Software: Final Thoughts

- "Purity" (software doing only what you expect) or at least "transparency" (letting you know about extra) becoming important
  - Impure: Anti-cheating Warden snooping your computer in World of Warcraft
  - Opaque: Microsoft LiveOneCare in 2007 changing user settings to re-enable Windows services disabled on purpose
Only some software

- Security issues arise heavily from small group of programs
  - Windows
  - Web Browsers (2?), Microsoft Office, Email Clients (3–5?), Media players (5), Backup
  - Security: Anti-virus and firewall
  - Server-side stuff (including all server OS!)
News flash

- Fox 9:00 p.m. news tonight will have Eugene Spafford on Illinois voting machines and procedures
authenticate  |  oʊ θ  entɪˈkæt |

verb [ trans. ]
prove or show (something) to be true or genuine: they were invited to authenticate artifacts from the Italian Renaissance.

• [ intrans. ] Computing (of a user or process) have one's identity verified.

DERIVATIVES
authentication  |  oʊ θ  entɪˈkæsh ən |
authenticator  |  -kætər |

ORIGIN early 17th cent.: from medieval Latin authenticate- ‘established as valid,’ from the verb authenticare, from late Latin authenticus ‘genuine’.
Authentication is key

- Privacy (i.e., confidentiality) and anonymity are important for our social, business well being, but authentication is essential for survival.
- Who and what to trust and not to trust!
- Human–Human and Human–physical world interactions: sight, sound, smell, observation of body language, etc.
Hog dog!

- Say you want a Chicago-style hot dog
- Maybe you go to Carm’s
- For sure, authentication is key...
Why a hot dog?

What’s the point of the story of getting a Chicago-style hot dog?
- Simple: Human-Human authentication is (relatively) easy
- The hard cases are:
  - Human–Computer System across network
  - Computer System–Computer System
Protocols

Passwords are the most common way to authenticate human to computer system; much more on authentication (password and otherwise) later.

- Can be considered as part of a (simple) protocol.

- But fancier things, or both principals devices, definitely require protocol

  - E.g., Key fob–car; IFF system
Protocols

- A set of rules for how ≥2 principals do something, typically over public communication channel
  - E.g., authenticate one to another; mutually authenticate; vote so all agree on outcome but votes are secret; commit to a value
- Must of course be specified precisely
- Often very delicate; can break if explicit/implicit assumptions don’t hold, or protocol is flat-out breakable.
Common Protocol Ingredients

- Two parties can have secure communication by using cryptography with shared key
  - But must have pre-established key, key distribution, or public-key crypto

- **Nonce** “number used once”—can generate arbitrary random number

- Can generate very crudely synched timestamps
Example: Challenge and response

- Car engine $E$ authenticating smart key fob transponder $T$ once key is inserted into ignition

- Two steps:
  1. $E$ sends $T$ a nonce $N$
  3. $T$ sends back $(T, N)$ encrypted with their shared key
Assumption needed for security

- Nonce must be *unpredictable* pseudorandom number; not just fresh number never used before, such as the date, or next in sequence 1,2,3,....

- Otherwise, car thief can figure out what next challenge to key fob will be, and ask the key fob himself as owner walks away from the car.

- This would work even if fob was checking the newness of the nonce! (Unlikely)
Man-in-the-middle attacks

- Say $E$ allowed fob transponder $T$ to transmit request without being inserted by sending “Please”
  - Crook sends “Please” to $E$, gets back challenge $N$, sends $N$ to $T$; $T$ sends proper response to crook thinking crook is $E$; crook gives this response to $E$.
  - Perhaps unreasonable for ignition key, but how about garage-door remote?
- Many protocols can be broken this way.
Famous Protocol: Needham-Schroeder

- Key distribution protocol from the late 1970s.
- Parties are arbitrary pool of principals and trusted key server S. Allows any one principal A to request S to give a new session key for use by A and B.
- I.e., starts by A telling S that she wants a new session key to communicate with B.
- Each principal has unique shared key with S; denote shared key of A and S by $K_{AS}$
Protocol Notation (so fits on one slide)

- Each line has two parts (separated by colon): 1st parts specifies principal sending and principal receiving; second part gives the message. So
  - $E \rightarrow T: N$ means “E sends T the nonce N” (N will mean a nonce)

- Putting things in brackets with a key subscript means encrypted with that key:
  - E.g., $T \rightarrow E : \{T, N\}_{K_{ET}}$ means “T sends to E T & N encrypted with E and T’s shared key”.
Needham-Schroeder Protocol

\[ A \rightarrow S : \quad A, B, N_A \]
\[ S \rightarrow A : \quad \{N_A, B, K_{AB}, \{K_{AB}, A\}_K_{BS}\}_K_{AS} \]
\[ A \rightarrow B : \quad \{K_{AB}, A\}_K_{BS} \]
\[ B \rightarrow A : \quad \{N_B\}_K_{AB} \]
\[ A \rightarrow B : \quad \{N_B - 1\}_K_{AB} \]
Problem with N–S

- Anybody who steals Alice’s key with Sam ($K_{AS}$) can impersonate Alice to 3rd parties!
- Is this okay?
- Probably not today, but really it’s all about what assumptions you make.
- (Using timestamp for nonce would fix this problem.)
Back to classic user authentication

- User authentication is absolutely crucial
- If you can impersonate someone else (be authenticated as them), you can do anything they can do
- If you can impersonate anyone (totally breaking authentication), you can do (almost) anything on the computer
- Usually hard part of taking over a computer is getting in as any one legitimate user
3 Ways to Authenticate

- Authentication is normally done by one or more of:
  1. What you know (typically a password)
  3. What you have (typically a chip/card of some sort)
  5. What you are (biometrics)

- All of these can fail!
Must balance Errors

- Since authentication errors, must balance:
  - False Acceptance Rate (FAR) (fraud)
  - False Rejection Rate (FRR) (insult)

- Rule of thumb: choose setting where these two are equal ("Crossover Error Rate") but depends on what is being authenticated.
Passwords

- Most commonly used, cheapest, and clearly insecure these days
- Problem is clash of security requirements versus human capability
Password desiderata

- Make them hard to guess: No words in dictionary, no personal info (Birth date, SSN of you or family)
- Use ≥1 digit/punctuation mark & Mixed Case
- Do not reuse
  - Else distinct security protocols become entwined!
- Memorize; never write them down
- Change periodically
Guideline problem

- Password guidelines of previous slide are impossible to carry out
- Nobody can memorize that many distinct high-quality passwords
  - Typical person who does a lot online has 50–100 web accounts
- I know Turing Award winners in crypto/security who do not follow these guidelines!
- Passphrases maybe help some
Inside an organization

- Want an aggressive enough password policy to ward off dictionary attacks
- Key question is “Can you convince your users not to reuse their passwords elsewhere?”
- Helps if you can give them Single Sign-On (SSO)
Password attacks &
countermeasures

- Dictionary/Brute Force attacks: Hence length & character diversity requirements
  - And retry counters, but must balance with difficulty people have entering passwords

- Eavesdropping attacks (including “shoulder surfing”): Be careful when entering in person; design systems not to ever transmit passwords in the clear over LAN

- Bogus machines/Spoofing: Need a trusted path
What you have

- Keys
- Cards/Chips
  - Time-generated number
  - Dumb cards: Returning same thing every time
  - Smarter Cards: Challenge and Response
- Computer itself
What you have attacks

- Stealing or finding
- Copying
- “Side channel”:
  - Measure power consumption of smart card (it takes more power to read bit=1 than bit=0 of secret key because ultimately something electronics)
  - Or timing, radiation, etc.
Biometrics

- Most expensive to maintain
- Inherently imperfect even with perfect users
- Main types:
  - Fingerpring/palm scan (but gelatin molds)
  - Hand geometry
  - Retina/iris scan (very high accuracy)
Biometric techniques (cont)

- Voice print
  - can be distorted by colds, defeated by recordings
- Keyboard dynamics
  - Can record and playback
Social engineering

- A whole universe of clever attacks
Coda: Kerberos

- Computer network authentication protocol, developed at MIT, today distributed as free software by MIT
  - Named for monstrous 3-headed dog guarding Hades
- Classified as a munition by US and therefore illegal to export until crypto policy change around 2000 in light of *Bernstein v. U.S.*
- Used in Windows 2000, XP, Vista; Mac OS X
Kerberos Protocol

- Based on Needham-Schroeder, but (of course!) uses timestamp instead of nonce; adds notion of lifetime
- Trusted 3rd party, Key Distribution Center (KDC), has 2 logically separate entities:
  - Authentication Server (AS), to which users log on
  - Ticket Granting Server (TGS) gives tickets allowing access to resources (e.g., files)
Protocol itself

1. Alice logs onto AS using password, and gets session key $K_{AS}$ for talking with TKS.

2. To get access to resource B, Alice uses $K_{AS}$ for protocol with TKS that is like Needham–Schroeder except: Alice doesn’t send nonce in her first message; instead TKS sends time stamp a lifetime in its response.

3. Result is key with time stamp and lifetime used to authenticate Alice’s traffic with resource B.
Kerberos Weaknesses

- Requires clock synchronization; complex deliberate attack could even attack the clocks
- Single point of failure: When the Kerberos server is down, nobody can log in.