Authentication
CS 136
Computer Security
Peter Reiher
January 22, 2008

Outline

- Introduction
- Basic authentication mechanisms
- Authentication on a single machine
- Authentication across a network

Introduction

- Much of security is based on good access control
- Access control only works if you have good authentication
- What is authentication?

Authentication

- Determining the identity of some entity
  - Process
  - Machine
  - Human user
- Requires notion of identity
- And some degree of proof of identity

Authentication Vs. Authorization

- *Authentication* is determining who you are
- *Authorization* is determining what someone is allowed to do
- Can't authorize properly without authentication
- Purpose of authentication is usually to make authorization decisions

Proving Identity in the Physical World

- Most frequently done by physical recognition
  - I recognize your face, your voice, your body
- What about identifying those we don't already know?

Other Physical World Methods of Identification

- Identification by recommendation
  - You introduce me to someone
- Identification by credentials
  - You show me your driver's license
- Identification by knowledge
  - You tell me something only you know
- Identification by location
  - You're behind the counter at the DMV
- These all have cyber analogs

Differences in Cyber Identification

- Usually the identifying entity isn't human
- Often the identified entity isn't human, either
- Often no physical presence required
- Often no later rechecks of identity

Identifying With a Computer

- Not as smart as a human
  - Steps to prove identity must be well defined
- Can't do certain things as well
  - E.g., face recognition
- But lightning fast on computations and less prone to simple errors
  - Mathematical methods are acceptable

Identifying Computers and Programs

- No physical characteristics
  - Faces, fingerprints, voices, etc.
- Generally easy to duplicate programs
- Not smart enough to be flexible
  - Must use methods they will understand
- Again, good at computations

Physical Presence Optional

- Often authentication required over a network or cable
- Even if the party to be identified is human
- So authentication mechanism must work in face of network characteristics
  - E.g., active wiretapping

Identity Might Not Be Rechecked

- Human beings can make identification mistakes
- But they often recover from them
  - Often quite easily
- Based on observing behavior that suggests identification was wrong
- Computers and programs rarely have that capability
  - If they identify something, they believe it

Authentication Mechanisms

- Something you know
  - E.g., passwords
- Something you have
  - E.g., smart cards or tokens
- Something you are
  - Biometrics
- Somewhere you are
  - Usually identifying a role

Passwords

- Authentication by what you know
- One of the oldest and most commonly used security mechanisms
- Authenticate the user by requiring him to produce a secret
  - Usually known only to him and to the authenticator

Problems With Passwords

- They have to be unguessable
  - Yet easy for people to remember
- If networks connect terminals to computers, susceptible to password sniffers
- Unless fairly long, brute force attacks often work on them

Proper Use of Passwords

- Passwords should be sufficiently long
- Passwords should contain non-alphabetic characters
- Passwords should be unguessable
- Passwords should be changed often
- Passwords should never be written down
- Passwords should never be shared

Passwords and Single Sign-On

- Many systems ask for password once
  - Resulting authentication lasts for an entire "session"
- Unless other mechanisms in place, complete mediation definitely not achieved
- Trading security for convenience

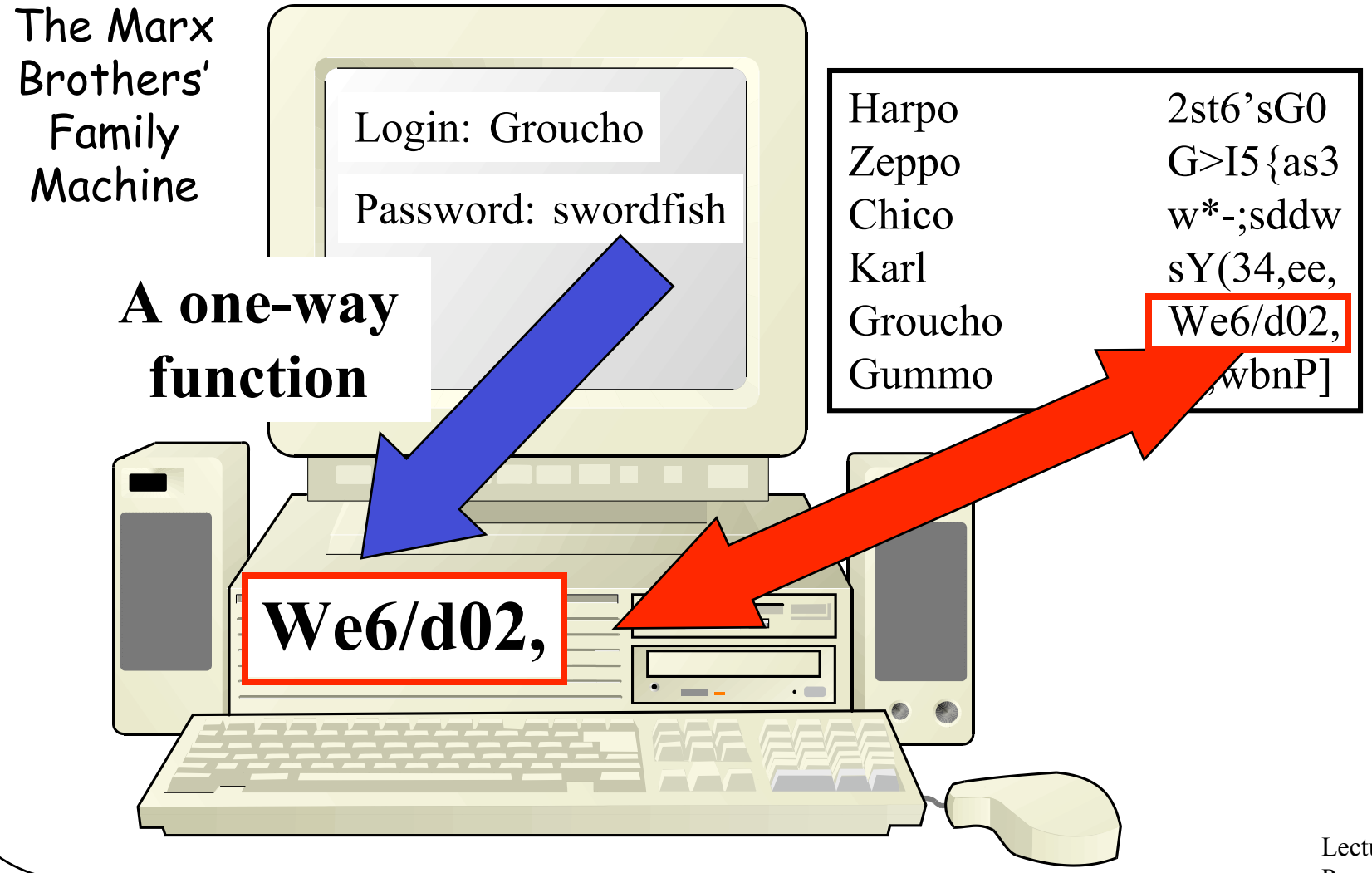
Handling Passwords

- The OS must be able to check passwords when users log in
- So must the OS store passwords?
- Not really
  - It can store an encrypted version
- Encrypt the offered password
  - Using a *one-way function*
- And compare it to the stored version

One Way Functions

- Functions that convert data A into data B
- But it's hard to convert data B back into data A
- Often done as a particular type of cryptographic operation
  - E.g., cryptographic hashing
- Depending on particular use, simple hashing might be enough
  - Discussed in more detail in crypto lectures

Standard Password Handling



Is Encrypting the Password File Enough?

- What if an attacker gets a copy of your password file?
- No problem, the passwords are encrypted
  - Right?
- Yes, but . . .

Dictionary Attacks on an Encrypted Password File

Harpo	2st6'sG0
Zeppo	G>I5{as3
Chico	sY(34,ee
Karl	
Groucho	
Gummo	3(;wbnP]

Now you can hack the Communist Manifesto!

sY(34,ee

Rats!!!!

Dictionaries

- Real dictionary attacks don't use Webster's
- Dictionary based on probability of words being used as passwords
- Partly set up as procedures
  - E.g., try user name backwards
- Checks common names, proper nouns, etc. early
- Tend to evolve to match user trends

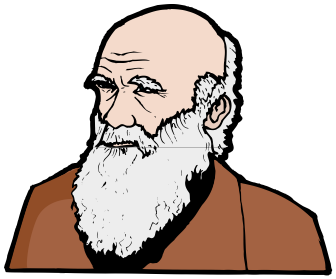
A Serious Issue

- All Linux machines use the same one-way function to encrypt passwords
- If someone runs the entire dictionary through that function,
  - Will they have a complete list of all encrypted dictionary passwords?

Illustrating the Problem

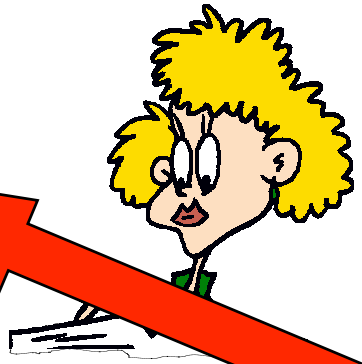
Karl Marx

^*eP6la-



Charles Darwin

^*eP6la-



aardvark
aardwolf

340ja
K[ds
sY(34
e

beard

^*eP61a-

The Real Problem

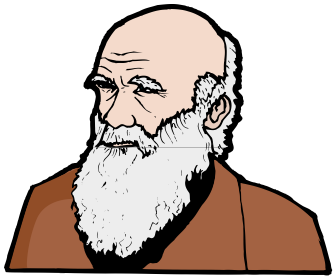
- Not that Darwin and Marx chose the same password
- But that anyone who chose that password got the same encrypted result
- So the attacker need only encrypt every possible password once
- And then she has a complete dictionary usable against anyone

Salted Passwords

- Combine the plaintext password with a random number
  - Then run it through the one-way function
- The random number need not be secret
- It just has to be different for different users

Did It Fix Our Problem?

Karl Marx



Charles Darwin

D0Cls6&

aardvark
aardwolf

340jafg;
K[ds+3a,
sY(34,ee

beard ^*eP61a-

)#4,doa8

What Is This Salt, Really?

- An integer that is combined with the password before hashing
- How will you be able to check passwords by hashing them, then?
- By storing the salt integer with the password
  - Generally in plaintext
- Why is it OK (or OK-ish) to leave this important information in plaintext?

Password Management

- Protecting the password file
- Forgotten passwords
- Generating new passwords
- Password transport

Protecting the Password File

- So it's OK to leave the encrypted version of the password file around?
- No, it isn't
- Why make it easy for attackers?
- Dictionary attacks against single accounts can still work
- Generally, don't give access to the encrypted file, either

Other Issues for Proper Handling of Users' Passwords

- Sites should store unencrypted passwords for as little time as possible
  - Partly issue of how they store the file
  - Partly issue of good programming
- Sites should take care not to leave passwords in temporary files or other places
- Should not be possible to print or save someone's unencrypted password
- Use encrypted network transport for passwords
- If your server is compromised, all of this might not help

Handling Forgotten Passwords

- Users frequently forget passwords
- How should your site deal with it?
- Bad idea:
  - Store plaintext passwords and send them on request
- Better idea:
  - Generate new passwords when old ones forgotten

Generating New Passwords

- Easy enough to generate a random one
- But you need to get it to the user
- If attacker intercepts it, authentication security compromised
- How do you get it to the user?

Transporting New Passwords

- Engineering solution is usually to send it in email
  - To address user registered with you earlier
- Often fine for practical purposes
- But there are very serious vulnerabilities

User Issues With Passwords

- Password proliferation
- Choosing passwords
- Password lifespan

Password Proliferation

- Practically every web site you visit wants you to enter a password
- Should you use the same password for all of them?
- Or a different password for each?

Using the Same Password

+ Easier to remember
- Much less secure
  - One password guesser gets all your authentication info
  - Do you trust all the sites you visit equally?

Using Different Passwords

+ Much more secure
- But how many passwords can you actually remember?
- And you might "solve" this problem by choosing crummy passwords

Other Options

- Use a few passwords
  - Maybe classified by type of site or degree of trust
- Write down your passwords
  - Several disadvantages
  - Could write down hints, instead
- Password vaults

Password Vaults

- Programs to store passwords for you
  - Typically on your own machine
  - Indexed by site, usually
- Bad ones store plaintext versions
- Good ones keep it all encrypted
  - Using a single password
- When you need a specific password, it's pulled out of the vault
- For encrypted versions, issues similar to single sign-on

Choosing Passwords

- Typically a compromise between:
  - Sufficient security
  - Remembering it
- Major issues:
  - Length
  - Complexity

How Long Should Passwords Be?

- Generally a function of how easy it is for attackers to attack them
- Changes as speed of processors increase
- Nowadays, 15 character password are pretty safe
  - If they aren't guessable . . .

Complexity vs. Length

- Should the password be long?
- Or is it enough to include numbers, special characters, different cases, etc.?
- Basic formula for password guessing is X^L
  - X is number of possible symbols
  - L is number of symbols in password
  - You can make passwords better by increasing either

Increasing Length

- Takes longer to type
  - More prone to typos
- You have to remember a longer password
- Can be made easier by mnemonic tricks
  - E.g., take the first letter of each word of 15 word phrase

Increasing Complexity

- Throwing in symbols, numbers, upper case letters
- Only gets full effect if you (might) use all possible symbols
  - In all possible positions in passwords
- Most typically, only occur at beginning or end
- Most typically, only one of each
- Thus, typical use doesn't gain full advantage

Password Lifespan

- How long should you use a given password?
- Ideally, change it frequently
- Practically, will you remember the new one?
- Is a good, old password worse than a bad, new one?

Issues for Password Lifetimes

- How good is the password?
- How many sites do you use the password for?
  - How confident are you that they handle it properly?
- How bad will it be if your password is divulged?

Challenge/Response Authentication

- Authentication by what questions you can answer correctly
  - Again, by what you know
- The system asks the user to provide some information
- If it's provided correctly, the user is authenticated

Differences From Passwords

- Challenge/response systems ask for different information every time
- Or at least the questions come from a large set
- Best security achieved by requiring what amounts to encryption of the challenge
  - But that requires special hardware
  - Essentially, a smart card

Problems With Authentication Through Challenge/Response

- Either the question is too hard to answer without special hardware
- Or the question is too easy for intruders to spoof the answer
- Still, commonly used in real-world situations
  - E.g., authenticating you by asking your childhood pet's name

A Short Digression on "Security Questions"

- Common in web sites
- If you forget your password, answer a "security question"
- Answering that properly gets you access
- Which means knowing the security question's answer is as good as knowing the password
- How secure are these "security questions?"
- How could the concept be improved?

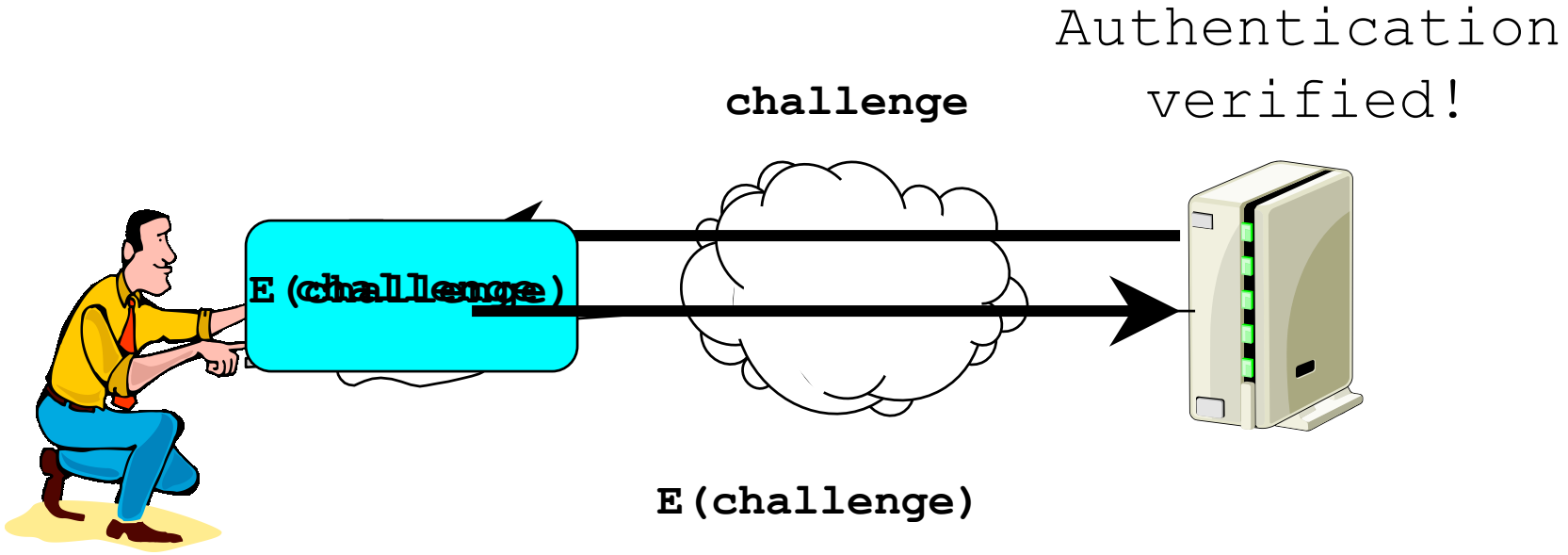
Identification Devices

- Authentication by what you have
- A smart card or other hardware device that is readable by the computer
- Authenticate by providing the device to the computer

Simple Use of Authentication Tokens

- If you have the token, you are identified
- Generally requires connecting the authentication device to computer
  - Unless done via wireless
- Weak, because it's subject to theft and spoofing

Authentication With Smart Cards



How can the server be sure of the remote user's identity?

Some Details on Smart Cards

- Cryptography performed only on smart card
  - So compromised client machine can't steal keys
- Often user must enter password to activate card
  - Should it be entered to the card or the computer?

Problems With Identification Devices

- If lost or stolen, you can't authenticate yourself
  - And maybe someone else can
  - Often combined with passwords to avoid this problem
- Unless cleverly done, susceptible to sniffing attacks
- Requires special hardware

Authentication Through Biometrics

- Authentication based on who you are
- Things like fingerprints, voice patterns, retinal patterns, etc.
- To authenticate to the system, allow system to measure the appropriate physical characteristics
- Biometric converted to binary and compared to stored values
  - With some level of match required

Problems With Biometric Authentication

- Requires very special hardware
  - Possibly excepting systems that examine typing patterns
- May not be as foolproof as you think
- Many physical characteristics vary too much for practical use
- Generally not helpful for authenticating programs or roles
- What happens when it's cracked?
  - You only have two retinas, after all

When Do Biometrics (Maybe) Work Well?

- When you use them for authentication
  - Carefully obtain clean readings from legitimate users
  - Compare those to attempts to authenticate
- When biometric readers are themselves secure
- In conjunction with other authentication

When Do Biometrics (Definitely) Work Poorly?

- Finding "needles in haystacks"
  - Face recognition of terrorists in airports
- When working off low-quality readings
- When the biometric reader is easy to bypass or spoof
  - Anything across a network is suspect
- When the biometric is "noisy"
  - Too many false negatives

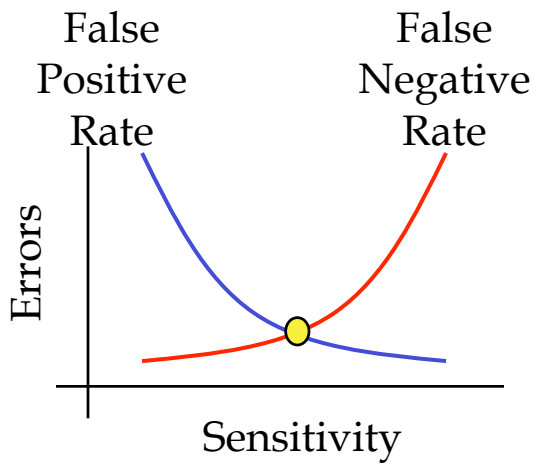
Characterizing Biometric Accuracy

How many false positives?

Match made when it shouldn't have been

Versus how many false negatives?

Match not made when it should have been



The Crossover Error Rate (CER)

Generally, the higher the CER is, the better the system

But sometimes one rate more important than the other

Some Typical Crossover Error Rates

Technology	Rate
Retinal Scan	1:10,000,000+
Iris Scan	1:131,000
Fingerprints	1:500
Facial Recognition	1:500
Hand Geometry	1:500
Signature Dynamics	1:50
Voice Dynamics	1:50

Data as of 2002

- Things can improve a lot in this area over time
- Also depends on how you use them
- And on what's important to your use

A Biometric Cautionary Tale

- A researcher in Japan went out and bought some supplies from a hobby store
- He used them to create gummy fingers
  - With gummy fingerprints
- With very modest tinkering, his gummy fingers fooled all commercial fingerprint readers
- Maybe today's readers are better
  - Maybe not . . .

Authentication by Where You Are

- Sometimes useful in ubiquitous computing
- The issue is whether the message in question is coming from the machine that's nearby
- Less important who owns that machine
- Requires sufficient proof of physical location
- And ability to tie a device at that location to its messages
- Sometimes used in conjunction with other authentication methods
  - E.g., the door opens only if an authorized user is right outside it

Authentication on Physical Machines

- Generally controlled by the operating system
- Sometimes at application level
- At OS level, most frequently done at login time
- How does the OS authenticate later requests?

Process Authentication

- Memory protection is based on process identity
  - Only the owning process can name its own virtual memory pages
- Virtual memory completely in OS control
  - Pretty easy to ensure that processes can't fake identities
- OS and virtual memory security discussed in more detail later

How the OS Authenticates Processes

- System calls are issued by a particular process
- The OS securely ties a process control block to the process
  - Not under user control
- Thus, the ID in the process control block can be trusted

How Do Processes Originally Obtain Access Permission?

- Most OS resources need access control based on user identity or role
  - Other than virtual memory pages and other transient resources
- How does a process get properly tagged with its owning user or role?
- Security is worthless if OS carefully controls access on a bogus user ID

Users and Roles

- In most systems, OS assigns each potential user an ID
- More sophisticated systems recognize that the same user works in different *roles*
  - Effectively, each role requires its own ID
  - And secure methods of setting roles

Securely Identifying Users and Roles

- Passwords
- Identification devices
- Challenge/response systems
- Physical verification of the user

Authenticating Across the Network

- What new challenges does this add?
- You don't know what's at the other end of the wire
- So, when does that cause a problem?
- And how can you solve it?