Lecture 10
October 12, 2022
Link and End-to-End Protocols

Link Protocol

End-to-End (or E2E) Protocol
Encryption

• Link encryption
  • Each host enciphers message so host at “next hop” can read it
  • Message can be read at intermediate hosts

• End-to-end encryption
  • Host enciphers message so host at other end of communication can read it
  • Message cannot be read at intermediate hosts
Examples

• SSH protocol
  - Messages between client, server are enciphered, and encipherment, decipherment occur only at these hosts
  - End-to-end protocol

• PPP Encryption Control Protocol
  - Host gets message, deciphers it
    - Figures out where to forward it
    - Enciphers it in appropriate key and forwards it
  - Link protocol
Cryptographic Considerations

- Link encryption
  - Each host shares key with neighbor
  - Can be set on per-host or per-host-pair basis
    - Windsor, stripe, seaview each have own keys
    - One key for (windsor, stripe); one for (stripe, seaview); one for (windsor, seaview)

- End-to-end
  - Each host shares key with destination
  - Can be set on per-host or per-host-pair basis
  - Message cannot be read at intermediate nodes
Traffic Analysis

• Link encryption
  • Can protect headers of packets
  • Possible to hide source and destination
    • Note: may be able to deduce this from traffic flows

• End-to-end encryption
  • Cannot hide packet headers
    • Intermediate nodes need to route packet
  • Attacker can read source, destination
Adding Security to Email

• Goal: provide privacy (confidentiality), authentication of origin, and integrity checking for email

• Two systems
  • Privacy-Enhanced Electronic Mail (PEM)
  • PGP, GPG, OpenPGP — all basically the same

• Ideas underlying both protocols are the same
  • PEM is older and simpler; not used much today
  • PGP/GPG/OpenPGP newer, used widely

• Here, discuss PEM and show differences between it and OpenPGP
Design Principles

• Do not change related existing protocols
  • Cannot alter SMTP

• Do not change existing software
  • Need compatibility with existing software

• Make use of PEM optional
  • Available if desired, but email still works without them
  • Some recipients may use it, others not

• Enable communication without prearrangement
  • Out-of-bands authentication, key exchange problematic
Basic Design: Keys

• Two keys
  • *Interchange keys* tied to sender, recipients and is static (for some set of messages)
    • Like a public/private key pair (indeed, may be a public/private key pair)
    • Must be available *before* messages sent
  • *Data exchange keys* generated for each message
    • Like a session key, session being the message
Basic Design: Confidentiality

Confidentiality:

- $m$ message
- $k_B$ Bob’s interchange key (his public key, in a public key system)
Basic Design: Integrity

Integrity and authentication:

- $m$ message
- $h(m)$ hash of message $m$ — Message Integrity Check (MIC)
- $k_A$ Alice’s interchange key (her private key, in a public key system)

Non-repudiation: if $k_A$ is Alice’s private key, this establishes that Alice’s private key was used to sign the message
Basic Design: Everything

Confidentiality, integrity, authentication:
• Notations as in previous slides
• If $k_A$ is Alice’s private key, get non-repudiation too

$$\{ m \} k_s || \{ h(m) \} k_A || \{ k_s \} k_B$$

Alice $\rightarrow$ Bob
Practical Considerations

• Limits of SMTP
  • Only ASCII characters, limited length lines

• Use encoding procedure
  1. Map local char representation into canonical format
     – Format meets SMTP requirements
  2. Compute and encipher MIC over the canonical format; encipher message if needed
  3. Map each 6 bits of result into a character; insert newline after every 64th character
  4. Add delimiters around this ASCII message
Problem

- Recipient without PEM-compliant software cannot read it
  - If only integrity and authentication used, should be able to read it
- Mode MIC-CLEAR allows this
  - Skip step 3 in encoding procedure
  - Problem: some MTAs add blank lines, delete trailing white space, or change end of line character
  - Result: PEM-compliant software reports integrity failure
PEM vs. OpenPGP

• Use different ciphers
  • PGP allows several ciphers
    • Public key: RSA, El Gamal, DSA, Diffie-Hellman, Elliptic curve
    • Symmetric key: IDEA, Triple DES, CAST5, Blowfish, AES-128, AES-192, AES-256, Twofish-256
    • Hash algorithms: MD5, SHA-1, RIPE-MD/160, SHA256, SHA384, SHA512, SHA224
  • PEM allows RSA as public key algorithm, DES in CBC mode to encipher messages, MD2, MD5 as hash functions
PEM vs. OpenPGP

- Use different key distribution models
  - PGP uses general “web of trust”
  - PEM uses hierarchical structure

- Handle end of line differently
  - PGP remaps end of line if message tagged “text”, but leaves them alone if message tagged “binary”
  - PEM always remaps end of line
Authentication Basics

• Authentication: binding of identity to subject
  • Identity is that of external entity (my identity, Matt, etc.)
  • Subject is computer entity (process, etc.)
Establishing Identity

• One or more of the following
  • What entity knows (eg. password)
  • What entity has (eg. badge, smart card)
  • What entity is (eg. fingerprints, retinal characteristics)
  • Where entity is (eg. In front of a particular terminal)
Authentication System

- \((A, C, F, L, S)\)
  - A: information that proves identity
  - C: information stored on computer and used to validate authentication information
  - \(F\): complementation function; for \(f \in F, f : A \to C\)
  - \(L\): functions that prove identity; for \(l \in L, l : A \times C \to \{\text{true, false}\}\)
    - \(l\) is lowercase “L”
  - \(S\): functions enabling entity to create, alter information in \(A\) or \(C\)
Example

- Password system, with passwords stored on line in clear text
  - A set of strings making up passwords
  - $C = A$
  - $F$ singleton set of identity function \( \{ I \} \)
  - $L$ single equality test function \( \{ eq \} \)
  - $S$ function to set/change password
Passwords

• Sequence of characters
  • Examples: 10 digits, a string of letters, *etc.*
  • Generated randomly, by user, by computer with user input

• Sequence of words
  • Examples: pass-phrases

• Algorithms
  • Examples: challenge-response, one-time passwords
Storage

• Store as cleartext
  • If password file compromised, *all* passwords revealed

• Encipher file
  • Need to have decipherment, encipherment keys in memory
  • Reduces to previous problem

• Store one-way hash of password
  • If file read, attacker must still guess passwords or invert the hash
Example

• UNIX system original hash function
  • Hashes password into 11 char string using one of 4096 hash functions

• As authentication system:
  • \( A = \{ \text{strings of 8 chars or less} \} \)
  • \( C = \{ \text{2 char hash id} | | \text{11 char hash} \} \)
  • \( F = \{ \text{4096 versions of modified DES} \} \)
  • \( L = \{ \text{login, su, ...} \} \)
  • \( S = \{ \text{passwd, nispasswd, passwd+, ...} \} \)
Anatomy of Attacking

• Goal: find $a \in A$ such that:
  • For some $f \in F$, $f(a) = c \in C$
  • $c$ is associated with entity

• Two ways to determine whether $a$ meets these requirements:
  • Direct approach: as above
  • Indirect approach: as $l(a)$ succeeds iff $f(a) = c \in C$ for some $c$ associated with an entity, compute $l(a)$
Preventing Attacks

• How to prevent this:
  • Hide one of $a$, $f$, or $c$
    • Prevents obvious attack from above
    • Example: UNIX/Linux shadow password files hides $c$’s
  • Block access to all $l \in L$ or result of $l(a)$
    • Prevents attacker from knowing if guess succeeded
    • Example: preventing any logins to an account from a network
      • Prevents knowing results of $l$ (or accessing $l$)
Picking Good Passwords

• “WtBvStHbChCsLm?TbWtF.+FSK”
  - Intermingling of letters from Star Spangled Banner, some punctuation, and author’s initials
• What’s good somewhere may be bad somewhere else
  - “DCHNH,DMC/MHmh” bad at Dartmouth ("Dartmouth College Hanover NH, Dartmouth Medical Center/Mary Hitchcock memorial hospital"), ok elsewhere (probably)
• Why are these now bad passwords? 😞
Passphrases

• A password composed of multiple words and, possibly, other characters

• Examples:
  • “home country terror flight gloom grave”
    • From Star Spangled Banner, third verse, third and sixth line
  • “correct horse battery staple”
    • From xkcd

• Caution: the above are no longer good passphrases
Remembering Passphrases

- Memorability is a good example of how environment affects security
  - Study of web browsing shows average user has 6-7 passwords, sharing each among about 4 sites (from people who opted into a study of web passwords)
    - Researchers used an add-on to a browser that recorded information about the web passwords but not the password itself
- Users tend not to change password until they know it has been compromised
  - And when they do, the new passwords tend to be as short as allowed
- Passphrases seem as easy to remember as passwords
  - More susceptible to typographical errors
  - If passphrases are text as found in normal documents, error rate drops
Password Manager (Wallet)

• A mechanism that encrypts a set of user’s passwords
• User need only remember the encryption key
  • Sometimes called “master password”
  • Enter it, and then you can access all other passwords
• Many password managers integrated with browsers, cell phone apps
  • So you enter the master password, and password manager displays the appropriate password entry
  • When it does so, it shows what the password logs you into, such as the institution with the server, and hides the password; you can then have it enter the password for you