

Authentication

Computer Security

Peter Reiher

January 26, 2021

Outline

- Introduction
- Basic authentication mechanisms

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 Identification Methods

- Identification by credentials
 - You show me your driver's license
- Identification by recommendation
 - You introduce me to someone
- 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
 - Active wiretapping
 - Everything is converted to digital signal

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 remote devices to computers, susceptible to password sniffers
- Unless quite 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
- Hard to achieve all this simultaneously

Passwords and Single Sign-On

- Many systems ask for password once
 - Resulting authentication lasts for an entire “session”
- Used on its own, complete mediation definitely not achieved
- Trading security for convenience
- Especially if others can use the authenticated machine

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

Standard Password Handling

The Marx
Brothers'
Family
Machine

Login: Groucho

Password: swordfish

**A one-way
function**

Harpo	2st6'sG0
Zeppo	G>I5{as3
Chico	w*-;sddw
Karl	sY(34,ee,
Groucho	We6/d02,
Gummo	wbnP]

We6/d02,

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 in the process
- Tend to evolve to match user trends

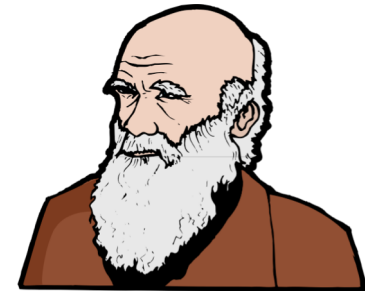
A Serious Issue

- 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?
 - For all Linux systems?

Illustrating the Problem



Karl Marx



Charles Darwin

^*eP6la-

^*eP6la-



aardvark
Aardwolf

340ja
K[ds-
sY(34

beard

^*eP6la-

The Real Problem

- Not just 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

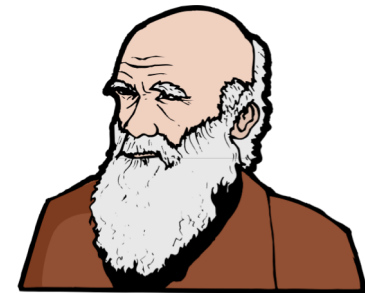
Salted Passwords

- 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



D0C1s6&

aardvark
aardwolf

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



)#4,doa8

beard

^*eP61a-

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
- Note the resemblance to nonces and IVs
- Why is it OK (or OK-ish) to leave this important information in plaintext?

Modern Dictionary Attacks

- Modern machines are very fast
- Even with salting, huge dictionaries can be checked against encrypted passwords quickly
- In 2012, Ars Technica challenged 3 hackers to crack 16,000 hashed, salted passwords
 - Using dictionary attacks, they got 90% of them in 20 hours

GPU Password Cracking

- GPUs Even salted, hashed cracking
 - High passwords are in peril.
 - 200x faster than CPU
- Possible to build cheap 1 GPU unit for this purpose (less than \$2000)
The moral?
- Prototypes crack 20-50% of passwords in example files in a few minutes

Password Management

- Limit login attempts
- Encrypt your passwords
- Protecting the password file
- Forgotten passwords
- Generating new passwords
- Password transport

Limit Login Attempts

- Don't allow dictionary attacks “over the wire”
- After some reasonable number of failed login attempts, do something
 - Lock account
 - Slow down
 - iPhone does this, Android doesn't
 - Watch more closely

Encrypt Your Passwords

- Using the techniques we just covered
- One would think this advice isn't necessary, but . . .
 - Yahoo lost half a million unencrypted passwords in 2012
- Encryption is more expensive and less convenient
 - But a lot more secure

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 on single accounts still work
- And there are “popular” passwords, leading to easy dictionary attacks even with encryption
- Generally, don't give access to the encrypted file, either

Other Issues for Proper Handling of Users' Passwords

- Sites should store unencrypted passwords as briefly as possible
 - Partly issue of how they store the file
 - Partly issue of good programming
- Don't leave passwords in temp files or elsewhere
- 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

Wireless Networks and Passwords

- Wireless networks are often unencrypted
- Web sites used to request and transport passwords in the clear
- So eavesdroppers could hear passwords being transported
- Important to encrypt these messages
 - Use HTTPS if your web site requires passwords

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
- Example of common security theme:
 - Security often at odds with usability

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 an address the user registered with you earlier
- Often fine for practical purposes
- But there are very serious vulnerabilities
 - E.g., unencrypted wireless networks
- If you really care, use something else
 - E.g., surface mail

User Issues With Passwords

- Password proliferation
- Choosing passwords

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?

Compromise in one place compromises you everywhere

Real attacks are based on this vulnerability

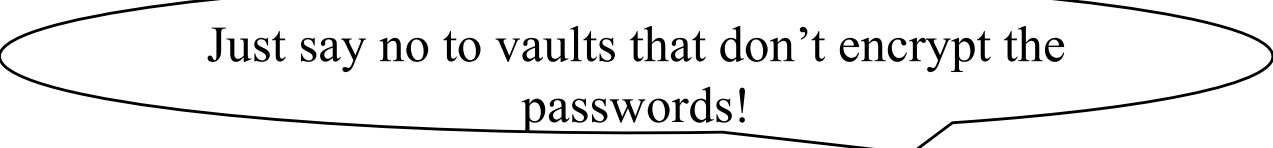
Using Different Passwords

- + Much more secure
- But how many passwords can you actually remember?
- And you might “solve” this problem by choosing crummy passwords
- Or by altering one good password in a predictable way

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
- Use algorithm customized to sites
- Password vaults

Password Vaults

- Also known as key chains and password managers
Just say no to vaults that don't encrypt the passwords!
- Store a lot of passwords in encrypted form
- Require another password to decrypt them
- Typically integrated with web browsers
 - Most users log into sites via browsers, anyway
- Single sign-on issue

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 . . .
- Old sites may demand shorter ones

Some Password Strategies

Some password vaults will choose good passwords for you

- Use first letters from a phrase you remember (or entire phrase, if allowed)
- Use several randomly chosen words
- Replace letters with numbers and symbols
 - Helps, but less if you use common replacements (e.g., “0” for “o”)
 - Also less useful if you limit it to 1st and last character of password

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

Challenge/Response Problems

- 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
 - “Security questions” used as an alternative to passwords

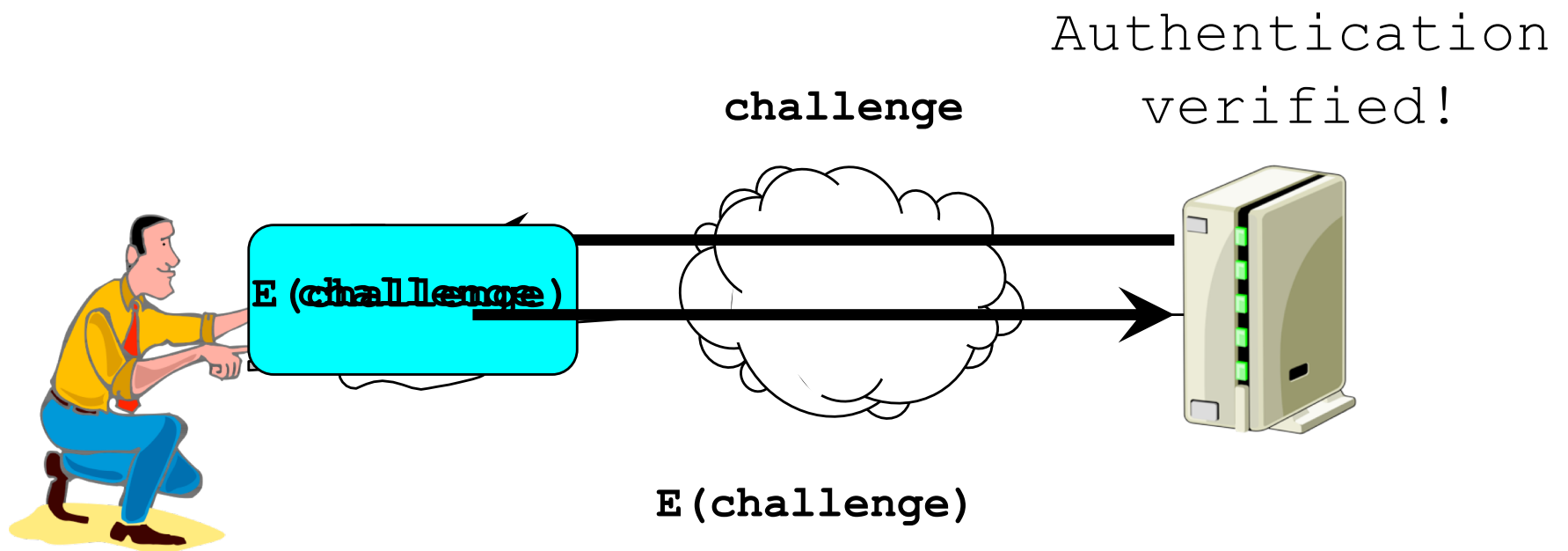
Identification Devices

- 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
- How can we do better?

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 (two factor authentication)
- 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
 - Except systems that use 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
- When attacks are rare or difficult
- 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

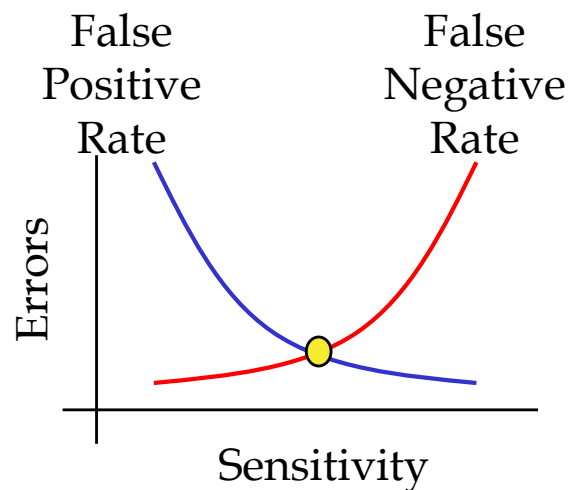
Characterizing Biometric Accuracy

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 lower 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

Biometrics and Usability

- Always a tradeoff in false positives vs. false negatives
- For consumer devices, false negatives are very, very bad
 - People discard devices that won't let the legitimate user in
- Can you make the false positive rate non-trivial with almost no false negs?

Didn't Carnegie Mellon Perfect Facial Recognition?

- Not really
- Quick and dirty version got 1 in 3 right
- With more photos and time, did better
- Facebook claims to have improved on that (97+% on matching photos, 2014)
- But think about how accurate your use of biometrics needs to be
- In many cases, you need 5 nines or so

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

Multi-Factor Authentication

- Our options for authentication have a problem:

They all suck!

- People choose lousy passwords
- Smart cards get stolen
- Biometrics have the Goldilocks problem

So What To Do?

- Maybe reduce the problems of each approach by using the strengths of the others
- Require more than one type of authentication
- Unless all work, don't authenticate

Practical Multifactor Authentication

- Most commonly:
 - Something you know + something you have
 - E.g., password and smart phone
 - Provide the password and respond to a probe of your phone
 - ATMs have used this idea for years
 - Your PIN and ATM card
- Other combos, are possible, though

Is This Better?

- Now two things need to go wrong for false authentication
 - But now either thing can go wrong for a false negative . . .
- Are the factors really orthogonal?
- Are both factors non-trivial?
- Is one factor likely to suffer a catastrophic break?

Multifactor Authentication and Users

- Users generally don't like it
 - At least at first
- Only works when they're motivated
 - E.g., they can't avoid it and can't access vital things without it
- They might get over it . . .

Regardless, It's State-of-the-Art

- All expert security advice says to use multi-factor authentication
- Especially for anything with any serious implications
- Not quite everywhere yet
- But becoming more common