CIS 551 / TCOM 401

Computer and Network Security

Spring 2007

Lecture 19

Announcements

- Reminder: Project 3 is available on the web pages
  - Handout for SDES needed for the project
  - Due: April 3rd

Secure Shell (SSH)

- Secure Shell (SSH) is a program to log into another computer over a network, to execute commands in a remote machine, and to move files from one machine to another.
- It provides strong authentication and secure communications over unsecure channels.
- It is intended as a replacement for telnet, rlogin, rsh, and rcp.

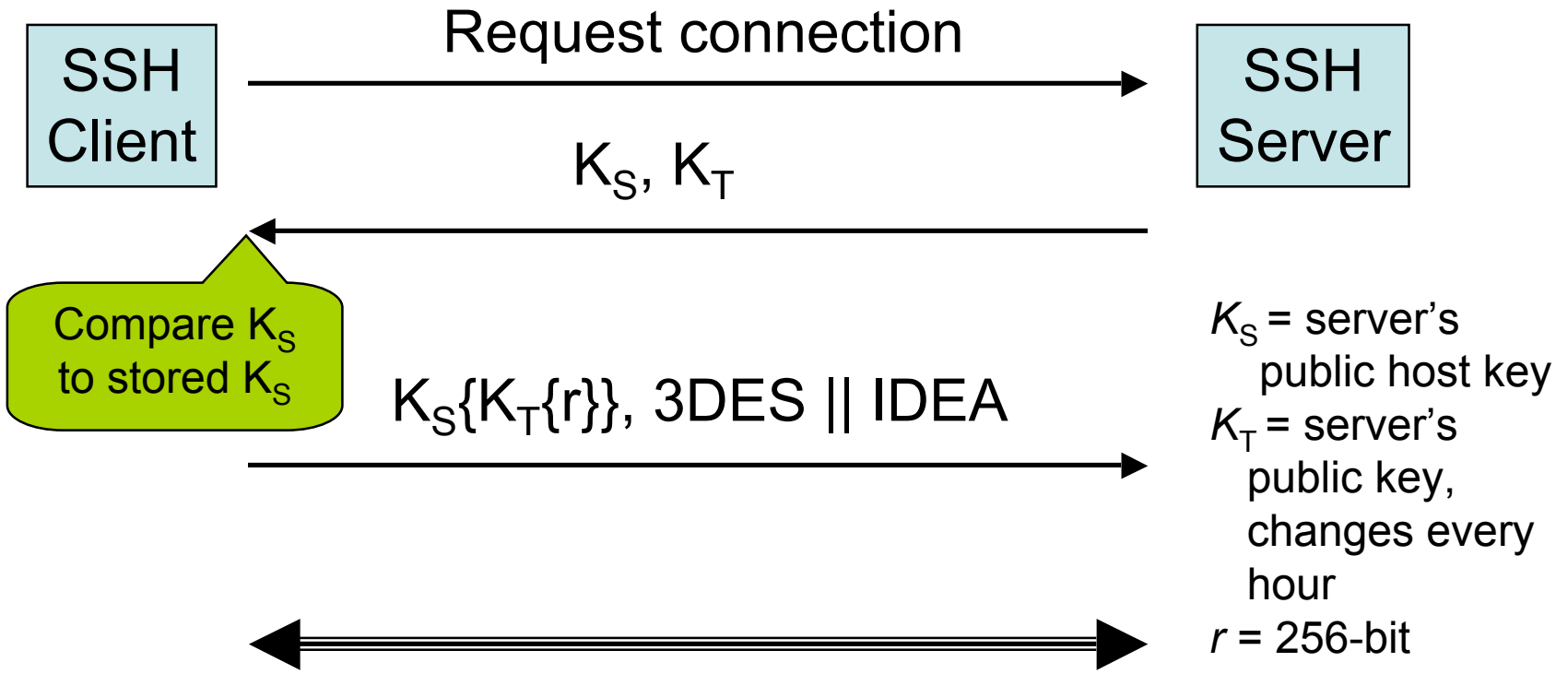
SSH protocol (overview)

- See: http://www.snailbook.com/protocols.html
  - RFC's 4250,4251,4252,4253,4254
- Connection Setup / Version number exchange
- Session key exchange / Server Authentication
  - Each side sends a list of preferred algorithms (e.g. Diffie-Hellman with certain parameters)
  - Guess which algorithm is used by other side
  - Optimistically send first message of key exchange (if guess is wrong the recipient will ignore it)
  - Key exchange produces a shared key K and exchange hash H (used as a session identifier)
  - Server authenticates by signing the hash H

SSH Protocol Continued

- Client Authentication
  - Negotiate an authentication mechanism
  - Public key (RSA or DSA)
    - Keys created using ssh-keygen facility
    - Stored in ~/.ssh/identity.pub
  - Password
  - Kerberos
  - /etc/hosts.equiv
- Transport Protocol
  - Negotiate encryption type
- Connection Protocol (for shells)

SSH Protocol



Encryption Ciphers

SSH uses the following ciphers for encryption (with varying options for key sizes):

Cipher	SSH1	SSH2
• DES	yes	no
• 3DES	yes	yes
• IDEA	yes	yes
• Blowfish	yes	yes
• Twofish	no	yes
• Arcfour	no	yes
• AES	no	yes
• Serpent	no	yes
• Cast128-cbc	no	yes

SSH Protection

- ssh protects against:
  - IP spoofing, where a remote host sends out packets which pretend to come from another, trusted host.
  - DNS spoofing, where an attacker forges name server records
  - Interception of cleartext passwords and other data by intermediate hosts
  - Modification of data by people in control of intermediate hosts
  - Attacks based on listening to X authentication data and spoofed connections to an X11 server
- In other words, ssh never trusts the net; somebody hostile who has taken over the network can only force ssh to disconnect, but cannot decrypt traffic, play back the traffic, or hijack the connection.

SSH vs TELNET and RSH

Security

Telnet/rsh sends all communications in cleartext

SSH encrypts all communications and optionally compresses

X/Port Forwarding

Makes it easy to run remote X applications

(xterm, netscape)

Can "tunnel" connections between two hosts

Other Features

Password-less logins via public/private key encryption

Secure file copy (scp/sftp) - replacement for ftp/rcp

Example Use

Logging into hosts:

$ **ssh -l username hostname**
$ **ssh username@hostname**
$ **ssh hostname**

Example:

```
$ ssh stevez@eniac.seas.upenn.edu uptime
The authenticity of host 'eniac.seas.upenn.edu (158.130.64.177)'
  can't be established.
RSA key fingerprint is bf:b1:e4:01:4c:d3:69:e2:83:8b:8d:f9:b7:06:a3:a9.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'eniac.seas.upenn.edu' (RSA) to the list of
  known hosts.
stevez@eniac.seas.upenn.edu's password: <PASSWORD>
 10:36am  up 31 day(s), 17:47,  72 users,  load average: 0.17, 0.19, 0.20
```

SSH1 vs. SSH2

SSH1

- Uses RSA, had patent issues in US
  - Also supports 3DES and Blowfish
  - Some support IDEA, but OpenSSL lacks it
- Uses CRC for data integrity
  - Flawed, attacks possible
  - Less of a factor when using 3DES

SSH2

- Uses DSA, supported best in commercial SSH
  - Not restricted by patents
- Uses a different approach to get around CRC issues

SSH1 and SSH2 are not compatible with each other.

Man vs. Machine

- Machine
  - Good at authenticating other machines
  - Good at mathematical manipulations, etc.
  - Can handle keys, secrets, etc.
  - Very good memory of things stored in it
- Man
  - Good at identifying people
  - Use small clues that when combined yield an unmistakable picture
    - Voice
    - Height

Authenticating Humans: Foundations

- Authentication is based on one or more of the following:
- Something you know
  - password
- Something you have
  - driver's license, Penn Card
- Something inherent about you
  - Biometrics, location

- What's the most common method of authentication?

Passwords

- Shared code/phrase
- Client sends to authenticate

- Simple, right?
- How do you…
  - Establish them to begin with?
  - Stop them from leaking?
  - Stop them from being guessed?

SOURCE: NASA

http://www.captcha.net/

Prime Mover Problem

- Out of band
  - Physical mail
  - Email
  - Attached to the box
- Piggybacking
  - Swipe Penn Card to make PennKey
  - But where does the chain stop?
    - Penn Card -> drivers license -> birth certificate

Leaks

- Social engineering
  - Warnings
- Legal and responsibility
  - Shared password == shared liability
- Writing the password down on paper

Guessing

- The "no such user" mistake
- The "here's who we are" mistake
- Common words, phrases for passwords
- Null passwords, "password", username, backwards, etc.
- Dictionary attacks

- How bad is it?

1979 Survey of 3,289 Passwords

- With no constraints on choice of password, Morris and Thompson got the following results:
  - 15 were a single ASCII letter.
  - 72 were strings of two ASCII letters.
  - 464 were strings of three ASCII letters.
  - 47 were strings of four alphanumerics.
  - 706 were five letters, all upper-case or all lower-case.
  - 605 were six letters, all lower case.

1990s Surveys of 15K Passwords

- Klein (1990) and Spafford (1992)
  - 2.7% guessed in 15 minutes
  - 21% in a week
  - Sounds ok? Not if the passwords last 30 days
- Tricks
  - Letter substitutions, words backwards, common names, patterns, etc.
  - Anything you can think of off the top of your head, a hacker can think of too
- Lazy users!
  - Weakest link is always the way of the attack

Heuristics for Guessing Attacks

- The dictionary with the words spelled backwards
- A list of first names (best obtained from some mailing list). Last names, street names, and city names also work well.
- The above with initial upper-case letters.
- All valid license plate numbers in your state. (About 5 hours work in 1979 for New Jersey.)
- Room numbers, social security numbers, telephone numbers, and the like.

What makes a good password?

- Password Length
  - 64 bits of randomness is hard to crack
  - 64 bits is roughly 20 “common” ASCII characters
  - But… People can’t remember random strings
  - Longer not necessarily better: people write the passwords down
- Pass phrases
  - English Text has roughly 1.3 random bits/char.
  - Thus about 50 letters of English text
  - Hard to type without making mistakes!

- In practice
  - Non-dictionary, mixed case, mixed alphanumeric
  - Not too short (or too long)

Hacks on plaintext password file

- Is the password file readable by the OS?
  - Then if I break the OS
- Can privileged users see the file?
  - … and make copies
- Is the file backed up somewhere
  - … insecure?
- Is the file in plaintext somewhere in memory?
  - Core dump
- Fool the user
  - A program that masquerades as the authentication program

Counter-hacks

- Control-Alt-Del for logging in
  - For windows only
- Slow down
  - Make guessing take too long
- Encrypt the password file
  - “Salt" - to prevent duplicates
  - Use one way hashes or encryptions on the passwords
- Password rules
  - Min length, upper and lower case, no common words
  - Use letters and numbers and symbols
  - Change often
  - Keep a password history
  - Don't write it down!

Add Salt

- "Salt" the passwords by adding random bits.
  - Decreases the likelihood that two identical passwords will appear as identical entries in the password file.
- 12 bit salt results in 4,096 versions of each password.
- /etc/passwd entry:

user_id	$salt_u$	$Hash(salt_u + passwd_u)$	…

- Actually most modern implementations use so-called *shadow* password files /etc/shadow that aren't world readable.

One Time Passwords

- Shared lists.
- Sequentially updated.
- One-time password sequences based on a one-way (hash) function.

- Used in practice: SKey mechanism

Hash-based 1-time Passwords

- Alice identifies herself to verifier Bart using a well-known one-way hash function H.

- One-time setup.
  - Alice chooses a secret w.
  - Fixes a constant t for the number of times the authentication can be done.
  - Alice securely transfers $H^t(w)$ to Bart

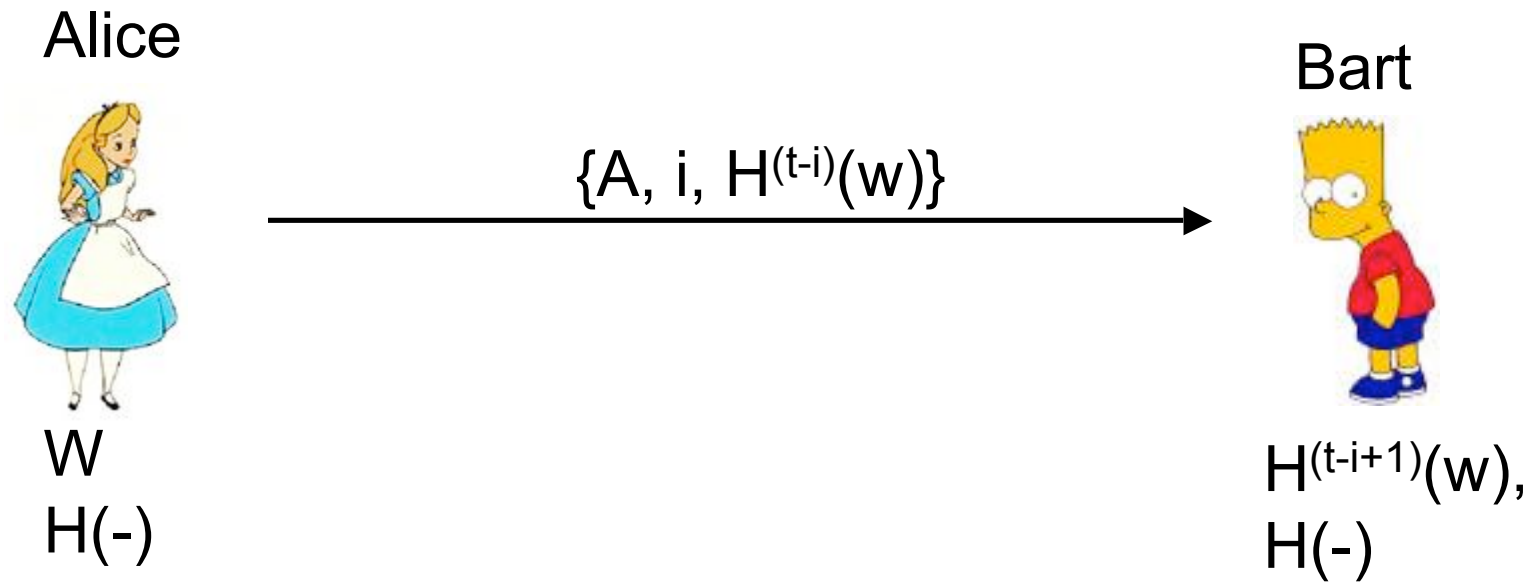
$$\underbrace{H(H(H\ldots(H(w))\ldots))}_{t \text{ times}}$$

Hash-based 1-time Passwords

- Protocol actions. For session i, claimant A does the following to identify itself:
  - A computes w' = $H^{(t-i)}(w)$ and transmits the value to B.
  - B checks that i is the correct session (i.e. that the previous session was i-1) and checks to see if H(v) = w' where v was the last value provided by A (as part of session i-1).
  - B saves w' and i for use in the next session.

- It's hard to compute x from H(x).
  - Even though attacker gets to see $H^{(t-i)}(x)$, they can't guess then next message $H^{(t-(i+1))}(x)$.

One-time passwords: i^{th} authentication



- Alice does the following to identify herself:
  - A computes w' = $H^{(t-i)}(w)$ and transmits the value to B.
  - B checks that i is the correct session (i.e.. that the previous session was i-1) and checks to see if H(w') = v where v was the last value provided by A (as part of session i-1).
  - B saves w' and i for use in the next session.