

# Case Study: Building an Authentication System



# Let's Build an Authentication System

- Actually, let's build two
- The first is for a social network; the second is for a bank
- We'll do the social network first
- How should we authenticate?

# What Should it Look Like?

# Wrong!

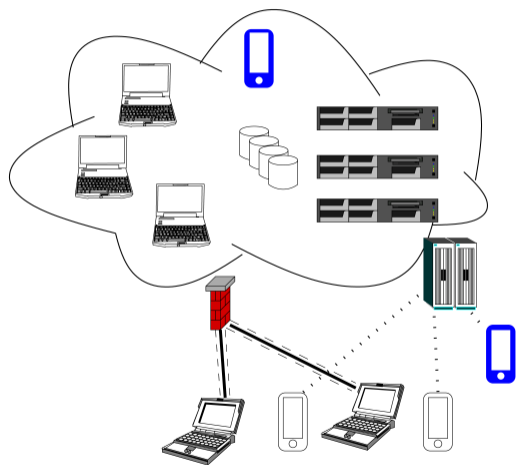
- We haven't answered our first question yet: what are we trying to protect, and against whom?
- We can't answer those until we know more about the service

# The Social Network

**Inside** Developers,  
servers, database,  
social network  
admin, sysadmins

**Border** Firewall, web server

**Outside** Remote developers  
via VPN, users,  
social network  
admin



# What Are the Resources to Protect?

# What Are the Resources to Protect?


- User interaction database
- Code base
- Developers machines?
- Social network admins' machines?
- Users' machines?

# Who Are the Attackers?



# Who Are the Attackers?

- Joy hackers?
- Targeted mischief makers?
- Intelligence services?

- Probably all of the above!
  - Imagine what a hostile intelligence agency could do with the account of an important politician
  - Just building the social graph is useful to intelligence agencies
-  Do spooks socialize more with other spooks? Probably...

- Passwords or multi-factor?
- 👉 What is a good second factor?
- One authentication service or two?

# Execution Environments for Different Actors

**Developers** Can view and modify the codebase

**Sysadmins** Can control more or less anything

**Social network admins** Can view and modify the user interaction database

**Users** Can interact, and can control own posts

# Where Does Good Site Authentication Help?

	Interaction DB	Code Base	Developer Machines	Social Admin Machines	User Machines
Developers	?	✓?	✓	-	-
Rem. Devels	?	✓	?	-	-
Sysadmins	✓	✓	✓	✓	-
Social Admins	✓	-	-	X	-
Users	✓	-	-	-	X

✓ Strong site authentication helps

? It might help, but there are other good attacker paths

X It doesn't help

- Site authentication is irrelevant

# The Limits of Authentication

- The site doesn't control all authentication issues, e.g., users to their phones
- There is always the potential for hacking—and for some situations, it's more of a threat than stolen credentials
- Sysadmins have a lot of power (which we'll talk about later in the term)

# Where Should Authentication Servers Live?

- Developer and sysadmins need a server inside the firewall
- Remote developers authenticate first to the firewall, and are then regular developers, i.e., inside
- Sysadmins are inside
- Users authenticate outside
- Social network administrators could be inside or outside

# One Authentication Server or Two?



# One Authentication Server or Two?

- You do not want outsiders consulting an internal authentication database—it's too sensitive
- It's safer for inside social network administrators to call out through the firewall
- 👉 They have to do that anyway, to reach the web server
- Conclusion: two authentication databases is probably the right path

- 1 We will use two authentication systems

# Password Myths

- Require frequent password changes?
  - 👉 No! Research suggests that not only does it not help, it's actually harmful: people forget their passwords and have to resort to secondary authentication
- Require lots of special characters?
  - 👉 No—longer passwords are more secure than using special characters.  
( $95^8 \approx 6.6 \cdot 10^{15}$ ;  $26^{12} \approx 10^{17}$ )


The latest NIST guidance agrees with these points

# Simple Passwords or Multifactor Authentication?

# Simple Passwords or Multifactor Authentication?

- Multifactor. . .
- We've seen far too many attacks on user social network accounts
- Other resources are even more important
- But—will users accept it?

- There are a number of other issues to consider, with varying tradeoffs
- Having two different authentication databases lets us make different tradeoffs for the different databases

- Who likes using MFA to log in to, e.g., Courseworks?
- Answer: no one, especially when it's in addition to a password
- Then why do we all use it?
-  Because we have to!
- Will users accept a social network that *requires* multifactor authentication? Or will they go elsewhere?
- What is the *cost* of dealing with compromised user accounts? What is the *cost* of lost business?

# Do Some Users Want MFA?

- Absolutely—they recognize the threat
- Many high-profile Twitter accounts have been compromised
- Example: when the Associated Press account was hacked and a fake tweet was sent about a bomb at the White House and Obama being injured, the Dow Jones average dropped 1% in seconds



- Is your chosen MFA technology accessible to, e.g., blind employees or users
- 👉 Can a phone's screen reader properly handle a TOTP display?
- 👉 What about finding the QR code to load the secret, etc., into the phone?
- What does the law say about accessibility issues for employees? For users? What if it's a heavily regulated industry, e.g., a bank?
- Apart from legal issues, there's a moral imperative!

# There May Be Regulatory Issues Requiring MFA

- A Twitter administrative account was once hacked because they didn't use MFA and an admin stored his Twitter password in his gmail account—and that fell to password-guessing
- Another time, an administrative account was brute-forced via online guessing
- Fake tweets were sent from a variety of user accounts, including President Obama and Fox News
- MFA would have prevented these attacks
- The Federal Trade Commission acted. . .

# The FTC: (Some) Problems at Twitter

- Require employees to use hard-to-guess administrative passwords that they did not use for other programs, websites, or networks;
- Suspend or disable administrative passwords after a reasonable number of unsuccessful login attempts;
- Provide an administrative login webpage that is made known only to authorized persons and is separate from the login page for users;
- Enforce periodic changes of administrative passwords
- Impose other reasonable restrictions on administrative access, such as by restricting access to specified IP addresses.

- There's a strong argument for making it mandatory for employees
- There's a strong argument for making it available, but perhaps not mandatory, for users

## Developers

- 1 MFA use should be required

## Social Network Users

- 1 MFA should be available. Note that social network admins are employees, another reason for that interface to support it

# Types of Authentication: Employees

- Should we use passwords as one factor? Probably.
- What's second, biometrics or some sort of token?
- And what sort of biometric or token?

# What Type of Biometric?

# What Type of Biometric?

- Note: I'm talking about a login biometric, not a phone-unlock one
- Only two types appear vaguely suitable, facial recognition (using laptop cameras) or fingerprints
- But laptop fingerprint readers are rare, and when present are used for device unlock
- And what about remote users?
- 👉 Biometrics do not appear suitable
- Conclusion: we need a token as a second factor



# What Type of Token?

- There are dedicated TOTP tokens—but do they offer enough security advantages over a phone-based soft token?
- FIDO2, e.g., Yubikey
- Text message

# What is FIDO2?


- Supports an industry-standard authentication protocol
- Supported by all major browsers
- Usable for login on MacOS, Linux, Windows 10 and later
- Provides cryptographic authentication and prevents MitM attacks
- But—users have to carry extra hardware with them

- Pretty good, but doesn't prevent MitM attacks
- But soft tokens are cheap, and almost everyone has a smartphone (and probably *every* developer)
- A good choice, but not as good as FIDO2

- Rapidly falling out of favor
- Issue: SIM-jacking
- Issue: SS7 hijacking
- Both attacks seen in the real world (especially if governments are the attackers)

- Have the login screen display a QR code
- A phone app reads the code and generates a TOTP
- Cheap!
- Like TOTP, but without MitM issues (but vulnerable to phishing)
- But—doesn't authenticate phone logins; FIDO2 can

- What do these solutions cost?
- FIDO2 tokens cost money, but not very much compared to the loaded cost of a developer
- Soft tokens cost even less
- Camera-based TOTP should be low, too
- But—what does the software cost?

- What does the company-side software cost?
  - Look at the full picture: support for all platforms, administration platform, provisioning, database backup, logging and auditing, and more
  - Remember that people cost much more than hardware
  - The big reason that the SecurID card did so well in the market is that they provided a full software suite
-  Software costs suggest that supporting more than one scheme may be infeasible

# Which Second Factor for Users?

- FIDO2 seems very secure, but you probably can't afford to buy them for your users
- You could add support for users who buy their own
- You could also add TOTP support
- Camera plus TOTP? Maybe, but you might have to write the app
- Text message? Far better than just a password, but not strong enough against serious attackers
- Do you have to pay twice for your software package?
- If your users log in from their phones, how is a phone a second factor?



# Conclusion: Tokens for Employees

- FIDO2 is the strongest; TOTP the cheapest, and is secure enough for local logins
- (Some FIDO2 tokens even work with phones)
- Need to balance cost against threats
- The rapid growth of host and browser support for FIDO2 is encouraging, but watch out for the support software costs

## Conclusion: Tokens for Users

- If you think users may be targeted by serious actors, support FIDO2
- Btw: you probably need FIDO2 support anyway, for your social network admins
- TOTP is pretty good, but is vulnerable to phishing attacks

## Developers

- 1 MFA use should be required, including for social network admins
- 2 FIDO2 is probably the best choice

## Social Network Users

- 1 MFA should be available
- 2 FIDO2 support is needed for employees; TOTP with soft tokens is more accessible to most users

# Lost Credential Recovery

- What about recovery from lost credentials: forgotten passwords, lost FIDO2 tokens, phone problems?
- You have to recover, and recovery *securely*
- Local employees are easy: have them show their badge to the help desk (or equivalent), or let their manager request/authorize a replacement
- What about remote employees? Users?

# Remote Employees

- A *very* difficult problem
- Can you overnight a new token to them? What if they're on the road?
- What is the tradeoff between cost and lost productivity?
- Do you fall back to secondary authentication? Not very secure!
- TOTP via a phone app might be a good fallback: use secondary authentication to the help desk to enable that for that employee for a limited time
- But that's more software complexity
- There are no perfect answers!

# Lost Credential Recovery: Users

- *Very* difficult and *utterly* mandatory
- Common fallback: email for password recovery
- But what about a lost FIDO2 token?
- A password plus email? Many people will use the same password for email and your service
- Or maybe their password for your service is stored in their email account
- Worst of all: you can't afford to spend much on recovery

# Bootstrapping Authentication

- Ultimately, it's a cost/benefit tradeoff
- Email plus password is probably the best you can do for a lost FIDO2 or TOTP credential
- Possibly—though setting it up and administering it is expensive—have a paid support tier (but except for popular services, very few will take advantage of it)

# Authentication Architecture

- Where do we put authentication databases?
- Developers: inside the firewall
- But—*really* lock down the machine
- What about the user authentication database?



# User Authentication Database

- Clearly, inside the firewall
- There is also a database of user profiles
- Do we put authentication data in the same database?
- Remember: if you have two databases, they *will* get out of sync

# Separate Databases!

- The user database has far more kinds of activity and hence has a larger attack surface
- There are many activities that will write to it
- 👉 It is in the execution environment of more processes
  - So: put authentication data in a separate database
  - (More design details later in the term)

## Developers


- 1 MFA use should be required, including for social network admins
- 2 FIDO2 is probably the best choice
- 3 Internal, locked-down database
- 4 Recovery via management chain and overnight shipping

## Social Network Users

- 1 MFA should be available
- 2 FIDO2 support is needed for employees; TOTP with soft tokens is more accessible to most users
- 3 Separate database for authentication only
- 4 Recovery via email, plus password for token loss

# What About Banks?

- Most of the analyses are the same
- However: more is at risk, but there's more money to spend solving the problems
- Should MFA be mandatory?
- Hard—it's a competitive market
- Query: who is liable for financial losses, the customer or the bank?

- Under US law, consumers are generally not liable; businesses are
-  But laws differ in other countries—are you a multinational bank?
- But—consumers generally have far less money, so the bank's losses are limited
- Always offer strong MFA—remember that phishing is a serious problem, so MitM resistance is important
- Perhaps give FIDO2 tokens to high-value customers and business customers
- Let people visit a branch for replacement tokens
- Or: mail or overnight replacements—you always have physical addresses
- (Might a serious attacker—or disgruntled former spouse/partner—stake out the mailbox?)

- Authentication is a systems problem
- It's much more complex than just “use passwords” or “use MFA”
- There are tradeoffs, and there are problems with no great solutions

# Questions?



(Red-shouldered hawk, Central Park, December 26, 2021)