IN4MATX 285: Interactive Technology Studio

Practice: Logins

Today's goals

By the end of today, you should be able to…

- Articulate why logins are needed to create desired user experiences
- Differentiate authentication from authorization
- Describe the utility of supporting authentication and authorization in interfaces
- Describe the advantages and disadvantages of OpenId

Going back to storage…

Storage

- Your browser can hold a (small) amount of information about you and your activities on a website
  - Things you search for or type into a text box
  - **Some form of login credentials**
  - Types of content you often to engage with
- When you later return to a website, it can tailor the content based on what was stored

Storage

- Your browser isn’t storing your Instagram feed, emails, Amazon shopping history, etc.
- But it is keeping you logged in, and will ask the server to retrieve your specific content

Storage

- Why do you need a login, rather than personalizing content to your device?
  - Need to support shared devices and public devices (e.g., you can log out, someone else can log in)
  - Need to support sharing information across your devices (e.g., your phone and laptop need to be associated with the same account)

Logins: how do they work?

Key terms: authentication & authorization

What is authentication?

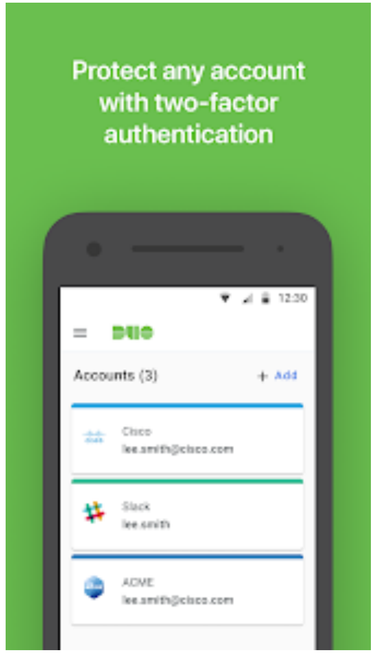
- The process of establishing and verifying identity
- Identification: who are you? (username, account number, etc.)
- Authentication: prove it! (password, PIN, etc.)

What is authorization?

- Once we know a user's identify,
  we must decide what they are allowed to access or modify
- One way is the app defines permissions upfront based on a user's role
  - A student can access their own grades, but not modify them
  - A TA and a professor can access and modify everyone's grades
- Another way is for the app to request the user grant certain permissions
  - A Twitter app may ask, "can I Tweet on your behalf?"

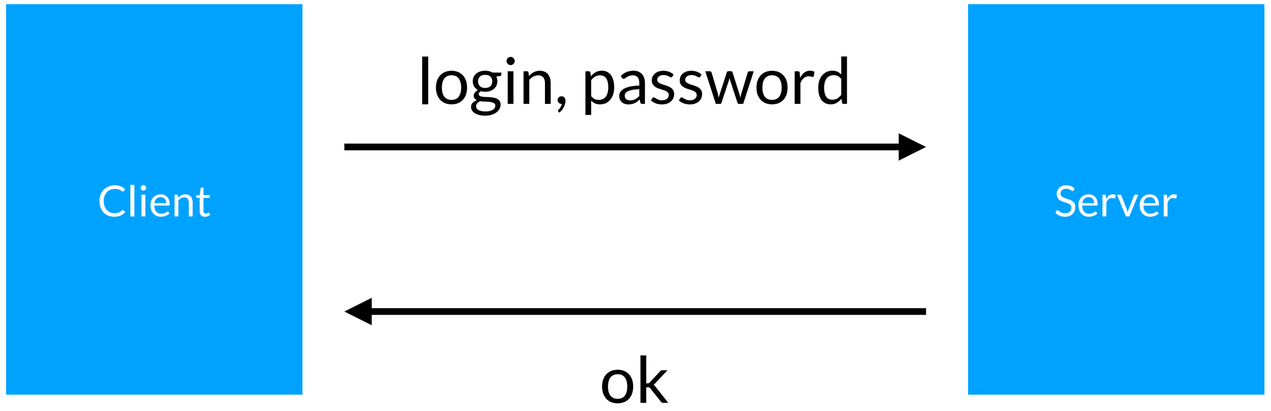
Multi-factor authentication

- Should be a mix of things that you *have/possess* and things that you *know*
- ATM machine: 2-factor authentication
  - ATM card: something you *have*
  - PIN: something you *know*
- Password + code delivered via SMS/Push: 2-factor authentication
  - Password: something you *know*
  - Code: validates that you *possess* your phone
- Two passwords != Two-factor authentication



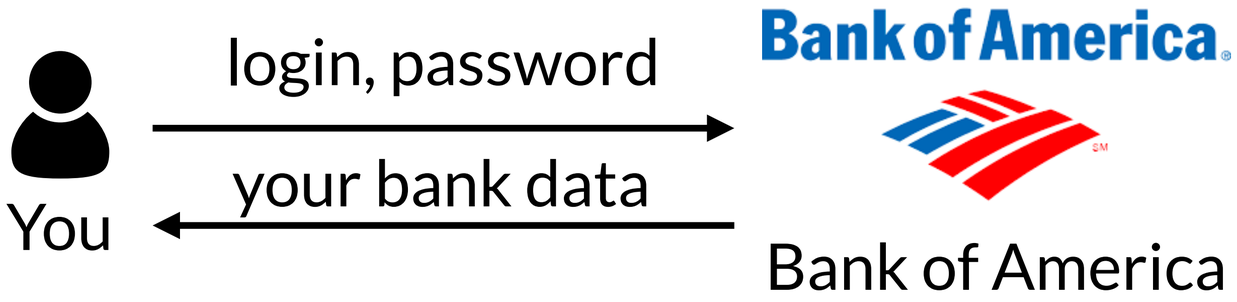
Password protocol

- Send a login and a password to a server
- Server checks your credentials and okays you



Password protocol: sending data

- Once you've logged in,
  the server can send you whatever data you're allowed to see



Logins and security

Logins and security

- Passwords ***should not*** be stored as plain text
- Why not? Breaches. So many breaches

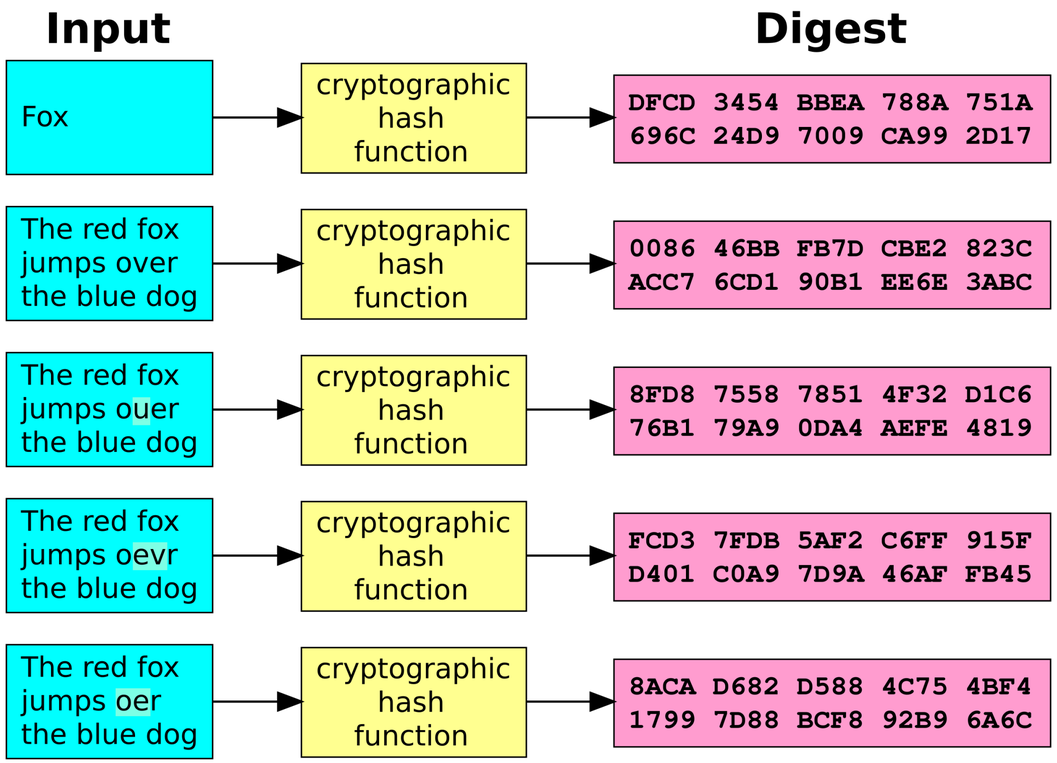
Update, May 6, 2025: This story, originally published May 3, has been updated with details of the SMS phishing threat posed by the Chinese Panda Shop cybercrime group, and an open letter to the cybersecurity industry asking why the phishing threat behind the stolen passwords epidemic has yet to be fixed.

In just the last few months, I have reported on confirmed lists of stolen passwords being made available on the dark web and in criminal forums that have risen from 800 million to 1.7 billion and even as high as 2.1 billion, mainly thanks to the rise and rise of infostealer malware attacks. But a new report has just blown even those shockingly large statistics out of the water with an analysis of 19 billion such passwords that are available online right now to any hackers who want to seek them out. The takeaway being that you need to take action now to prevent becoming a victim of the automatic password hacking machine epidemic.

https://www.forbes.com/sites/daveywinder/2025/05/06/new-warning---19-billion-compromised-passwords-create-hacking-arsenal/

Logins and security

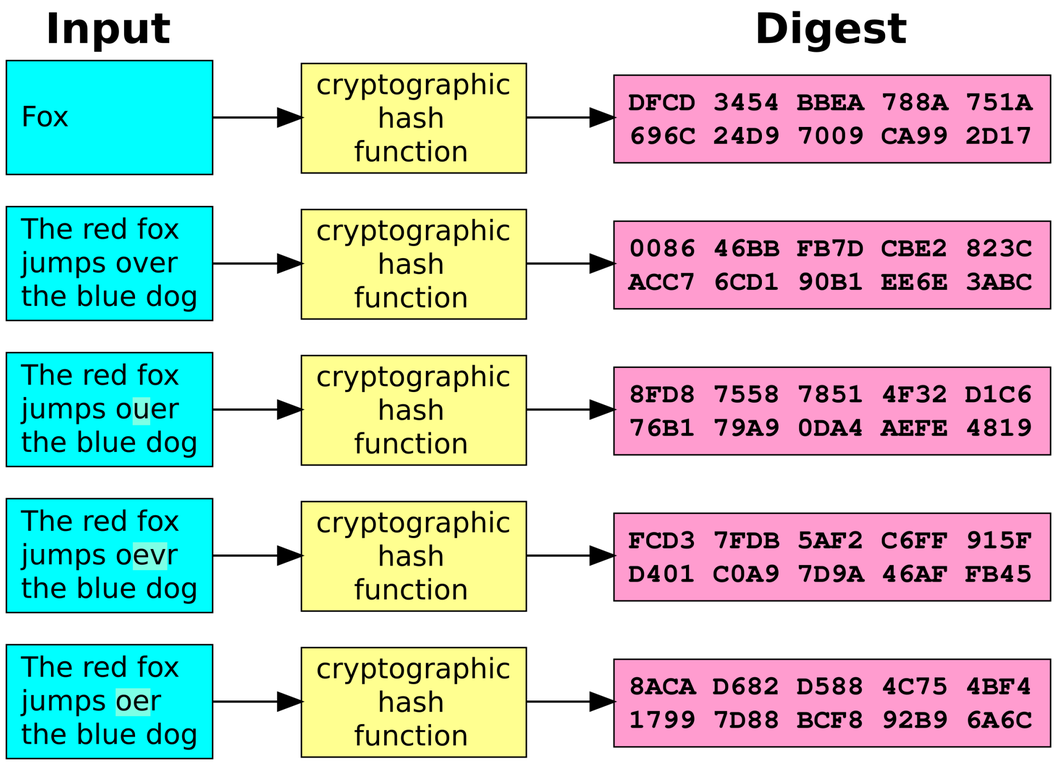
- Then, how do we store passwords?
- We *hash* them
  - Given an input string, create an unpredictable, but consistent, output string
  - Don't store [`Fox`], instead store [`DFCD 3454 ...`]



https://en.wikipedia.org/wiki/Cryptographic_hash_function

Logins and security

- Then, when a user types their password, hash what they type and compare it
  - You know whether the password is correct or not, but can't correct typos
  - You can't say "oh you forgot to capitalize the first letter", because that hash would be radically different



https://en.wikipedia.org/wiki/Cryptographic_hash_function

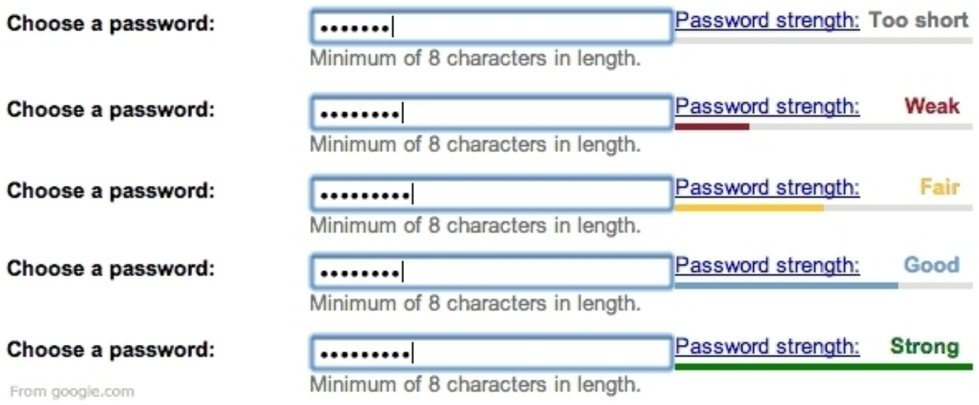
Logins and security

- But hashed passwords can still be *cracked*, or guessed
  - People build systems to randomly guess passwords over and over again
- Passwords which are easier to crack:
  - Shorter passwords
  - Passwords with fewer special characters
  - Passwords with common words/phrases, like “password” or “123”

https://en.wikipedia.org/wiki/Password_cracking

Logins and security

- Interface features like password meters help with creating less hackable passwords
- But, you're trading off password quality for user experience
  - Studies suggest that people do create better passwords, but the process takes longer and they find it more annoying

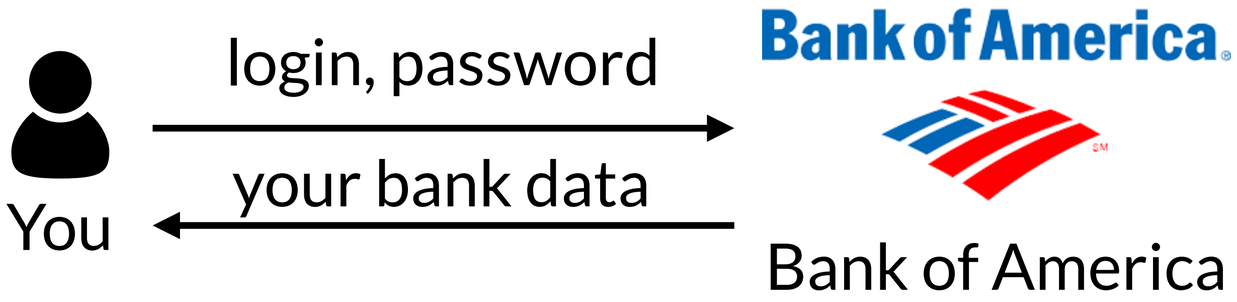


https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/ur

Third party access

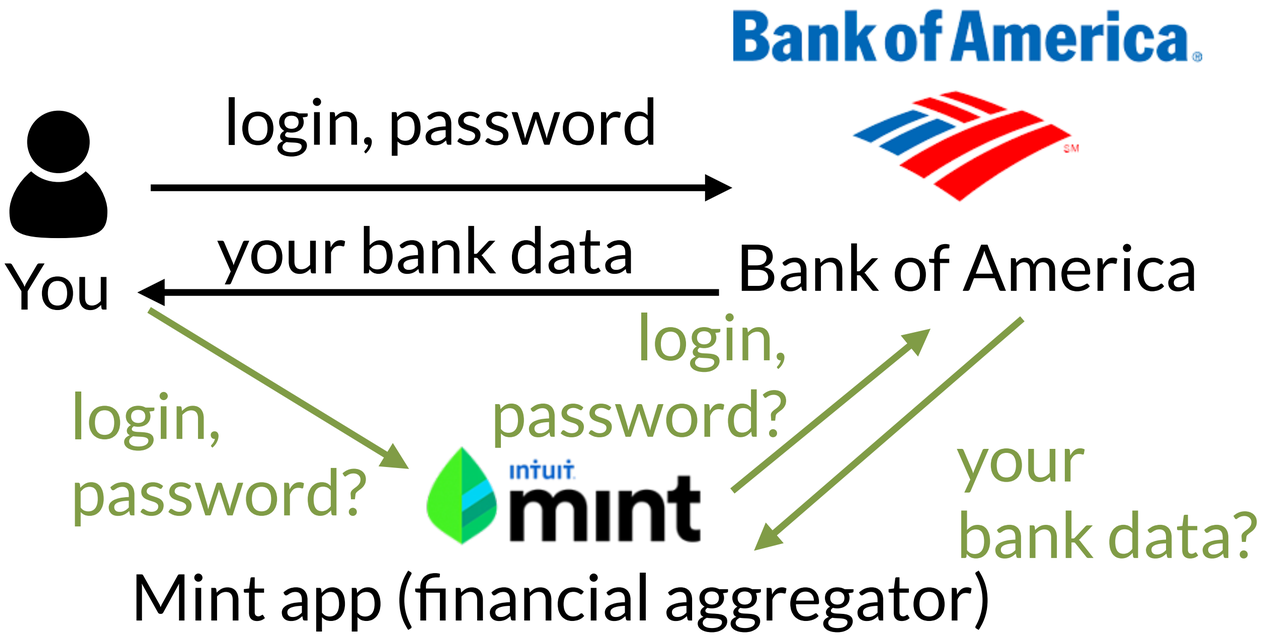
Password protocol: sending data

- Once you've logged in,
  the server can send you whatever data you're allowed to see



Sending data to a third party

- You want to send data that a server has to a third party
  - You could give them your username and password…
  - **Why is this a bad idea?**



Sending data to a third party

- Now you have to trust *another* service to manage your password
- What if you don't want them to have full access?
  - e.g., you want Mint to load your savings account but not your checking account
- What if you want to revoke access later?
  - Can change your password, but that's not a good solution

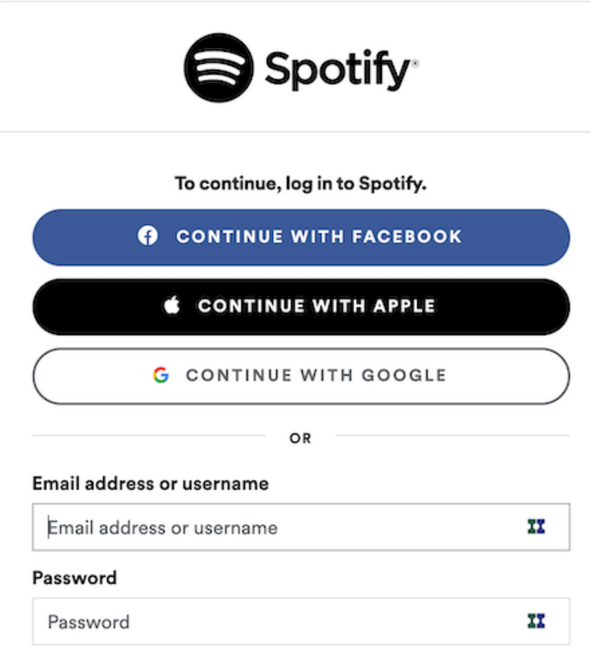
Oauth

- Open authentication, a public standard for supporting third party access
- Goal: support users in granting access to third-party applications
  - Do not require users to share their passwords with the third-party applications
  - Allow users to revoke access from the third parties at any time

https://oauth.net/2/

OpenID Connect

- Ever seen a button with "sign in with Google", etc.?
- Implemented with OpenID Connect
  - Added layer on top of Oauth

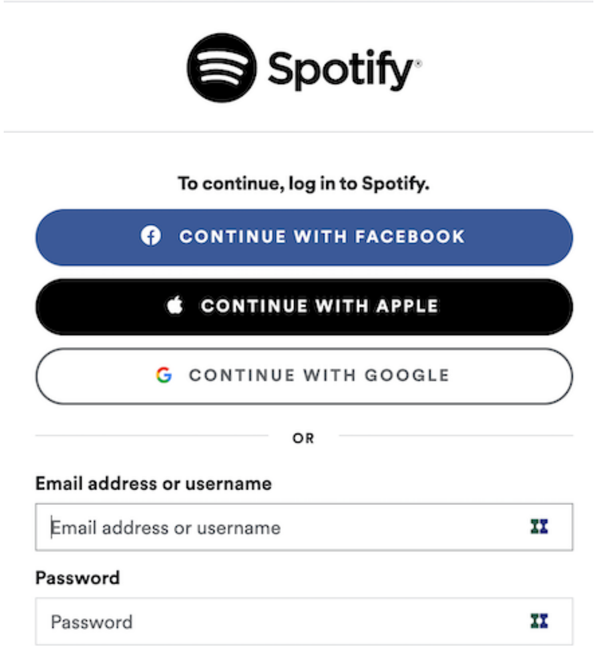


OpenID

https://openid.net/connect/

OpenID Connect

- Benefits:
  - No need to get an ID for every service
  - Only one password to remember/store
- Drawbacks
  - Facebook/Google/Apple/etc. gather (more) information about you and and the websites you go to



OpenID

Other third parties

- What about credit cards or other more-sensitive data?
  - Hashes won't work, you need the actual number to communicate with the credit card
  - When done properly, they aren't stored directly, but might store an authorization token which can be revoked
- Similar to OpenID, can rely on services like Stripe, PayPal, Visa

https://en.wikipedia.org/wiki/Payment_Card_Industry_Data_Security_Standard

Reflecting on logins

- Logins are critical to making personalized experiences in interfaces
  - But, they also introduce potential security vulnerabilities
- Overall, a lot of development effort goes into securely managing logins

Reflecting on logins

- It’s worth asking, do you need personalization at all?
- Do you need your own login scheme, or can you rely on OpenID etc.?
  - This can eliminate having to manage your own scheme
  - But for especially sensitive services (banks?), this risk might not be worth it

Today's goals

By the end of today, you should be able to…

- Articulate why logins are needed to create desired user experiences
- Differentiate authentication from authorization
- Describe the utility of supporting authentication and authorization in interfaces
- Describe the advantages and disadvantages of OpenId

IN4MATX 285: Interactive Technology Studio

Practice: Logins