CSE/EE 461 – Lecture 15

David Wetherall
djw@cs.washington.edu
Last Time

- Naming

- Focus
  - How do we name hosts etc.?

- Topics
  - Domain Name System (DNS)
  - Email/URLs
This Time

- A whirlwind tour of network security

- Focus
  - How do we secure distributed systems?

- Topics
  - Privacy, integrity, authentication
  - Cryptography and key distribution
  - Firewalls and Denial-of-service
  - TCP/IP vulnerabilities
What do we mean by “Security”? 

- Networks are fundamentally shared 
  - Need means to protect legitimate participants in a distributed system from others with access to the network 

- Privacy: messages can’t be eavesdropped 
- Integrity: messages can’t be tampered with 
- Authenticity: messages were sent by the right party 

- Denial-of-Service: overwhelm system with bogus tasks causing service to be denied for legitimate tasks
Approaches at 10,000 ft

- Physical security
  - Tackle the problem of sharing directly
- “Security through obscurity”
  - Hope no-one will find out what you’re doing!
- Throw math at the problem
  - Cryptography

- Why is security difficult?
  - It’s a negative goal: can you be sure there are no flaws?
  - Often assumptions turn out to be invalid, esp. randomness
Basic Cryptography

- Cryptographer chooses functions $E$, $D$ and keys $K^E$, $K^D$
  - Mathematical basis
- Cryptanalyst try to “break”
  - Depends on what is known: $E$ and $D$, $M$ and $C$?
  - Attacks: traffic analysis, known plaintext, chosen plaintext
Secret Key Functions (DES, IDEA)

- Single key (symmetric) shared among parties
- Keys randomly chosen
  - But how do computers generate random numbers? Pitfall!
  - Ultimately need to tie to physical processes
Basics of DES

Each Round:

DES uses a 64 bit key (56 + 8)
Message encrypted 64 bits at a time
16 rounds in the encryption
Each round scrambles 64 bits

DES uses a 64 bit key (56 + 8)
Message encrypted 64 bits at a time
16 rounds in the encryption
Each round scrambles 64 bits
Repeat process for larger messages with "chaining"
Public and private key related mathematically

- Public key can be published; private is a secret
Basics of RSA

- To generate keys:
  - Choose two large prime numbers \( p \) and \( q \) \((\geq 256 \text{ bits})\). Let \( n = pq \)
  - Choose key \( e \) such that \( e \) and \((p-1) \times (q-1)\) are relatively prime.
  - Compute key \( d \) such that \( d = \frac{1}{e} \mod ((p-1) \times (q-1)) \)
  - Public key (encryption) is \((e, n)\), private key (decryption) is \((d, n)\)

- To use:
  - Encrypt: \( C = M^e \mod n \)
  - Decrypt: \( M = C^d \mod n \)

- Why it works:
  - \( M^{ed} = M \) in modulo arithmetic
  - Believe need to factor \( n \) into \( p \) and \( q \) to break and this is hard
Message Digests (MD5, SHA)

- Act as a cryptographic checksum or hash
  - Typically small compared to message (MD5 128 bits)
  - "One-way": infeasible to find two messages with same digest

```
Initial digest
```

```
Transform
```

```
Transform
```

```
512 bits
```

```
Message (padded)
```

```
512 bits
```

```
Message digest
```

```
Some Tradeoffs

- Number of brute force operations to crack depend on size of key
  - DES marginal now, 3DES used, RSA used with 1024 bit keys
  - 1977 RSA challenge solved after 17 years using the Internet 😊

- Message digests and private key encryption typically much faster than public key encryption
  - e.g., Peterson says MD5 100Mbps, DES 40Mbps, RSA 1Kbps
  - Can improve system performance by using RSA to transfer DES keys for use in a session, or RSA to authenticate digests only

- Also, compress before encrypting 😊
Authentication Protocols

- Three-way handshake for mutual authentication
  - Client and server share secrets, e.g., login password

Client:
- ClientId, $E(x, \text{CHK})$
- $E(x + 1, \text{SHK}), E(y, \text{SHK})$
- Session key exchanged

Server:
- $E(y + 1, \text{CHK})$
- $E(SK, \text{SHK})$

Client authenticates server here

Server authenticates client here
Via Trusted Third Party (Kerberos)

Authentication server

A authenticates B

A, B

E((T, L, K, B), K_A),
E((T, L, K, A), K_B)

E((A, T), K),
E((T, L, K, A), K_B)

B authenticates A

E(T + 1, K)
Public Key Authentication

A authenticates B

(similarly for B to authenticate A)

E(x, PublicB)
Message Integrity Protocols

- Sometimes we don’t care about privacy but do care about integrity/authenticity

- Digital signatures (RSA)
  - Sign message with private key (encrypt); others verify with public key (decrypt)

- MD5 with RSA
  - Send signed digest of message along with message

- Keyed MD5
  - Send digest of message plus shared secret along with message

- Last two methods increase performance
Key Distribution

- Public key systems depend on the distribution of keys!
  - Public Key Infrastructures (PKIs), e.g., Verisign
  - An Achilles heel?
- Certificates (X.509)
  - Distribute keys by trusted certification authority (CA)
    - “I swear X’s public key is Y”, signed by CA
  - Still requires bootstrapping ...
  - Also allows us to can build chains of trust
    - e.g., public keys for a domain name so that “.edu” (root) certifies “washington.edu”s key, they certify “cs...”’s key ...
  - Certificate Revocation Lists needed to “undo” associations!
Example Systems

- **Pretty Good Privacy (PGP)**
  - For authentic and confidential email
- **Secure Sockets (SSL) and Secure HTTP (HTTPS)**
  - For secure Web transactions
- **IP Security (IPSEC)**
  - Framework for encrypting/authenticating IP packets
PGP

- Application level system
- Based on public keys and a “grass roots” Web of trust
- Sign messages for integrity/authenticity
  - Encrypt with private key of sender
- Encrypt messages for privacy
  - Could just use public key of receiver ...
  - But encrypt message with secret key, and secret key with public key of receiver to boost performance
SSL/TLS and HTTPS

- Secure transport layers targeted at Web transactions
  - SSL/TLS inserted between TCP and HTTP to make secure HTTP

- Extra handshake phase to authenticate and exchange shared session parameters
  - Such as secret keys used for encryption
  - Client might authenticate Web server but not vice-versa
    - Certificate Authority embedded in Web browser

- Performance optimization
  - Refer to shared state with session id
  - Can use same parameters across connections
    - Client sends session id, allowing server to skip handshake
IPSEC

- Framework for encrypted and authenticated IP packets
  - Choice of algorithms not specified
- Uses new protocol headers inside IPv4 packets
  - Authentication header
    - For message integrity and origin authenticity
    - Optionally “anti-replay” protection (via sequence number)
  - Encapsulating Security Payload
    - Adds encryption for privacy
- Depends on key distribution (ISAKAMP)
  - Sets up security associations
- Example use: secure tunnels between corporate offices
Filter-based Firewalls

- Sit between site and rest of Internet, filter packets
  - Enforce site policy in a manageable way
  - e.g. pass (*, *, 128.7.6.5, 80), then drop (*, *, *, 80)
  - Rules may be added dynamically to pass new connections

- Sometimes called a “level 4” switch
  - Acts like a router (accepts and forwards packets)
  - But looks at information up to TCP port numbers (layer 4)
Proxy-Based Firewalls

- Problem: Filter ruleset can be complex/insufficient
  - Adequate filtering may require application knowledge
- Run proxies for Web, mail, etc. just outside firewall
  - In the “de-militarized zone” DMZ
  - External requests go to proxies, only proxies connect inside
    - External user may or may not know this is happening
    - Proxies filter based on application semantics
Denial-of-Service Attacks

- Attacker can deny service to legitimate users if they can overwhelm the system providing the service
  - System has limited bandwidth, CPU, memory, etc. resources
  - Just sent it too many packets to handle ...

- Made more devastating by focusing on specific, limited resources and distributed nature of attacks
  - e.g., How many TCP connections can be open?
  - Today, build implementation to tolerate DOS
  - Tomorrow, design protocols to tolerate better, possibly network support for shutting down attack?
TCP/IP Vulnerabilities

- Low-level specifics in TCP/IP used to wreak havoc, especially if implementation is weak or buggy
- Many incidents of buffer over-runs
  - Attacker can send packet to crash or compromise host
- IP fragmentation:
  - End-system hangs on to fragments hoping to re-assemble ...
  - But for how long? Attacker can exhaust memory
  - Similarly, state created for TCP SYN packet
- Smurfing:
  - Send ICMP echo request to broadcast address with fake source address; source gets hosed
- These are just some representative examples ... many more (CERT)
Key Concepts

- Privacy, message integrity, origin authenticity
- Cryptographic mechanisms are used to support these properties: private key, public key and digests

- Firewalls are in widespread use today
- Denial-of-service consumes system resources