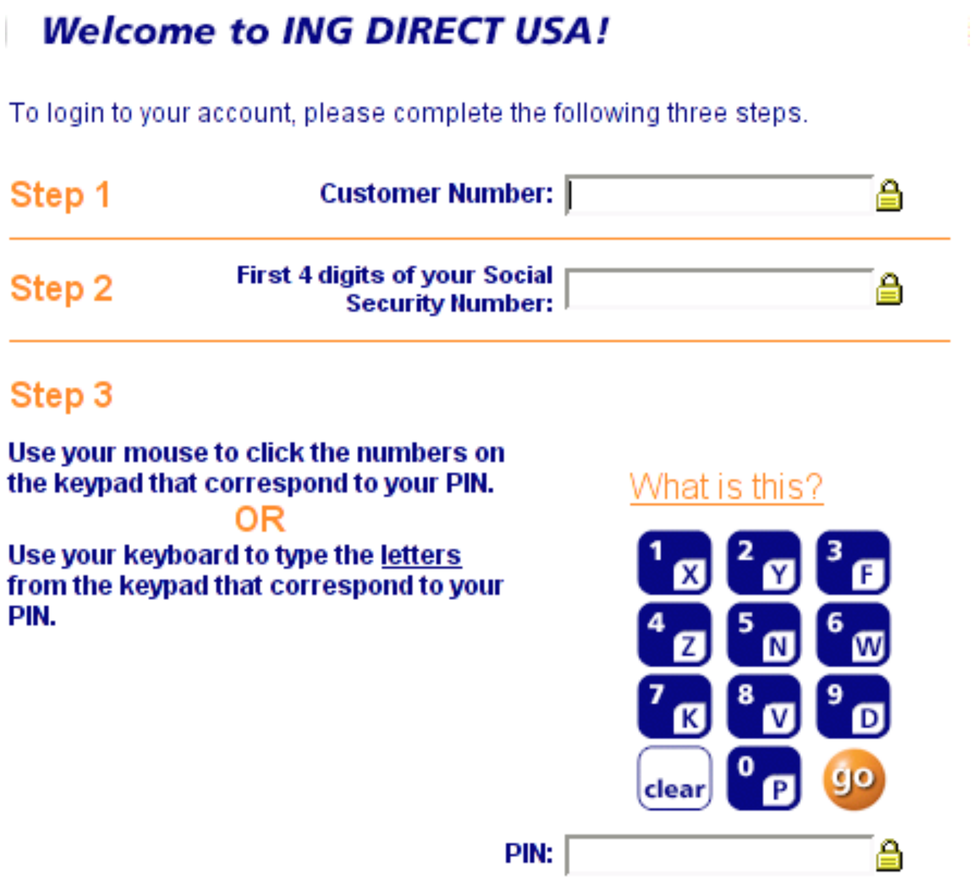
Corrupted Hosts

- Most security problems are due to buggy host software
- Attackers have been going after end-hosts
- Keystroke loggers, wait-for-login attacks, and more
- Hard to do business with a compromised endpoint...

Not Much to Do...

- Can try to avoid keystrokes or reusable passwords
- Bad guys catching up rapidly — screen+mouse capture for ING Direct login screen
- No good defense against wait-for-login attacks

Mouse-Click PINs



Outboard Cryptographic Authentication

- The user has no access to authetication secrets
- Trojans have no access to authetication secrets
- Sounds good, right?
- Unless the system is carefully designed, the malware can talk to the outboard device, too
- Solution: only let it authenticate after user physical action
- (But which program gets there first?)

ARP-Spoofing and Evil Twins

- ARP-spoofing redirects traffic on Ethernet and WiFi LANs
- Evil twins are fake WiFi access points — attacks mostly happen at hotspots
- Both serve the same purpose: all of your traffic goes directly to the attacker
- Again: how do you know the *authorized* party?

Multi-Party Protocols

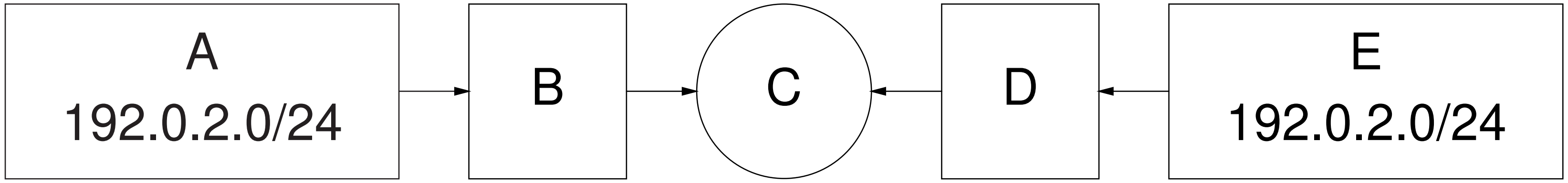
- More and more protocols are multi-party: BGP, SIP, AAA, p2p, etc.
- The client may not have a direct relationship with the ultimate server, and vice-versa
- How can either party verify the other's credentials?
- More seriously, how can either party verify the other's *authority*?
- Note: such connectivity 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}.

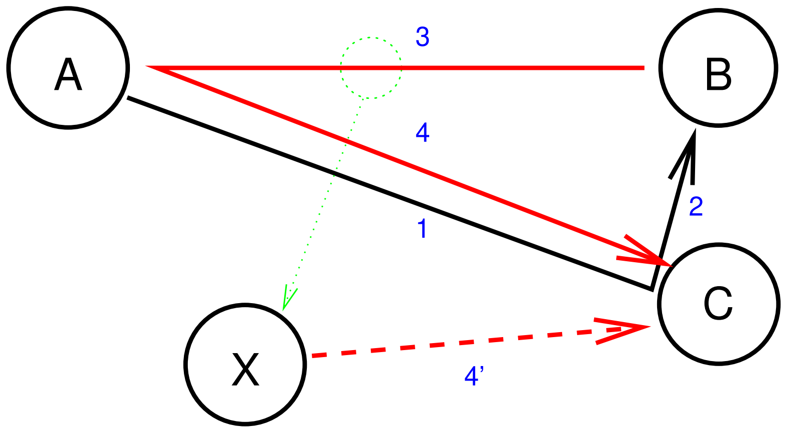
Similarly, E advertises the same prefix to D, which tells C that the path to 192.0.2.0/24 is {D,E}.



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?

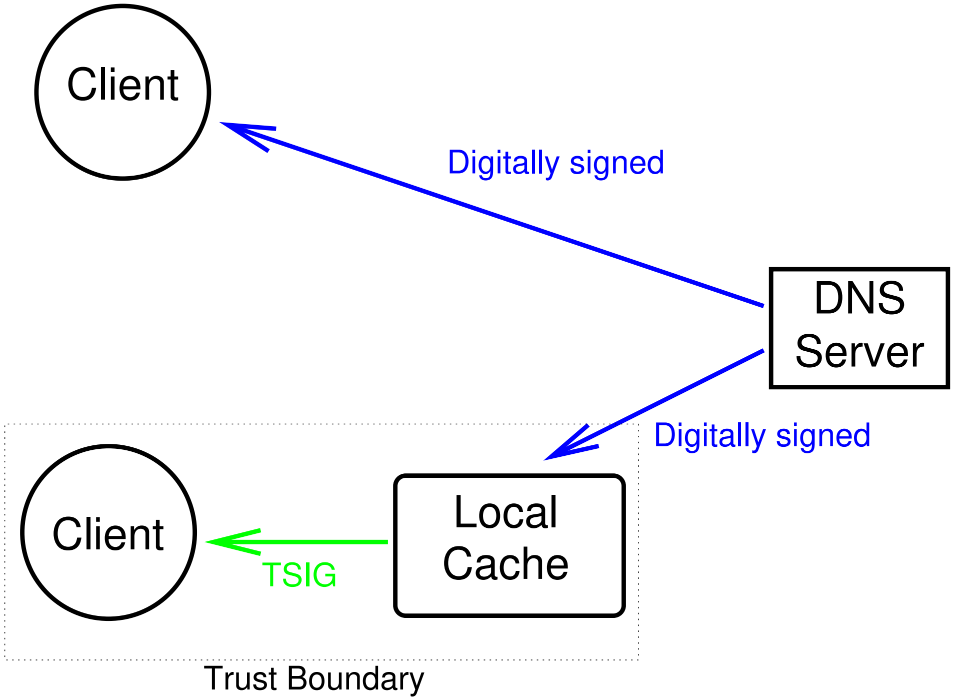
Security from the Beginning

- 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 cryptographic protocols cannot be used, contact a cryptographer
- Very few people are 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

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

The Parties

- Do they know each other?
- How sophisticated are they?
- Can they be educated?
- If not, how can you work around the problem?

Trust Relationships

- Must make technical mechanisms mirror business relationships
- For example, if A vouches for B, let B use a certificate signed by A.
- Make sure that transitivity, if needed, is protected by business relationships

Auditing Systems

- Follow the money
- Follow the authorization chain
- Follow the user behavior

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 design depends on four things:
  - Cryptographic primitives (RSA, AES, SHA-1, etc)
  - Cryptographic protocols
  - Threat model
  - 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