

A blue square is located in the top-left corner. A blue L-shaped line extends from the bottom-right corner of the square, consisting of a vertical line going down and a horizontal line going right across the top of the slide.

## *Protection and Security*

How to be a paranoid  
or just think like one

# The Problem

- ◆ Types of misuse
  - Accidental
  - Intentional
- ◆ Protection and security objective
  - Protect against/prevent misuse
- ◆ Three key components:
  - Authentication: Verify user identity
  - Authorization: Assign rights to users
  - Enforcement: Enforce access rights
- ◆ Today's class
  - An overview of authentication, authorization, and enforcement techniques
    - ❖ Isolated/local systems
    - ❖ Distributed systems

# *What are your security goals?*

- ◆ Authentication
  - User is who s/he says they are.
  - Example: Public key.
- ◆ Integrity
  - Adversary can not change contents of message
  - But not necessarily private (public key)
  - Example: secure checksum
- ◆ Privacy (confidentiality)
  - Adversary can not read your message
  - If adversary eventually breaks your system can they decode all stored communication?
  - Example: Anonymous remailer (how to reply?)
- ◆ Authorization, repudiation (or non-repudiation), forward security (crack now, not crack future), backward security (crack now, not cracked past)

# *What About Security in Distributed Systems?*

- ◆ Three challenges
  - Authentication
    - ❖ Verify user identity
  - Integrity
    - ❖ Verify that the communication has not been tempered with
  - Privacy
    - ❖ Protect access to communication across hosts
- ◆ Solution: Encryption
  - Achieves all these goals
  - Transform data that can easily reversed given the correct key (and hard to reverse without the key)
- ◆ Two common approaches
  - Private key encryption
  - Public key encryption
- ◆ Cryptographic hash
  - Hash is a fixed sized byte string which represents arbitrary length data. Hard to find two messages with same hash.

# *Authentication*

- ◆ Objective: Verify user identity
- ◆ Common approach:
  - Passwords: shared secret between two parties
  - Present password to verify identity
- ◆ Challenges
  1. How can the system maintain a copy of passwords?
    - Encryption: Transformation that is difficult to reverse without right key
    - Example: Unix /etc/passwd file contains encrypted passwords
    - When you type password, system encrypts it and then compared encrypted versions

# *Authentication (Cont'd.)*

## ◆ Challenges (cont'd.)

### 2. Passwords must be long and obscure

- Paradox:
  - ❖ Short passwords are easy to crack
  - ❖ Long passwords - users write down to remember → vulnerable
- Original Unix:
  - ❖ 5 letter, lower case password
  - ❖ Exhaustive search requires  $26^5 = 12$  million comparisons
  - ❖ Today: < 1us to compare a password → 12 seconds to crack a password!!
- Choice of passwords
  - ❖ English words: Shakespeare's vocabulary: 30K words
  - ❖ All English words, fictional characters, place names, words reversed, ... still too few words
  - ❖ (Partial) solution: More complex passwords
    - At least 8 characters long, with upper/lower case, numbers, and special characters

## *Are Long Passwords Sufficient?*

- ◆ Example: Tenex system (1970s - BBN)

- Considered to be a very secure system
- Code for password check:

```
For (i=0, i<8, i++) {  
    if (userPasswd[i] != realPasswd[i])  
        Report Error;  
}
```

- Looks innocuous - need to try  $256^8$  (=  $1.8E+19$ ) combinations to crack a password
- Is this good enough??

No!!!

## *Are Long Passwords Sufficient? (Cont'd.)*

- ◆ Problem:
  - Can exploit the interaction with virtual memory to crack passwords!
- ◆ Key idea:
  - Force page faults at carefully designed times to reveal password
  - Approach
    - ❖ Arrange first character in string to be the last character in a page
    - ❖ Arrange that the page with the first character is in memory
    - ❖ Rest is on disk (e.g., a|bcdefgh)
    - ❖ Check how long does a password check take?
      - ◆ If fast → first character is wrong
      - ◆ If slow → first character is right → page fault → one of the later character is wrong
    - ❖ Try all first characters until the password check takes long
    - ❖ Repeat with two characters in memory, ...
  - Number of checks required =  $256 * 8 = 2048$  !!
- ◆ Fix:
  - Don't report error until you have checked all characters!
  - But, how do you figure this out in advance??
  - Timing bugs are REALLY hard to avoid

# *Emerging alternatives/enhancements to Passwords*

- ◆ Two-factor authentication
  - Password and some other channel, e.g., physical device with key that changes every minute
  - <http://www.schneier.com/essay-083.html>
  - What about a fake bank web site? (man in the middle)
- ◆ Biometrics
  - Fingerprint, retinal scan
  - What if I have a cut? What if someone wants my finger?
- ◆ Facial recognition

# Authorization

- ◆ Objective:
  - Specify access rights: who can do what?
- ◆ Access control: formalize all permissions in the system

	File1	File2	File3	...
User A	RW	R	--	...
User B	--	RW	RW	..
User C	RW	RW	RW	...

- ◆ Problem:
  - Potentially huge number of users, objects → impractical
- ◆ Approaches:
  - Access control lists
    - ❖ Store permissions for all users with objects
    - ❖ Unix approach: three categories of access rights (owner, group, world)
    - ❖ Recent systems: more flexible with respect to group creation
  - Capability lists (a capability is like a ticket)
    - ❖ Each process stores information about objects it has permission to touch
    - ❖ Processes present capability to objects to access (e.g., file descriptor)
    - ❖ Lots of capability-based systems built in the past → idea out of favor today

# Enforcement

- ◆ Objectives:
  - Check password, enforce access control
- ◆ General approach
  - Separation between "user" mode and "privileged" mode
- ◆ In Unix:
  - When you login, you authenticate to the system by providing password
  - Once authenticated - create a shell for specific userID
  - All system calls pass userID to the kernel
  - Kernel checks and enforces authorization constraints
- ◆ Paradox
  - Any bug in the enforcer → you are hosed!
  - Make enforcer as small and simple as possible
    - ❖ Called the trusted computing base.
    - ❖ Easier to debug, but simple-minded protection (run a lot of services in privileged mode)
  - Support complex protection schemes
    - ❖ Hard to get it right!

# Private Key (Symmetric Key) Encryption

- ◆ Basic idea:
  - $(\text{Plain text})^K \rightarrow \text{cipher text}$
  - $(\text{Cipher text})^K \rightarrow \text{plain text}$
  - As long as key  $K$  stays secret, we get authentication and secrecy
- ◆ Infrastructure: Authentication server (example: kerberos)
  - Maintains a list of passwords; provides a key for two parties to communicate
- ◆ Basic steps:
  - $A \rightarrow S$  (Hi! I would like a key for AB)
  - $S \rightarrow A$  (Use  $K_{ab}$  (This is A! Use  $K_{ab}$ ) $^{K_{sb}}$ ) $^{K_{sa}}$
  - $A \rightarrow B$  (This is A! Use  $K_{ab}$ ) $^{K_{sb}}$
  - Master keys ( $K_{sa}$  and  $K_{sb}$ ) distributed out-of-band and stored securely at clients
- ◆ Refinements
  - Generate temporary keys to communicate between clients and authentication server

# Public Key Encryption

- ◆ Basic idea:
  - Separate authentication from secrecy
  - Each key is a pair: K-public and K-private
  - $(\text{Plain text})^{K\text{-private}} \rightarrow \text{cipher text}$
  - $(\text{Cipher text})^{K\text{-public}} \rightarrow \text{plain text}$
  - K-private is kept a secret; K-public is distributed
- ◆ Examples:
  - $(\text{I'm Emmett})^{K\text{-private}}$ 
    - ❖ Everyone can read it, but only I can send it (authentication)
  - $(\text{Hi, Emmett})^{K\text{-public}}$ 
    - ❖ Anyone can send it but only I can read it (secrecy)
- ◆ Two-party communication
  - $A \rightarrow B (\text{I'm A (use } K_{ab})^{K\text{-private}_A})^{K\text{-public}_B}$
  - No need for an authentication server
  - Question: how do you trust the "public key" server?
    - ❖ Trusted server:  $(K\text{-public}_A)^{K\text{-private}_S}$

## *Summary*

---

- ◆ Security in distributed system is essential
- ◆ .. And is hard to achieve!
- ◆ Lots of work ...
  - Take courses in security