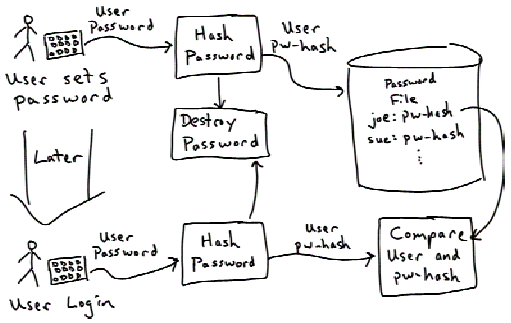
Authentication and Kerberos

User auth on a single machine



Single host password authentication

- More-or-less secure because:
  - Path is physically secure (from keyboard to computer a few feet away)
    - And password is in the clear only briefly
  - Attackers can't derive password from the one-way hash in the file

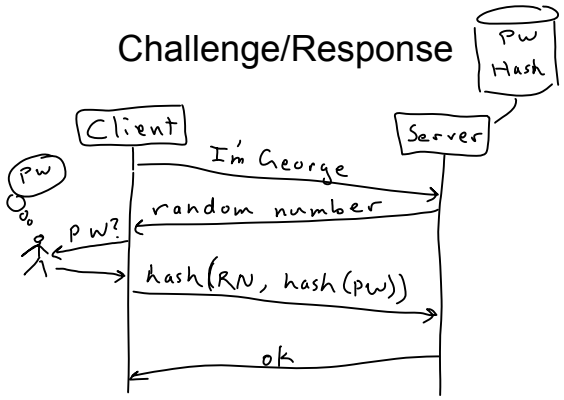
Single host password authentication

- But. . . .
  - The attacker can read the password file, and do a password-guessing attack
    - Guess password, hash it, matches hash in file if guess is correct
  - Attacker could spoof the login dialog (phishing)
  - A virus could monitor your keyboard strokes
- Ultimately, password-based systems suck

What about authentication over the network?

- Used to be, passwords sent in the clear!
- HTTPS (SSL) gives us a secure connection over which we can do a login dialog
  - As long as we trust the cert
- Other methods:
  - Challenge/Response
  - One-time passwords

Challenge/Response



Challenge/Response

- No replay attack
  - If random number suitably random
- Dictionary attack is possible
  - If exchange is eve-dropped
  - Run over SSL (or any Diffie-Hellman)

One-time Password

- Client and Server have identical lists of onetime passwords
  - Plus a user password
- Can be generated with time-synchronized pseudo-random number generators
- Or a list calculated in advance
- Attacker must obtain both list and password

What is Kerberos?

- A network authentication system:
- Allows users on client hosts to authenticate themselves to server hosts
  - I.e., allows a server to know that the user is who he says he is
- Assumes that users and hosts are untrusted
  - Clients and servers are physically accessible, and may have been compromised by attackers

What is Kerberos?

- Designed at MIT in the 80's
  - As part of a larger campus computing system called Athena
- Assumes that students will try to exploit the system
  - And that students are capable!
- By protecting the system from inside attackers, it also protects from outside attackers
- This (largely correct) notion that security must be pervasive drove the anti-firewall sentiment within IETF
  - In fact, firewalls and internal authentication systems are complementary technologies

What is Kerberos?

- Kerberos had a huge impact on subsequent security systems
  - For instance, Windows NT used a variant
- Kerberos is still widely used
  - Cornell's "sidecar" system uses Kerberos
- Designed as a toolkit with an API
  - Applications can use it however they please
  - Applications must be modified to use it, but then this is inevitable…

Kerberos model

- Kerberos was originally based on symmetric keys, now includes public keys
  - Public keys were patent protected, and there may have been other reasons?
- The Kerberos service runs on physically protected machines
  - But all Kerberos client systems (I.e. client and server hosts) are accessible
- The Kerberos service knows (a one-way hash of) all passwords
  - Users and servers know their own passwords only

Kerberos model

- Passwords never cross the network in the clear (of course!)
- Users type in password at login time, but not subsequently
  - (I.e., they don't have to type in the password again when they access authenticated services)
- Why?
  - Convenient for the user, but . . .
  - More importantly: minimizes the number of times the password ever exists in the clear

Minimizing clear passwords

- Password is in the clear:
  - As the user types it
    - Someone looking over your shoulder may see it
  - As the computer reads it and puts it in a buffer
    - An untrusted super-user could read this memory
- Kerberos minimizes the number of times that the password itself is used
- Kerberos never puts a user password on a host disk (even temporarily), and keeps it in memory for as short a time as possible
  - And over-writes the memory afterwards

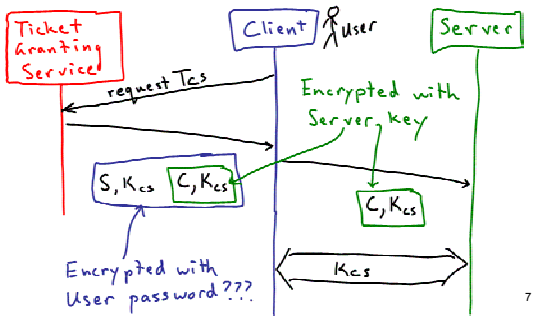
Kerberos password authentication

- But the path from a client to the Kerberos server is not physically secure
  - So, ultimately the Kerberos server must keep a copy of (a hash of) the password!
- This is why the Kerberos servers must be physically secure…

Kerberos Ticket

- When a client wants to talk to a server, Kerberos gives both client and server a "ticket"
- The ticket does two things:
  - Authenticates the client to the server (and optionally vice versa)
  - Provides a session key that the client and server can subsequently use (if they want)

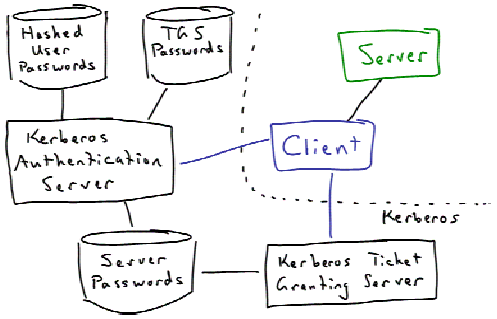
Very rough idea of Kerberos ticket (naïve version)



Problem with naïve version

- Client required storage of the user key to decrypt the (outer) ticket
- But, don't want to keep the user key on the client host
- And, don't want to have to ask the user for the password every time the user wants to access a new service

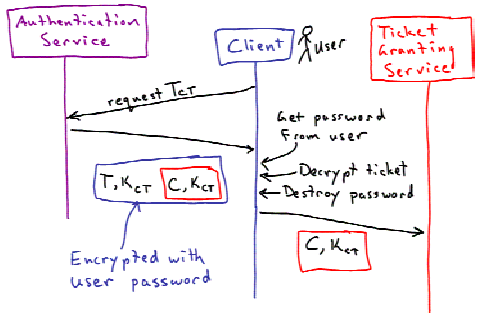
In fact, Kerberos has two types of servers…



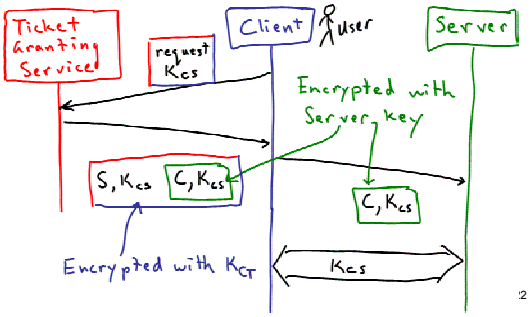
Authentication and Ticket Granting Servers

- At login, user's client goes to the Authentication Server to get a session key that allows it to talk to the TGS
  - This is the only time the user's password is needed
- Subsequently, the TGS session key is used to get tickets to talk to servers

Getting a Ticket Granting Ticket



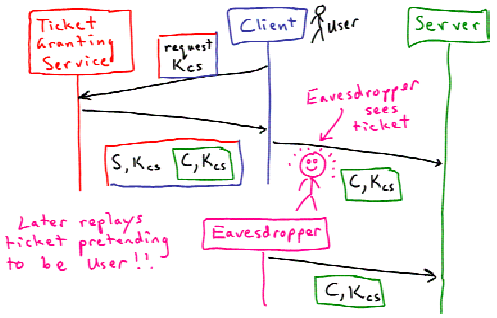
Using a Ticket Granting Ticket



That was authentication… what about authorization?

- Kerberos puts authorization function at the server itself
  - Idea is that it is easier to administer this information at the server
- TGS will give the client a ticket to talk to a server whether or not the client is authorized
- Server will reject ticket if client doesn't have proper authorization

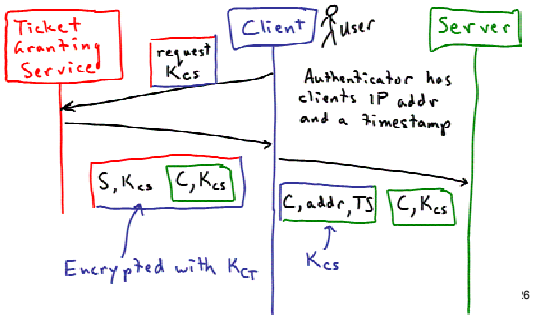
What about replay attack?



What about replay attack?

- This won't work if subsequent client/server session is encrypted
  - Because eavesdropper never sees Kcs
- But often client/server session is not encrypted, only authenticated

Client "authenticator"



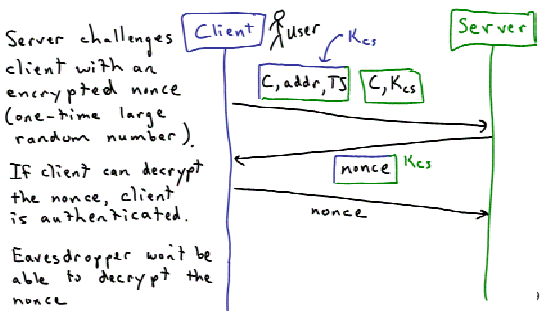
Client "authenticator"

- Client also sends an authenticator containing the client IP address and a timestamp
- The server only accepts the authenticator if from the right IP address and at the right time
  - Within a clock sync window of error, about five minutes
- To replay, attacker must replay from the same IP address within 5 minutes

Replay

- This is considered to be rather weak replay protection
- An attacker may be on the same machine as the user
  - Or may simply assign itself that IP address
- Kerberos Version 5 has an optional challenge/response

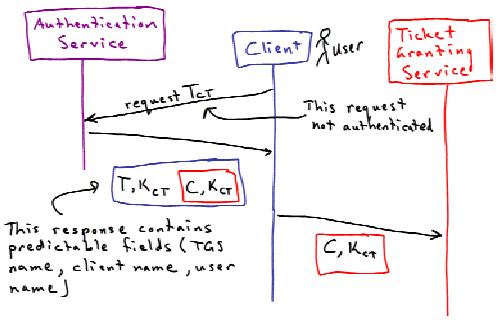
Challenge/Response



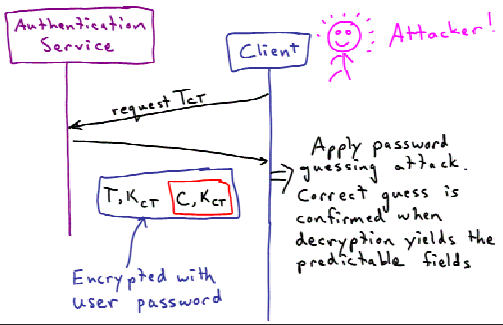
Challenge/Response

- Requires extra messages
  - And extra expensive crypto operations
- Requires temporary state stored at server
  - Though not a big deal

Password guessing attack



Password guessing attack



Password guessing attack

- Ultimately Kerberos relies on good user passwords
  - The usual thing:
    - no common words, no personal info (friends names, birthday, etc.), and no clever permutations of these
- Use of authenticated queries and Diffie-Hellman would make password guessing attack harder

Other Kerberos weaknesses

- Kerberos servers are a bottleneck
- Kerberos weak to denial of service attack
  - Take out the servers, and the whole network comes to a screeching halt!
- Use of public keys with Kerberos addresses these issues
  - Note: Kerberos originally didn't use public keys because of patent issues

Kerberos with public keys

- Replace Kerberos servers (authentication and ticket granting) with a Certificate Authority (CA)
  1. Client gets CA signed pub key of server
  2. Client sends server its cert (containing client pub key) and a random session key, signed by client priv key and encrypted by server pub key
  3. Server authenticates client, sends client a "ticket" (which can be used by Kerberos applications for backwards compatibility)