

CSE331: Introduction to Networks and Security

Lecture 30

Fall 2006

Announcements

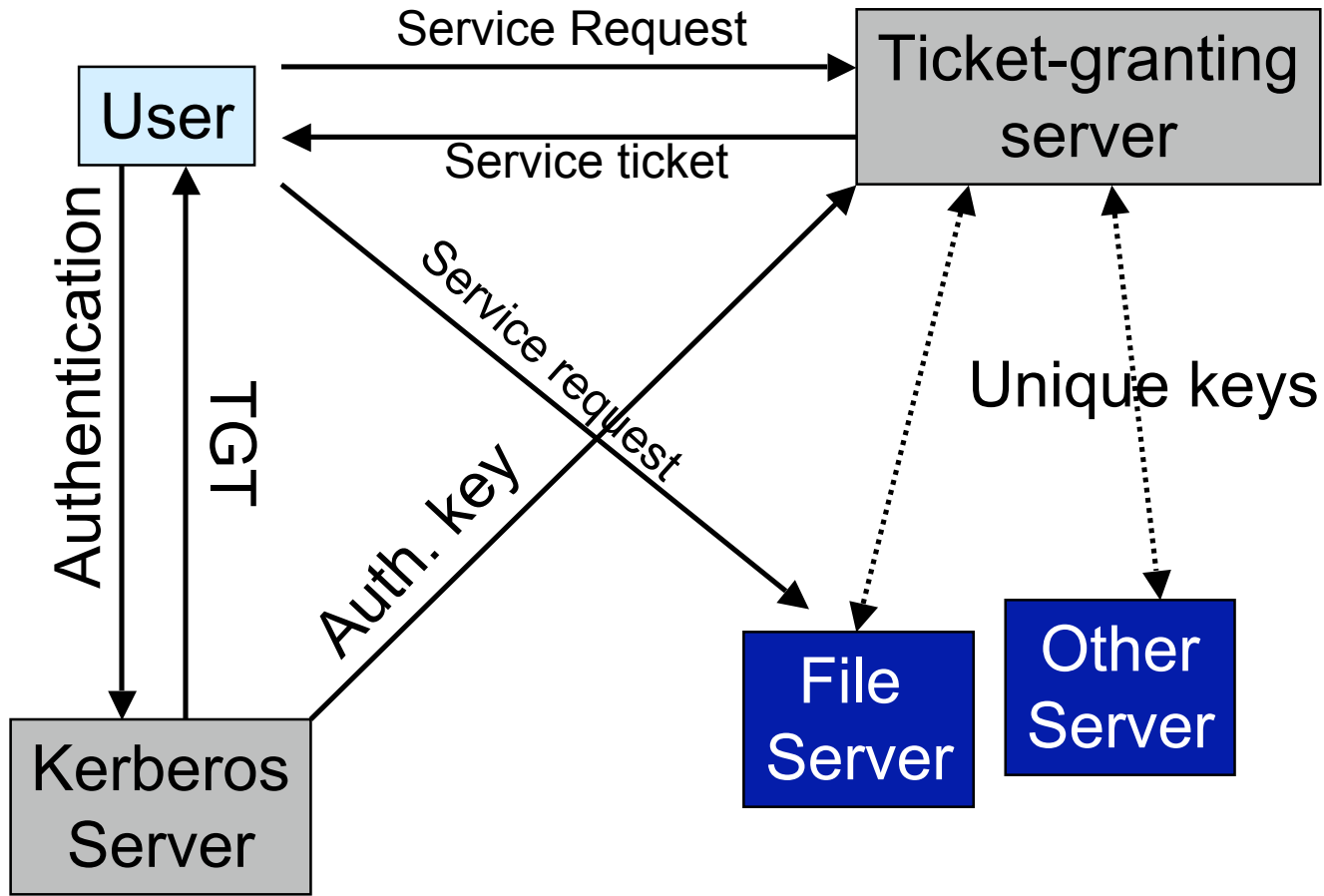
- Project 4 is on the web
 - Due last day of classes at midnight
 - Implementing cryptographic protocols in Java

- There was a bug in solutions for HW 2 problem 1(e)
 - Check out the new solutions on the web
 - If you were docked points for a correct answer, submit your HW to me for a regrade.

Kerberos

- Key exchange protocol developed at MIT in the late 1980's
- Central server provides “tickets”
- *Tickets* – (also known as *capabilities*):
 - Unforgeable
 - Nonreplayable
 - Authenticated
 - Represent authority
- Designed to work with NFS (network file system)
- Also saves on authenticating for each service
 - e.g. with rlogin or rsh.

Kerberos



Kerberos Login

- U = User's machine
- S = Kerberos Server
 - Has a database of user passwords: $\text{userID} \rightarrow \text{pwd}$
- G = Ticket granting server
- $U \rightarrow S : \text{userID}, G, n_U$
- $S \rightarrow U : k_{\text{pwd}}\{n_U, K_{UG}\}, K_{SG}\{T(U,G)\}$
- $S \rightarrow G : K_{SG}\{K_{UG}, \text{userID}\}$

Kerberos ticket
granting ticket

Session key

Ticket lifetime

Kerberos Service Request

- $U \rightarrow G : \text{userID}, K_{SG}\{T(U,G)\}, \text{req}(F), n'_U$
- $G \rightarrow U : K_{UG}\{K_{UF}, n'_U\}, K_{FG}\{T(U,F)\}$
- $U \rightarrow F : \text{userID}, K_{FG}\{T(U,F)\}$



Kerberos Benefits

- Distributed access control
 - No passwords communicated over the network
- Cryptographic protection against spoofing
 - All accesses mediated by G (ticket granting server)
- Limited period of validity
 - Servers check timestamps against ticket validity
 - Limits window of vulnerability
- Timestamps prevent replay attacks
 - Servers check timestamps against their own clocks to ensure “fresh” requests
- Mutual authentication
 - User sends nonce challenges

Kerberos Drawbacks

- Requires available ticket granting server
 - Could become a bottleneck
 - Must be reliable
- All servers must trust G, G must trust servers
 - They share unique keys
- Kerberos requires synchronized clocks
 - Replay can occur during validity period
 - Not easy to synchronize clocks
- User's machine could save & replay passwords
 - Password is a weak spot
- Kerberos does not scale well
 - Hard to replicate authentication server and ticket granting server
 - Duplicating keys is bad, extra keys = more management

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.

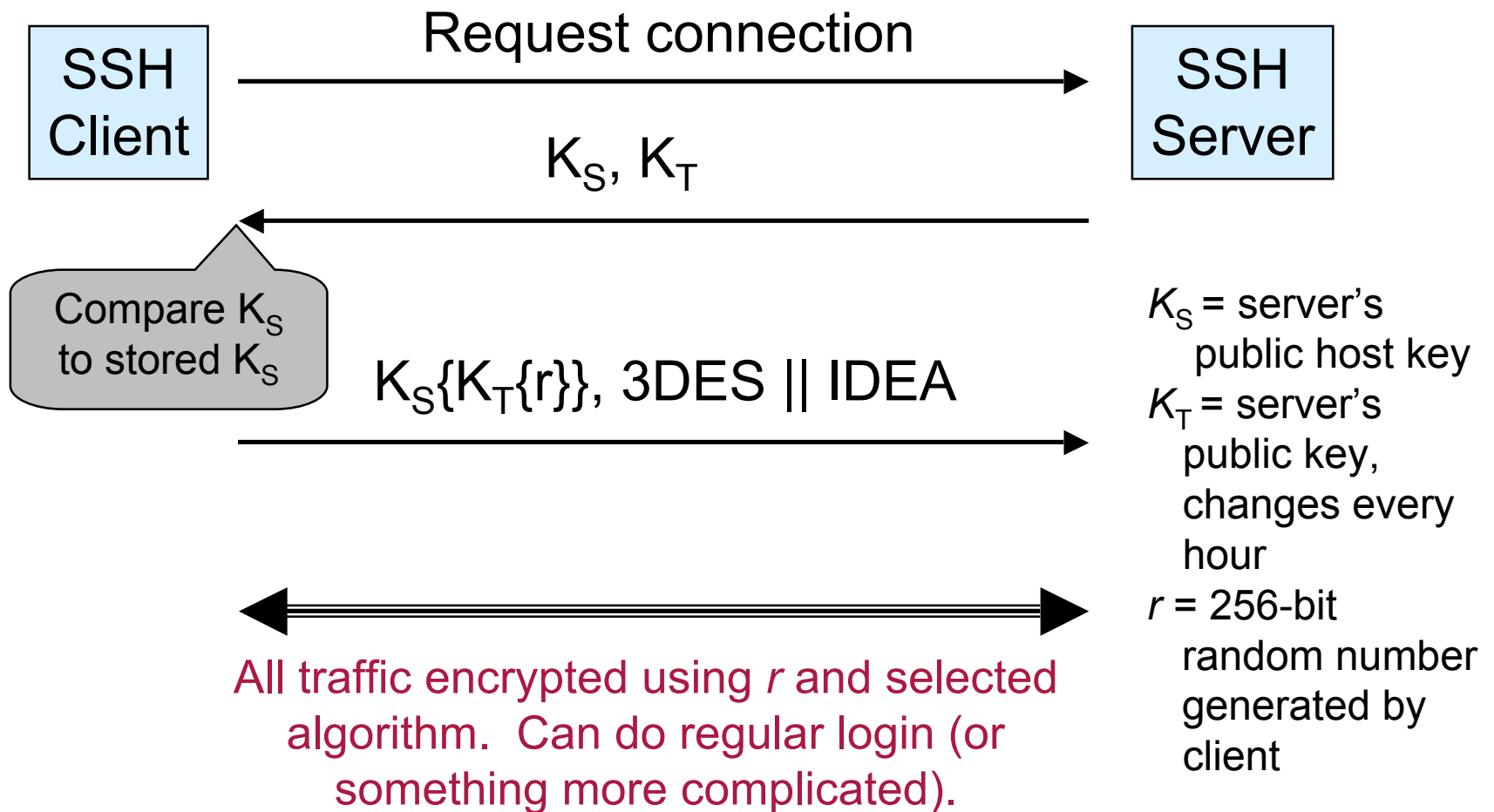
SSH protocol (overview)

- 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 (from the README):
 - **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, until recently 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.