Authentication

- Another trilogy: identification, authentication, authorization
- ACLs and the like are forms of authorization: what you're allowed to do
- Identification is whom you claim to be be
- Authentication is how you prove it

Forms of Authentication

- Something you know
- Something you have
- Something you are
- (Hmm, yet another trilogy)

Forms of Authentication

- Something you know: passwords
- Something you have: smart card
- Something you are: fingerprint

Something You Know

- Ancient: “what’s the secret word?
- Modern incarnation: passwords
- Most common form of authentication

Passwords

- Everyone understands the concept
- Passwords should be sufficient
- Not really. . .

Passwords are Really Bad

- Guessable
- Forgettable
- Enumerable
- Eavesdroppable (but that isn't a word. . .)
- Replayable
- Reuseable
- Leakable
- Probably a lot more reasons not to use them

Guessable Passwords

- People tend to pick bad passwords
- Own name, phone number, spouse's name, kids' names, etc.
- Easy to write password-guessing program (Morris and Thompson, CACM, Nov. 1979)

Password-Guessing Programs

- Try likely words: names, dictionaries, etc.
  Use specialized dictionaries, too: other languages, science fiction terms, etc.
- Try variants: “password” $\longrightarrow$ “passw0rd” or “Password”
- Use specialized, optimized algorithm
- In uncontrolled environments, 40-50% of people will have guessable passwords

How Are Passwords Stored?

- Not in plaintext
  - Administrator can see them
  - Can be stolen from backup media (or recycled disk drives. . .)
  - Editor bugs can leak them
  - Something that doesn't exist can't be stolen!
- Use a one-way hash; compare stored hash with hash of entered password
- Read-protect the hash passwords anyway

Guessing Strategies

- Online: try to log in as the user
- Offline: steal a copy of the password file and try on your own machine (or on many compromised machines)

Defenses

- Rate-limit online guesses
- Perhaps lock out the account – but that leaves you vulnerable to DoS attacks
- Make password-guessing inherently slow: use a slow algorithm

The Unix Password-Hashing Algorithm

- Use DES
- Don't encrypt the password, encrypt a constant (all 0s) using the password as the key

☞ This is where the 8-character limit comes from

- Any decent cryptosystem can resist finding the key, given the plaintext and ciphertext
- Repeat 25 times, to really frustrate an attacker
- Guard against specialized hardware attacks by using the "salt" to modify the DES algorithms

Salt

- Pick a random number — 12 bits, for Unix — and use it to modify the password-hashing algorithm
- Store the salt (unprotected) with the hashed password
- Prevent the same password from hashing to the same value on different machines or for different users
- Makes dictionary of precomputed hashed passwords much more expensives
- Doesn't make the attack on a single password harder; makes attacks trying to find some password $4096\times$ harder

Examples of Salting

Without Salt	*With Salt*
joe →`0x21763a`	joe →`0,0x21763a; 1,0x0e08e7; 2,0x4fea4b; .`
fred→`0xc19ecf`	fred→`3,0xc19ecf; 4,0x55be45; 5,0xf0b015; .`
pat →`0xfcef3d`	pat →`6,0xfcef3d; 7,0x261286; 8,0x2437ba; .`
sue→`0x71ca7a`	sue→`9,0x71ca7a; 10,0x83f700; 11,0x04ed54;`
...	...

Why Does Password-Guessing Work?

- People are predictable
- Passwords don't have much *information*
- According to Shannon, an 8-character word has 2.3 bits/character of information, or a total of 19 bits
- Empircally, the set of first names in the AT&T online phonebook had only 7.8 bits of information in the whole name
- 2^{19} isn't very many words to try...

Can We Lengthen Passwords?

- There are other possible hashing algorithms that don't have an 8-character limit.
- Using AES in the same way would let us use 32-character pass phrases; using HMAC-SHA1 would permit unlimited length
- Are long passphrases guessable?
- Running English text has entropy of 1.2-1.5 bits/character — but no one has built a guessing program to exploit that
- No one knows

Forgettable Passwords

- People forget seldom-used passwords
- What should the server do?
- Email them? Many web sites do that
- ☞ What if someone can read your email?
- Reset them?
- ☞ How do you authenticate the requester?
- Password hints?
- Is it bad to write down passwords? If your threat model is electronic-only, it's a fine thing to do. If your threat model is physical, forget it. (See the movie "Ghost")

Reuseable Passwords

- People tend to reuse the same passwords in different places
- If one site is compromised, the password can be stolen and used elsewhere
- At the root of “phishing” attacks

Eavesdroppable

- Wiretapping the net isn't hard, especially if wireless links are used
- Done on the Internet backbone in 1993-4
- Install a keystroke logger on the client
- Install a password capture device on the server
- Play games with the DNS or routing to divert the login traffic

Stealable

- Shoulder-surfing
- Bribery — trade a password for a candy bar (`http://www.securitypipeline.com/news/18902074`)

The Fundamental Problems

- Passwords have to be human-useable
- Passwords are static, and hence can be replayed

Something You Have

- Many forms of tokens
- Time-based cards
- USB widgets (“dongles”)
- Rings
- Challenge/response calculators
- Cell phones
- Smart cards
- Mag stripe cards
- More

Disadvantages of Tokens

- They can be lost or stolen
- Lack of hardware support on many machines
- Lack of software support on many machines
- Inconvenient to use
- Cost

NSA's STU-III Secure Phone

Photos courtesy of Richard Brisson

And the Crypto-Ignition Key

How STU-IIIs are Used

- The phones have cryptographic keying material, and are in controlled areas
- The keys also have keying material, and user's name and clearance level
- Keys are associated with particular phones
- You need both the key and access to the right phone to abuse it
- *Two-factor* authentication

Two-Factor Authentication

- Two of the three types of authentication technology
- Use second factor to work around limitations of first
- Example: SecurID card *plus* PIN

Eavesdropping Again

- Can't someone eavesdrop on a token-based or two-factor exchange?
- Sure!
- Must use other techniques as well: encryption and/or replay protection

Replay Protection

- SecurID: code changes every minute; database prevents replay during that minute
- Challenge/response: server picks a unique number; client encrypts it
- Cryptographic protocols

Cryptographic Authentication

- Use cryptographic techniques to authenticate
- Simultaneously, negotiate a key to use to protect the session
- But where do the original cryptographic keys come from?

Cryptographic Keys are Long

- An AES key is at least 128 bits. Care to remember 32 hex digits as your password?
- An RSA key is at least 1024 bits. Care to remember 256 hex digits as your password?
- Solution 1: store the key on a token
- Solution 2: store the key on a computer, but encrypted

Storing Keys on Tokens

- The most secure approach (my Java ring has an RSA key pair on it)
- Proper integration with host software can be tricky
- Generally want two-factor approach: use a password to unlock the token
- Ideally, the token is tamper-resistant

Storing Keys on Hosts

- Software-only approach is useful for remote logins
- *Must* use passphrase to encrypt key
- Not very resistant to capture of encrypted key — we're back to offline password guessing
- Can you trust the host to protect your key?

Use a Passphrase as a Key?

- Convert the user's passphrase to a key, and use it directly
- Approach used by Kerberos
- Remember the low information content of passphrases. . .
- Attack: eavesdrop on an encrypted message; guess at passphrases; see which one yields a sensible decryption
- Solution: use a SPAKA (Secure Password and Key Agreement) protocol

Why Should Tokens be Tamper-Resistant?

- Prevent extraction of key if stolen
- Note: recovery of authentication key *may* permit decryption of old conversations
- Prevent authorized-but-unfaithful user from *giving* away the secret — you can't give it away and still have use of it yourself.
- One guy put his SecurID on a webcam: `http://fob.webhop.net/`

Much More Next Class. . .

- Biometrics
- Systems issues