Authentication

Authentication

- A 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? (Supposedly dates to at least Roman times.)
- 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
- Reusable
- Leakable
- Probably a lot more reasons not to use them

Guessable Passwords

- People tend to pick bad passwords
- Their 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” → “passw0rd” or “Password”
- Use specialized, optimized algorithm
- In uncontrolled environments, at least 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 hashed passwords anyway

Guessing Mechanisms

- 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—including their GPUs)
- Note: that's why we read-protect the hashed passwords

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 Classic Unix Password-Hashing Algorithm

- Use DES (encryption algorithm with 56-bit keys in 8 bytes)
- Don't encrypt the password, encrypt a constant (all 0s) using the password as the key
- (Why not encrypt the password?)

The Classic Unix Password-Hashing Algorithm

- Use DES (encryption algorithm with 56-bit keys in 8 bytes)
- Don't encrypt the password, encrypt a constant (all 0s) using the password as the key
- (Why not encrypt the password?)
- ☞ This is where the 8-character limit comes from
- Any decent cryptosystem can resist finding the key, given the plaintext and ciphertext
- Iterate 25 times, to really frustrate an attacker
- Guard against specialized hardware attacks by using the "salt" to modify the DES algorithm

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 expensive
- Doesn’t make the attack on a single password harder; makes attacks trying to find *some* password 4096× 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 256-bit AES in the same way would let us use 32-character pass phrases; using HMAC 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 if it's even possible to exploit it

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?
- ☞ Only possible if the passwords are stored in plaintext
- 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")
- ☞ Don't neglect the threat of abusive domestic partners

Email and Password Recovery/Reset

- Emailing a password is the most common means of password recovery or reset
- This means that protecting your email account is crucial—it controls access to most of your other accounts
- Also: high-value systems, e.g., bank accounts, can't rely on email for reset

Eavesdroppable

- Wiretapping the net isn't hard, especially if wireless links are used
- Done on the Internet backbone in 1993-4; see CERT Advisory CA-1994-01
- 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://news.bbc.co.uk/2/hi/technology/3639679.stm)

Reusable 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

☞ A fraud incident on Stubhub is believed to have used passwords stolen from Adobe.com.

☞ Reusing passwords is a much greater ill than picking weak passwords

Password Managers

- Store passwords in an encrypted file
- Who can see this file?
- How strongly is it protected?
- People use many machines today—synchronize this database? How?
- Can malware get at the database?
- How is it used?

☞ If the manager recognizes web sites, it can help protect against phishing

Password Mangers: Cloud-Based

- Simplifies use from multiple devices
- Allows for provider-based intrusion monitoring
- (Allows the provider to charge a recurring fee for access. . .)
- But: can an attacker launch guessing attacks on the password used to protect this database?

The Fundamental Problems

- Passwords have to be human-usable
- 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
- Mobile 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

The Java Ring

This ring has a Java interpreter, a crypto chip, and certificate-processing code.
Google and others are pushing NFC devices of various sorts

NSA's STU-III (Ancient!) 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
- Each party’s phone will display the other party’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

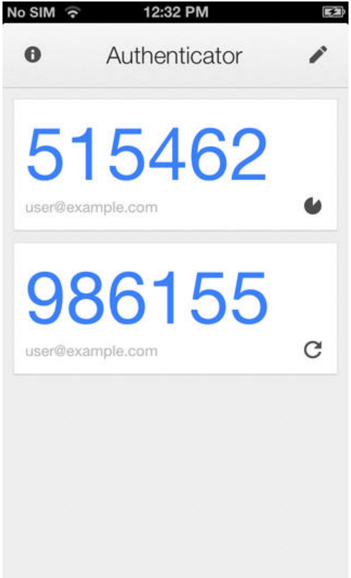
Two-Factor Authentication

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

SecurID Tokens

A SecurID token on two successive time cycles. The bars on the left of the second picture indicate how many 10-second ticks remain before the display changes, in this case about a minute. In essence, the display shows $H_k(T)$, where T is the time and H_k is a keyed hash function.
Generic name: TOTP (Time-based One-Time Passwords)

Soft Tokens



- Phone apps can do the same things as dedicated tokens (CU uses Duosec)
- The partially-filled circle shows the time left for that code; there's a refresh button to generate a new one
- But—is the cryptographic secret protected as well as on dedicated tokens? There are hardware and software attacks possible now

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 random hex digits as your password?
- An RSA key is at least 2048 bits. Care to remember 512 random 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 login 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.
- Folks have pointed cameras at their tokens and OCRed the digits. . . http://smallhacks.wordpress.com/2012/11/11/reading-codes-from-rsa-secureid-token/

Mobile Phones

- Use a phone as a token: send an SMS challenge to the phone
- Indepedent failure mode: will the attacker who has planted a keystroke logger on a computer also have access to the owner’s phone?
- ☞ Eavesdropping on a phone requires very different access and technology than hacking a computer or eavesdropping on WiFi.
- Are there privacy risks from everyone having your mobile number?
- What about malware on the phone?
- Twitter’s variant: app talks directly to Twitter and user; easier to use

Other Threats

- Bogus SIM cards, with the help of a deluded carrier
- An attacker who controls the phone network
- Inceasing linkage between hosts and phones reduces the second factor: it’s no longer independent

Federated Authentication

- Log in—via strong-but-inconvenient authentication—to Facebook, Google, etc.
- These sites vouch for your identity to other sites
- What about privacy? (Mozilla's solution tries to solve this.)
- Do you trust some other site to vouch for your users? Your employees?

Today’s Status

- The evils of passwords have become very, very apparent
- There is a strong push to get rid of them, especially by Google
- But will they succeed?
- Passwords seem easy and cheap, and don’t require (much) user training—but is that still true if you account for password recovery and compromise?

Analyzing Password Security Practices

What’s the First Question?

What's the First Question?

- What are we trying to protect, and against whom?

What's the First Question?

- What are we trying to protect, and against whom?
- That's always the first question! So: what are we trying to protect?

What’s the First Question?

- What are we trying to protect, and against whom?
- That’s always the first question! So: what are we trying to protect?
- We have to protect the plaintext password—possession of it gives access to this site and (probably) many others
- We want maximal protection against all enemies
    - Why “all enemies”?

What’s the First Question?

- What are we trying to protect, and against whom?
- That’s always the first question! So: what are we trying to protect?
- We have to protect the plaintext password—possession of it gives access to this site and (probably) many others
- We want maximal protection against all enemies
  - Why “all enemies”?
  - Passwords are used in almost all contexts, even if supplemented by 2FA

First Step

- Never store passwords in plaintext—what if the machine with the password store is hacked?
- Use hashed passwords, not encrypted passwords. (Why?)
- Implication: no password recovery, only password reset
- What's next?

Second Step

- Assume that the machine holding the hashed passwords is hacked—now what?

Second Step

- Assume that the machine holding the hashed passwords is hacked—now what?
- Harden the hash against password-guessing
- This is as far as Morris and Thompson went—is there more?

Third Step

- In 1979, computers were expensive; passwords were generally entered over dial-up phone lines
- Governments could do modem taps; few others could
- Today, passwords are entered over the Internet—much easier to tap
- Conclusion: we *must* use encryption
- More?

Fourth Step

- Passwords today are mostly for web services, but web servers are fragile
- Conclusion: store the hashed passwords on a login server that is (somehow!) less vulnerable to being hacked
- Ideally, encrypt the password from the user to the login server, so that even the web server can't see it (but this is rarely done)
- More?

The User Perspective

- The above is a server-centric perspective
- Users have to worry about many sites, not just one or two
- In Morris and Thompson's day, very few people had more than one login—today, many people have hundreds of logins
- User perspective: some sites *will* be hacked—how do they protect their logins on other sites?

The User Perspective

- The above is a server-centric perspective
- Users have to worry about many sites, not just one or two
- In Morris and Thompson's day, very few people had more than one login—today, many people have hundreds of logins
- User perspective: some sites *will* be hacked—how do they protect their logins on other sites?
- Answer: you *must* use a separate password for each site
- Thought exercise: does it work to have separate classes of password, for sites of different sensitivity?

Why Didn’t Morris and Thompson Go Further?

Why Didn't Morris and Thompson Go Further?

- Technology of the time!
- The external attack surface for a computer of that era was typically very small: the login service
- ☞ Networking was in its infancy
- Users had *no* local compute capacity and little or no local storage
- A separate login computer would have cost (at a minimum) tens of thousands of dollars; it and the network would have been an additional availability failure point
- What do you do with a stolen password file? Compute time was expensive and not that readily available

The worse mistake in technology is to give yesterday's answers to today's questions. The second worse mistake is to ignore yesterday's answers. . .

Implicit Constraints

- Some of the constraints Morris and Thompson faced were implicit
- Their threat model wasn't as clearly spelled out as would do today
- Getting this things right is crucial, especially for security for long-lasting systems
- Do you have an ongoing process for monitoring changes in technology and threat model?

Questions?

(Red-bellied woodpecker, Riverside Park, December 20, 2021)