Anti-pattern: “oracle”

- Any way a legitimate protocol service can give a capability to an adversary
- Can exist whenever a party decrypts, signs, etc.
- “Padding oracle” was an instance of this at the implementation level

Packet sniffing

- Watch other people’s traffic as it goes by on network
- Easiest on:
  - Old-style broadcast (thin, “hub”) Ethernet
  - Wireless
- Or if you own the router

TCP spoofing

- Forging source address only lets you talk, not listen
- Old attack: wait until connection established, then DoS one participant and send packets in their place
- Frustrated by making TCP initial sequence numbers unpredictable
  - Fancler attacks modern attacks are “off-path”

ARP spoofing

- Impersonate other hosts on local network level
- Typical ARP implementations stateless, don’t mind changes
- Now you get victim’s traffic, can read, modify, resend
**rlogin and reverse DNS**
- rlogin uses reverse DNS to see if originating host is on whitelist
- How can you attack this mechanism with an honest source IP address?

**Outline**
- Cryptographic protocols, cont'd
- Some classic network attacks
- Key distribution and PKI
- Announcements intermission
- SSH
- SSL/TLS

**Public key authenticity**
- Public keys don't need to be secret, but they must be right
- Wrong key → can't stop middleperson
- So we still have a pretty hard distribution problem

**Symmetric key servers**
- Users share keys with server, server distributes session keys
- Symmetric key-exchange protocols, or channels
- Standard: Kerberos
- Drawback: central point of trust

**Certificates**
- A name and a public key, signed by someone else
  \[ C_A = \text{Sign}_S(A; K_A) \]
- Basic unit of transitive trust
- Commonly use a complex standard “X.509”

**Certificate authorities**
- “CA” for short: entities who sign certificates
- Simplest model: one central CA
- Works for a single organization, not the whole world

**Web of trust**
- Pioneered in PGP for email encryption
- Everyone is potentially a CA: trust people you know
- Works best with security-motivated users
  - Ever attended a key signing party?
CA hierarchies
- Organize CAs in a tree
- Distributed, but centralized (like DNS)
- Check by follow a path to the root
- Best practice: sub CAs are limited in what they certify

PKI for authorization
- Enterprise PKI can link up with permissions
- One approach: PKI maps key to name, ACL maps name to permissions
- Often better: link key with permissions directly, name is a comment

The revocation problem
- How can we make certs “go away” when needed?
- Impossible without being online somehow
  1. Short expiration times
  2. Certificate revocation lists
  3. Certificate status checking

Outline
- Cryptographic protocols, cont’d
- Some classic network attacks
- Key distribution and PKI
- Announcements intermission
- SSH
- SSL/TLS

Project 1 status
- Probably don’t need reminder that second submission is Friday
- Some clarifications on Piazza, consider asking more questions there
- I’ll be available for questions after class

Short history of SSH
- Started out as freeware by Tatu Ylönen in 1995
- Original version commercialized
- Fully open-source OpenSSH from OpenBSD
- Protocol redesigned and standardized for “SSH 2”

OpenSSH t-shirt
SSH host keys

- Every SSH server has a public/private keypair
- Ideally, never changes once SSH is installed
- Early generation a classic entropy problem
  - Especially embedded systems, VMs

Authentication methods

- Password, encrypted over channel
- .ssh/*: like .rhosts, but using client host key
- User-specific keypair
  - Public half on server, private on client
- Plugins for Kerberos, PAM modules, etc.

Old crypto vulnerabilities

- 1.x had only CRC for integrity
  - Worst case: when used with RC4
- Injection attacks still possible with CBC
  - CRC compensation attack
- For least-insecure 1.x-compatibility, attack detector
- Alas, detector had integer overflow worse than original attack

Newer crypto vulnerabilities

- IV chaining: IV based on last message ciphertext
  - Allows chosen plaintext attacks
- Better proposal: separate, random IVs
- Some tricky attacks still left
  - Send byte-by-byte, watch for errors
  - Of arguable exploitability due to abort
- Now migrating to CTR mode

SSH over SSH

- SSH to machine 1, from there to machine 2
  - Common in these days of NATs
- Better: have machine 1 forward an encrypted connection
  1. No need to trust 1 for secrecy
  2. Timing attacks against password typing

SSH (non-)PKI

- When you connect to a host freshly, a mild note
  - When the host key has changed, a large warning

Outline

- Cryptographic protocols, cont'd
- Some classic network attacks
- Key distribution and PKI
- Announcements intermission
- SSH
- SSL/TLS

SSL/TLS

- Developed at Netscape in early days of the public web
  - Usable with other protocols too, e.g. IMAP
- SSL 1.0 pre-public, 2.0 lasted only one year, 3.0 much better
- Renamed to TLS with RFC process
  - TLS 1.0 improves SSL 3.0
- TLS 11 and 12 in 2006 and 2008, only gradual adoption
### IV Chaining Vulnerability

- **TLS 1.0** uses previous ciphertext for CBC IV
- But, easier to attack in TLS:
  - More opportunities to control plaintext
  - Can automatically repeat connection
- "BEAST" automated attack in 2011: TLS 1.1 wakeup call

### Compression Oracle Vulnerability

- Compression oracle vulnerability: $\text{Compr}(S \parallel A)$, where $S$ should be secret and $A$ is attacker-controlled
- Attacker observes ciphertext length
- If $A$ is similar to $S$, combination compresses better
- Compression exists separately in HTTP and TLS

### But Wait, There's More!

- Too many vulnerabilities to mention them all in lecture
- Kaloper-Meršinjak et al. have longer list
  - "Lessons learned" are variable, though
  - Meta-message: don't try this at home

### HTTPS Hierarchical PKI

- Browser has order of 100 root certs
  - Not same set in every browser
  - Standards for selection not always clear
- Many of these in turn have sub-CAs
- Also, "wildcard" certs for individual domains

### Hierarchical Trust?

- No. Any CA can sign a cert for any domain
- A couple of CA compromises recently
- Most major governments, and many companies you've never heard of, could probably make a google.com cert
- Still working on: make browser more picky, compare notes

### CA vs. Leaf Checking Bug

- Certs have a bit that says if they're a CA
- All but last entry in chain should have it set
- Browser authors repeatedly fail to check this bit
- Allows any cert to sign any other cert

### MD5 Certificate Collisions

- MD5 collisions allow forging CA certs
- Create innocuous cert and CA cert with same hash
  - Requires some guessing what CA will do, like sequential serial numbers
  - Also 200 PS3s
- Oh, should we stop using that hash function?

### CA Validation Standards

- CA's job to check if the buyer really is foo.com
- Race to the bottom problem:
  - CA has minimal liability for bad certs
  - Many people want cheap certs
  - Cost of validation cuts out of profit
- "Extended validation" (green bar) certs attempt to fix
Many HTTPS security challenges tied with user decisions

Is this really my bank?

 Seems to be a quite tricky problem

 Security warnings often ignored, etc.