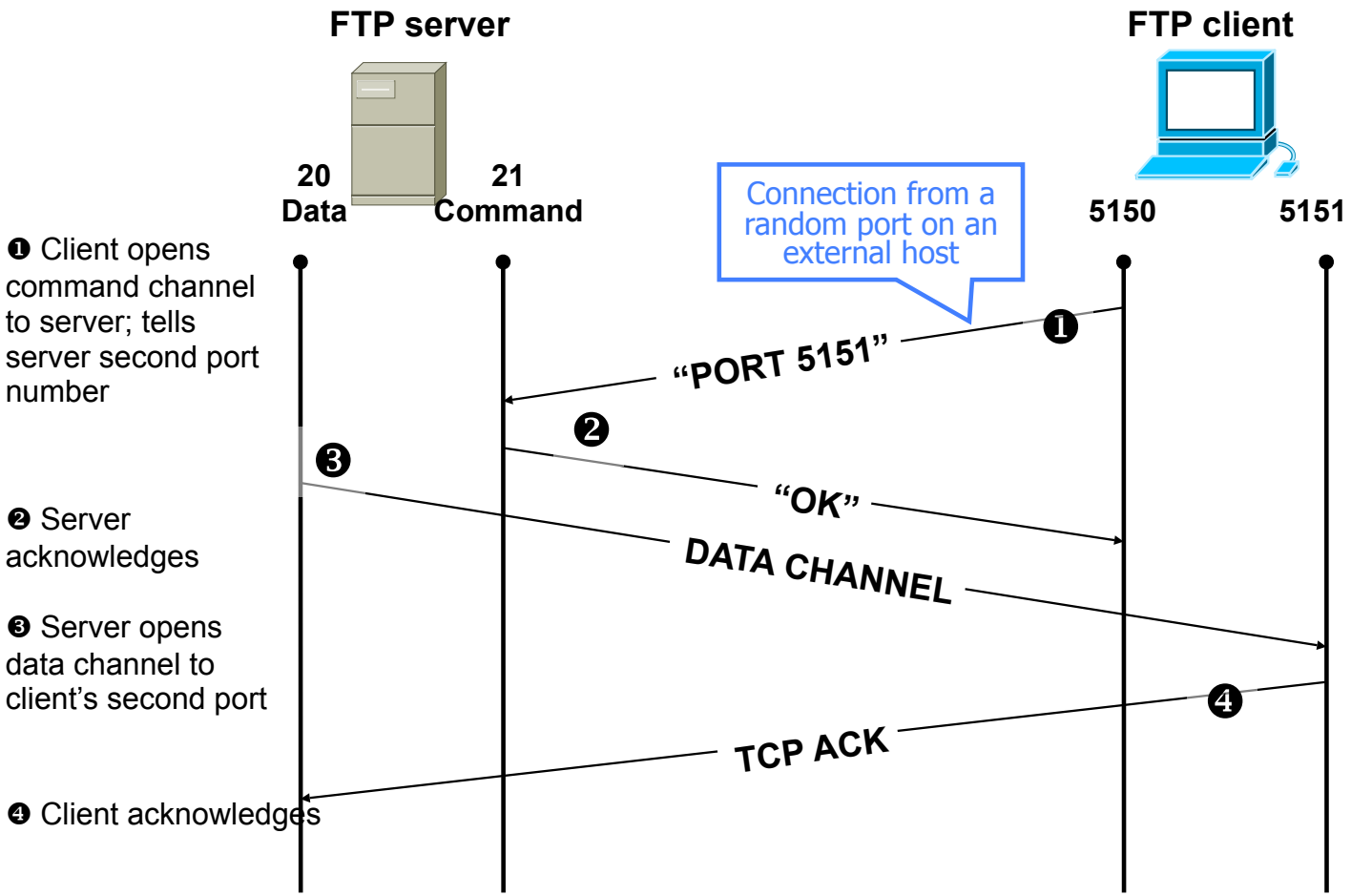
CSE 484 and CSE M 584 (Winter 2009)

Networks
Crypto -- Memory and Randomness
User Authentication

Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Example: FTP (borrowed from Wenke Lee)



slide

Session Filtering

- Decision is still made separately for each packet, but in the context of a connection
  - If new connection, then check against security policy
  - If existing connection, then look it up in the table and update the table, if necessary
    - Only allow incoming traffic to a high-numbered port if there is an established connection to that port
- Hard to filter stateless protocols (UDP) and ICMP
- Typical filter: deny everything that's not allowed
  - Must be careful filtering out service traffic such as ICMP
- Filters can be bypassed with IP tunneling

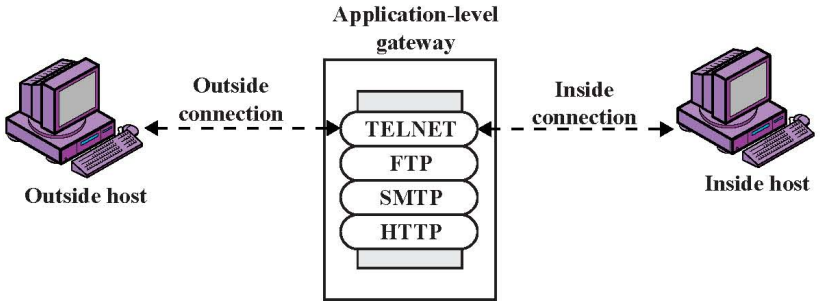
slide

Example: Connection State Table

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

slide

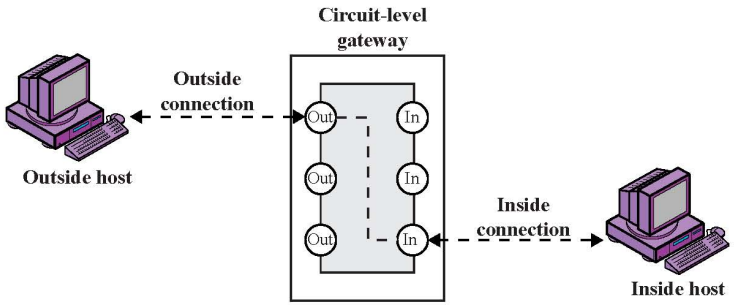
Application-Level Gateway



- Splices and relays two application-specific connections
  - Example: Web browser proxy
  - Daemon spawns proxy process when communication is detected
  - Big processing overhead, but can log and audit all activity
- Can support high-level user-to-gateway authentication
  - Log into the proxy server with your name and password
- Simpler filtering rules than for arbitrary TCP/IP traffic
- Each application requires implementing its own proxy

slide

Circuit-Level Gateway



- Splices two TCP connections, relays TCP segments
- Less control over data than application-level gateway
  - Does not examine the contents of TCP segment
- Client's TCP stack must be aware of the gateway
  - Client applications are often adapted to support SOCKS
- Often used when internal users are trusted
  - Application-level proxy on inbound connections, circuit-level proxy on outbound connections (lower overhead)

slide

Comparison

	Performance	Modify client application	Defends against fragm. attacks
Packet filter	Best	No	No
Session filter		No	Maybe
Circuit-level gateway		Yes (SOCKS)	Yes
Application-level gateway	Worst	Yes	Yes

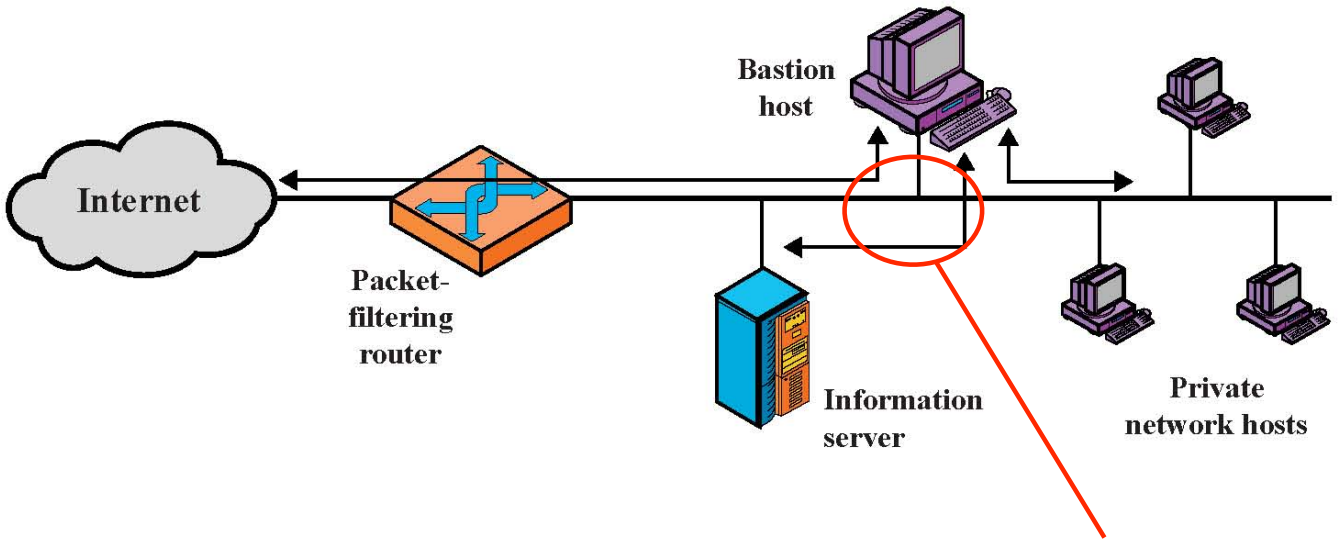
slide

Bastion Host

- Bastion host is a hardened system implementing application-level gateway behind packet filter
  - All non-essential services are turned off
  - Application-specific proxies for supported services
    - Each proxy supports only a subset of application's commands, is logged and audited, disk access restricted, runs as a non-privileged user in a separate directory (independent of others)
  - Support for user authentication
- All traffic flows through bastion host
  - Packet router allows external packets to enter only if their destination is bastion host, and internal packets to leave only if their origin is bastion host

slide

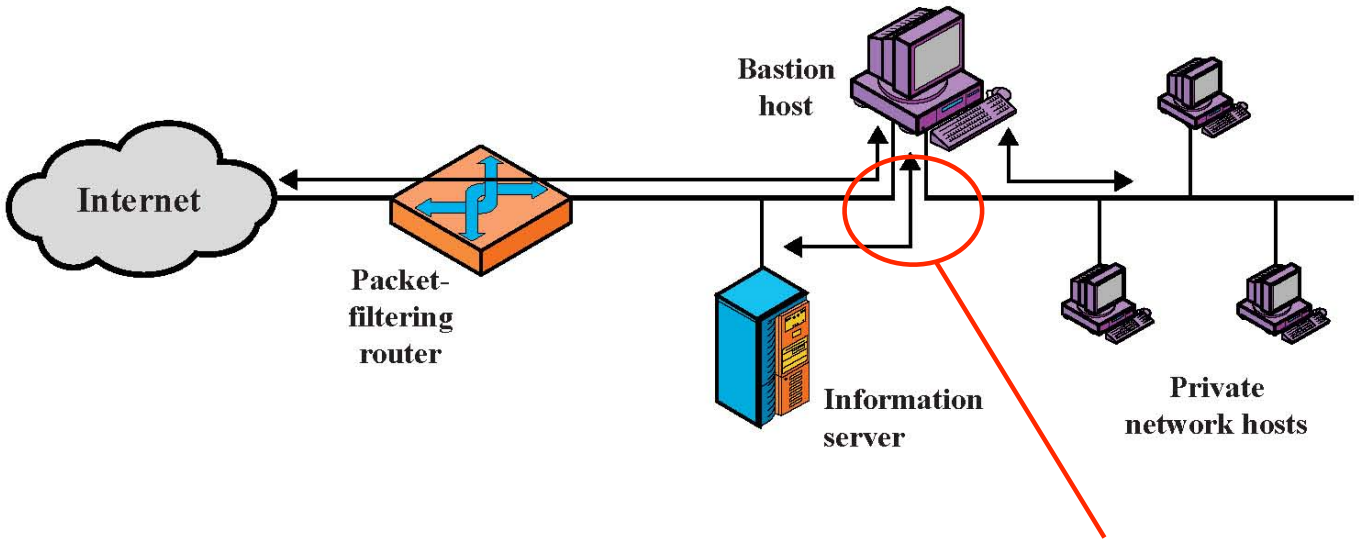
Single-Homed Bastion Host



If packet filter is compromised, traffic can flow to internal network

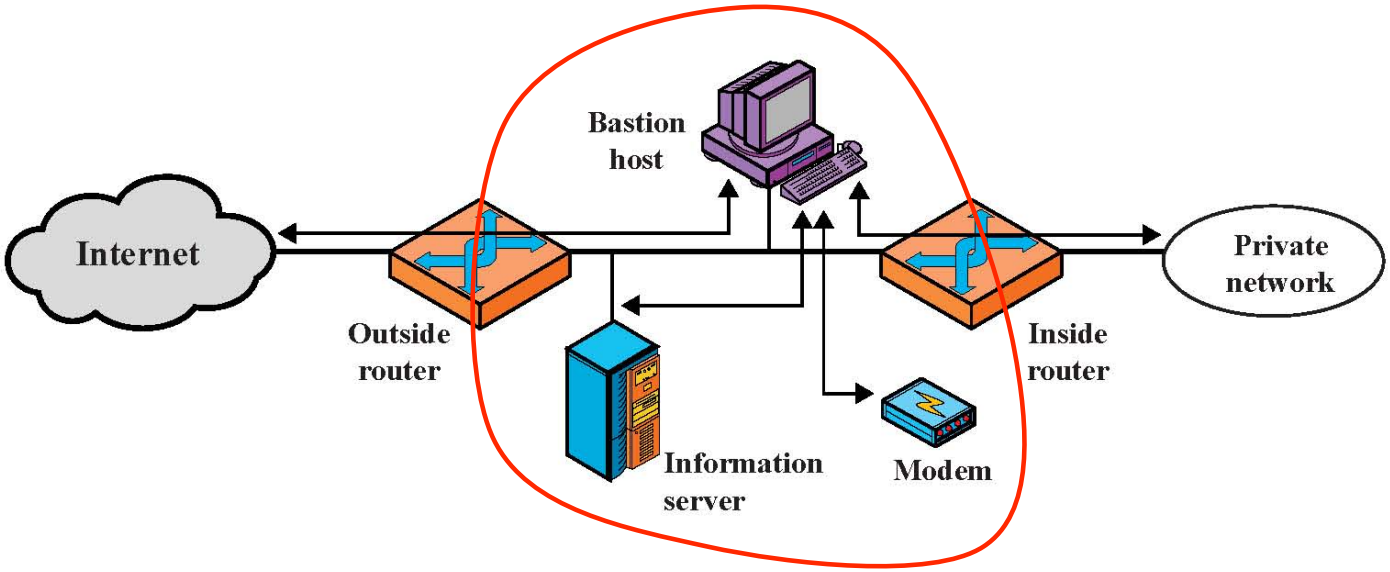
slide

Dual-Homed Bastion Host



slide

Screened Subnet



Only the screened subnet is visible
to the external network;
internal network is invisible

slide

Protecting Addresses and Routes

- Hide IP addresses of hosts on internal network
  - Only services that are intended to be accessed from outside need to reveal their IP addresses
  - Keep other addresses secret to make spoofing harder
- Use NAT (network address translation) to map addresses in packet headers to internal addresses
  - 1-to-1 or N-to-1 mapping
- Filter route announcements
  - No need to advertise routes to internal hosts
  - Prevent attacker from advertising that the shortest route to an internal host lies through him

slide

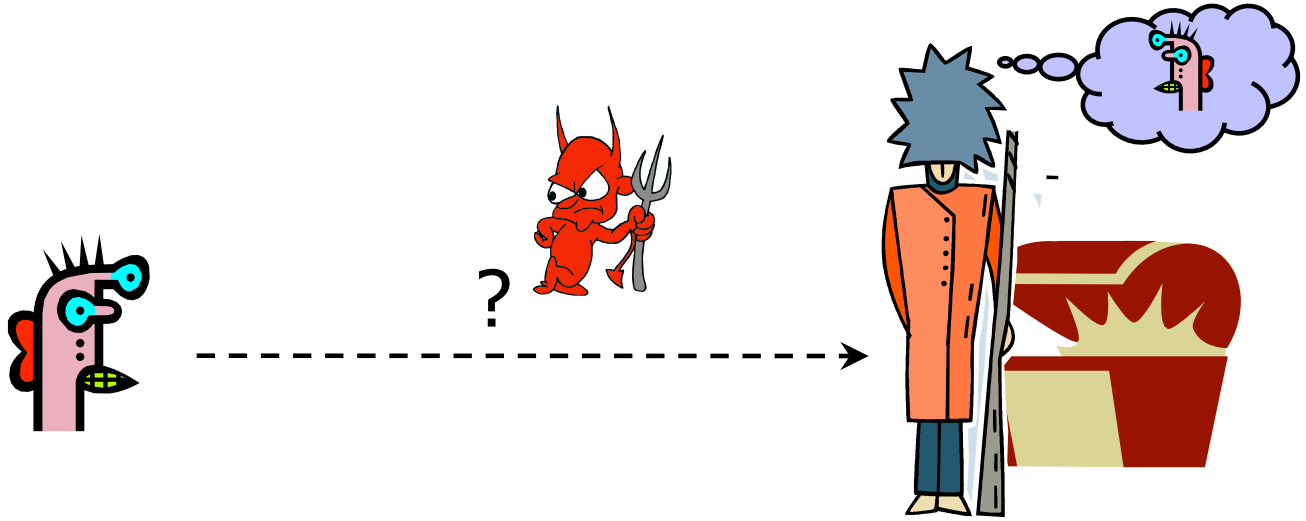
General Problems with Firewalls

- Interfere with networked applications
- Doesn't solve all the problems
  - Buggy software (think buffer overflow exploits)
  - Bad protocol design (think WEP in 802.11b)
- Generally don't prevent denial of service
- Don't prevent insider attacks
- Increasing complexity and potential for misconfiguration

slide

User Authentication

Basic Problem



How do you prove to someone that
you are who you claim to be?

Any system with access control must solve this problem

Many Ways to Prove Who You Are

- What you know
  - Passwords
  - Secret key
- Where you are
  - IP address
  - Physical location
- What you are
  - Biometrics
- What you have
  - Secure tokens
- All have advantages and disadvantages

Why Authenticate?

- To prevent an attacker from breaking into our account
  - Co-worker, family member, ...
- To prevent an attacker from breaking into any account on our system
  - Unix system
    - Break into single account, then exploit local vulnerability or mount a "stepping stones" attack
  - Calling cards
  - Building
- To prevent an attacker from breaking into any account on any system

Also Need

- Usability!
  - Remember password?
  - Have to bring physical object with us all the time?
- Denial of service
  - Stolen wallet
  - Try to authenticate as you until your account becomes locked
  - What about a military or other mission critical scenario
    - Lock all accounts - system unusable

Password-Based Authentication

- User has a secret password.

  System checks it to authenticate the user.
  - May be vulnerable to eavesdropping when password is communicated from user to system
- How is the password stored?
- How does the system check the password?
- How easy is it to remember the password?
- How easy is it to guess the password?
  - Easy-to-remember passwords tend to be easy to guess
  - Password file is difficult to keep secret

Common usage modes

Amazon = t0p53cr37

UWNetID = f0084r#1

Bank = a2z@m0$;

Image from http://www.interactivetools.com/staff/dave/damons_office/

Common usage modes

- Write down passwords
- Share passwords with others
- Use a single password across multiple sites
  - Amazon.com and Bank of America?
  - UW CSE machines and MySpace?
- Use easy to remember passwords
  - Favorite <something>?
  - Name + <number>?
- Other "authentication" questions
  - Mother's maiden name?

Some anecdotes [Dhamija and Perrig]

- Users taught how to make secure passwords, but chose not to do so

- Reasons:
  - Awkward or difficult
  - No accountability
  - Did not feel that it was important

Social Engineering

- "Hi, I'm the CEO's assistant. I need you to reset his password right away. He's stuck in an airport and can't log in! He lost the paper that he wrote the password on.
- "What do you mean you can't do it!? Do you really want me to tell him that you're preventing him from closing this major deal?
- "Great! That's really helpful. You have no idea how important this is. Please set the password to ABCDEFG. He'll reset it again himself right away.
- "Thanks!"

University of Sydney Study [Greening '96]

- 336 CS students emailed message asking them to supply their password
  - Pretext: in order to "validate" the password database after a suspected break-in
- 138 students returned their password
- 30 returned invalid password
- 200 changed their password
- (Not disjoint)

- Still, 138 is a lot!

Awkward

- How many times do you have to enter your password before it actually works?
  - Sometimes quite a few for me! (Unless I type extra slowly.)
- Interrupts normal activity
  - Do you lock your computer when you leave for 5 minutes?
  - Do you have to enter a password when your computer first boots? (Sometimes it's an option.)
- And memorability is an issue!

Memorability [Anderson]

- Hard to remember many PINs and passwords
- One bank had this idea
  - If pin is 2256, write your favorite 4-letter word in this grid
  - Then put random letters everywhere else

1	2	3	4	5	6	7	8	9	0
	b								
	l								
				u					
					e				

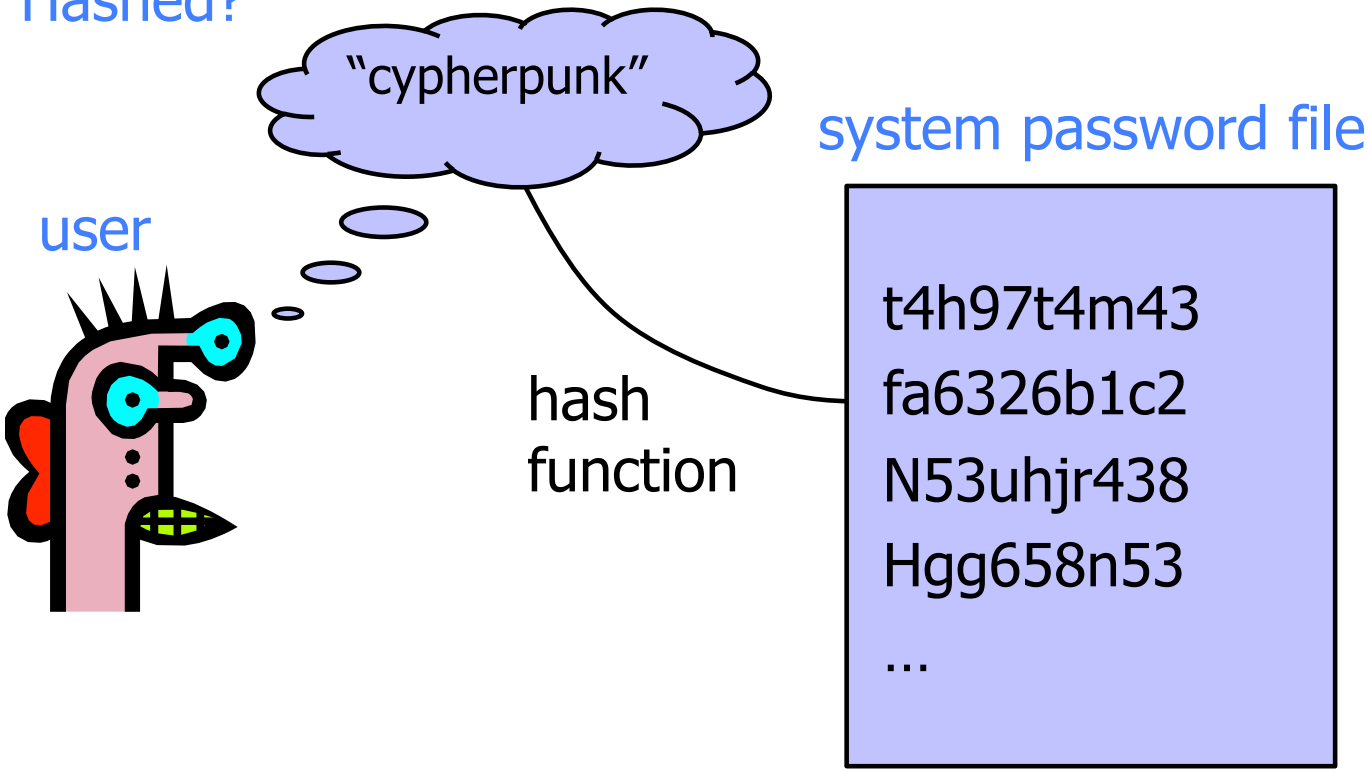
Memorability [Anderson]

- Problem!
- Normally 10000 choices for the PIN --- hard to guess on the first try
- Now, only a few dozen possible English words --- easy to guess on first try!

1	2	3	4	5	6	7	8	9	0
	b								
	l								
				u					
					e				

UNIX-Style Passwords

- How should we store passwords on a server?
  - In cleartext?
  - Encrypted?
  - Hashed?



Password Hashing

- Instead of user password, store H(password)
- When user enters password, compute its hash and compare with entry in password file
  - System does not store actual passwords!
  - System itself can't easily go from hash to password
    - Which would be possible if the passwords were encrypted
- Hash function H must have some properties
  - One-way: given H(password), hard to find password
    - No known algorithm better than trial and error
    - It should even be hard to find any pair p1,p2 s.t. H(p1)=H(p2)

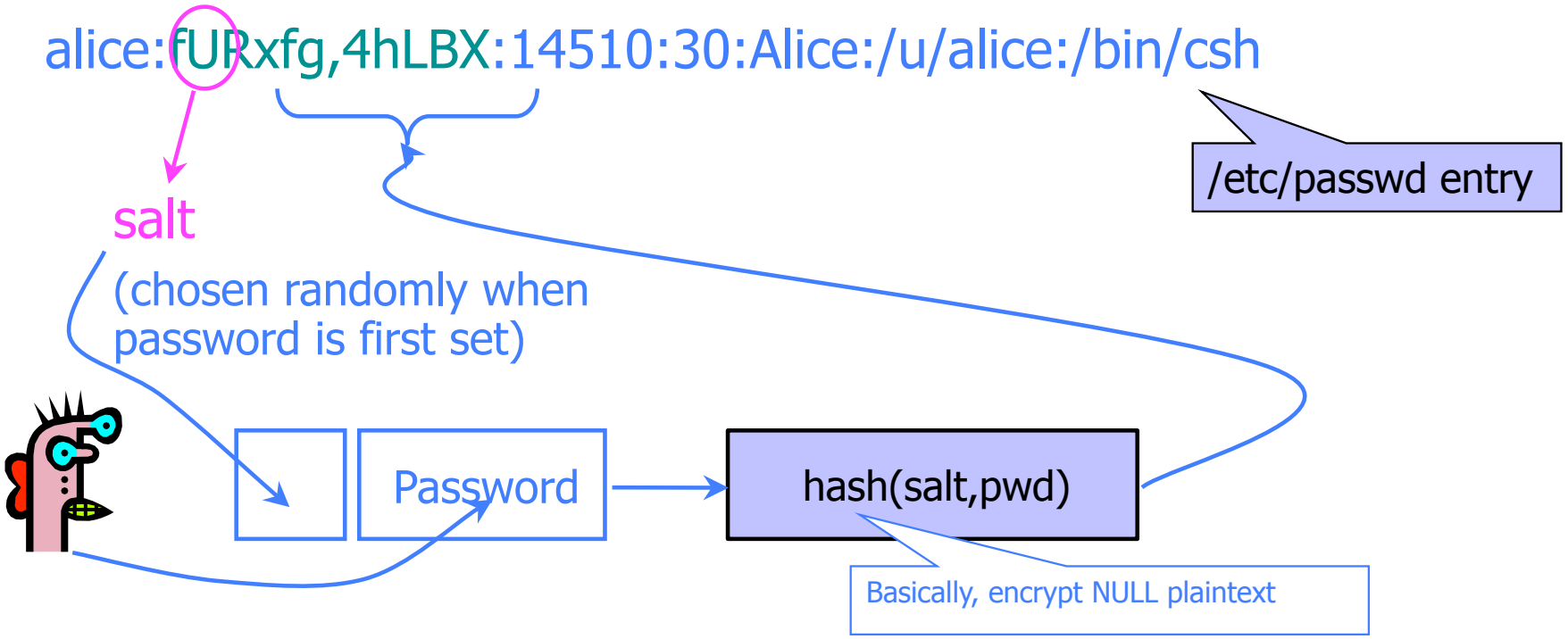
UNIX Password System

- Uses DES encryption as if it were a hash function
  - Encrypt NULL string using password as the key
    - Truncates passwords to 8 characters!
  - Artificial slowdown: run DES 25 times
    - Why 25 times? Slowdowns like these are important in practice!
  - ("Don't use DES like this at home.")
  - Can instruct modern UNIXes to use MD5 hash function
- Problem: passwords are not truly random
  - With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are $94^8 \approx 6$ quadrillion possible 8-character passwords (around 2^{52})
  - Humans like to use dictionary words, human and pet names ≈ 1 million common passwords

Dictionary Attack

- Password file /etc/passwd is world-readable
  - Contains user IDs and group IDs which are used by many system programs
- Dictionary attack is possible because many passwords come from a small dictionary
  - Attacker can compute H(word) for every word in the dictionary and see if the result is in the password file
  - With 1,000,000-word dictionary and assuming 10 guesses per second, brute-force online attack takes 50,000 seconds (14 hours) on average
    - This is very conservative. Offline attack is much faster!
    - As described, could just create dictionary of word-->H(word) once!!

Salt



- Users with the same password have <u>different</u> entries in the password file
- Dictionary attack is still possible!

Advantages of Salting

- Without salt, attacker can pre-compute hashes of all dictionary words once for all password entries
  - Same hash function on all UNIX machines
  - Identical passwords hash to identical values; one table of hash values can be used for all password files
- With salt, attacker must compute hashes of all dictionary words once for each password entry
  - With 12-bit random salt, same password can hash to 2^{12} different hash values
  - Attacker must try all dictionary words for each salt value in the password file
- Pepper: Secret salt (not stored in password file)

Other Password Issues

- Keystroke loggers
  - Hardware
  - Software / Spyware
- Shoulder surfing
  - It's happened to me!
- Online vs offline attacks
  - Online: slower, easier to respond
- Multi-site authentication
  - Share passwords?