COMP6443 25T1 - Week 1

Authentication

TrashPanda, 25th February 2025

What’s happening in Cyber Security?

Some research I’ve been reading

Any Questions?

https://questions.quoccacorp.com

Last Week in Review

Content

- We built a web server on sockets
- We learnt the format of a HTTP request
- We learnt a little about the HTTP methods/verbs
- We were able to use a browser to ask for information

Last Week in Review

What you should have done

- Created a CTFd account
- Set up mTLS
- Configured Burp and intercepted a request
- Written a script to solve the haas challenges
- Understood how a HTTP request and response are structured and sent
- Gone to a tutorial and met some cool people + a tutor (some tutors will fit into the group of "cool people")

Issues With Our Current Protocol

- Everyone can access everything all the time
- We don’t remember who has asked for what resources
- There’s no way to separate out resources owned by one person from that of another

Auth vs Auth

Authentication

Who you 👀

- “Are you who you say you are?”
- We are trying to verify the identity of the user.
- Response code - 401
- Response “reason phrase” - “Unauthorized”
  - Yes this is stupid.

Authorisation

Hey don’t touch that!

- Is the logged in user able to access the resource they are trying to access?
- This is something we’ll talk about next week.
- Response code - 403
- Response label - “Forbidden”

Together they form Access

from the CIA principles

- To access something you must be authenticated AND authorised

So then how do we authenticate?

Passwords!

What is a password?

I'm not going to belittle you all and explain this in detail

- It's a string that you assign on account creation and is re-entered each time to "log in" to that account

How are passwords stored?

Plaintext!

- Just put the password in the database, duh.
- Easy to check
- Nice and fast :)

How are passwords stored?

Hashing

- We “hash” the password and we store the hash in the database
  - What is a hash?
    - A one way function that deterministically produces a piece of data where the same input gives the same output.
- What are the issues with this?

How are passwords stored?

Salt

- A salt is a (hopefully randomly generated) string that gets stored alongside the hash of the password.
- When we log in, we ask for a password. That password is then appended to the salt and the entire thing is hashed:
  - H = h(salt + password)
  - We store H and we request password on login. Salts are stored in plaintext.

INTERESTING QUESTION

Should we hash on client side or hash on server side?

Attacks on Passwords

- Brute force best force
- Dictionary attack - when I take a list of common passwords and try all of them against the application
  - Works better if I have the hash and know the hashing algorithm
- Rainbow tables - I precompute every hash of whatever is in the dictionary above and just check when I leak the hashes
- Credential stuffing
- Pass the Hash attack - I steal the hash and then use that as if I'd authenticated
  - This is like stealing a session, which we'll talk about later
- Rubber hose cryptography - see next slide

Brute force

Dictionary attack

Rainbow tables

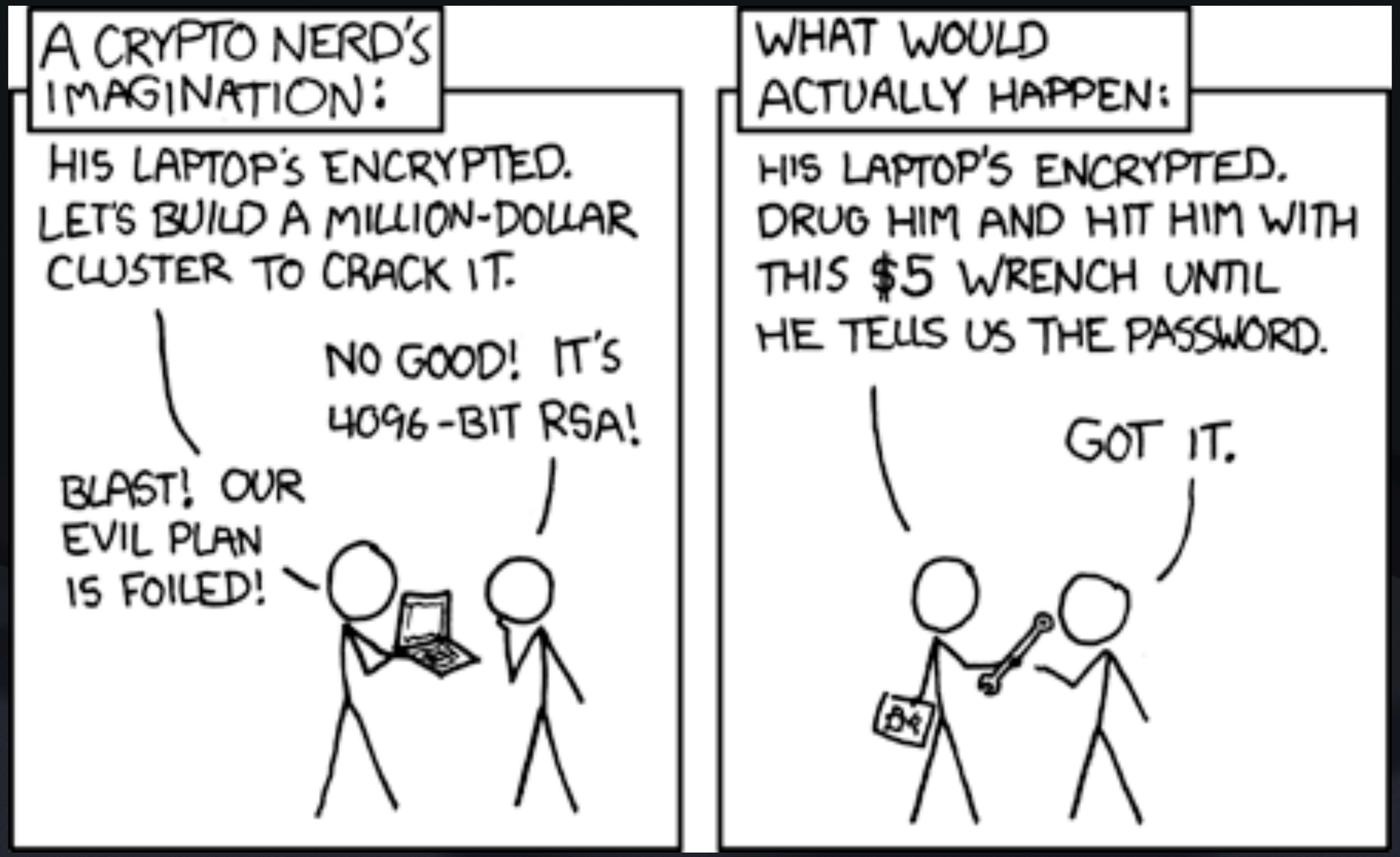
Credential stuffing

Pass the hash attack

Rubber hose cryptography

Rubber Hose Cryptography

https://xkcd.com/538



What makes a strong password?

See NIST guidelines - https://pages.nist.gov/800-63-3/sp800-63b.html

- DON'T require regular password resets (NO password expiration period)
- NO complexity requirements
- NO password hints
- NO knowledge based authentication (i.e. no secret questions to bypass passwords)
- Screen your passwords - when someone sets their password, check that it's not in an extensive list of breached passwords
- Limit failed password attempts on the application
- 8 characters minimum
- Support ALL ASCII characters
- Do NOT truncate the password when processing
- Use MFA

What is MFA?

Multi Factor Authentication

- Factors are:
  - Something you know
  - Something you have
  - Something you are
- Having multiple factors means more than one attack path
- Not perfect but better to have in case someone guesses your password
- The lines between these factors are blurry when a computer sees it.
  These are high level concepts - a computer still only sees 1s and 0s

Common MFA techniques

SMS

- Server will generate a unique code and text it to you. Then wait for a response.
- This is bad - go watch veritasium's video on why.
- Also, sometimes it'll just be like in your notifications?? So someone doesn't even need to steal your phone, they can just look while it's face up on your desk.
- Bad. Don't.
- Guess who uses it...

Common MFA techniques

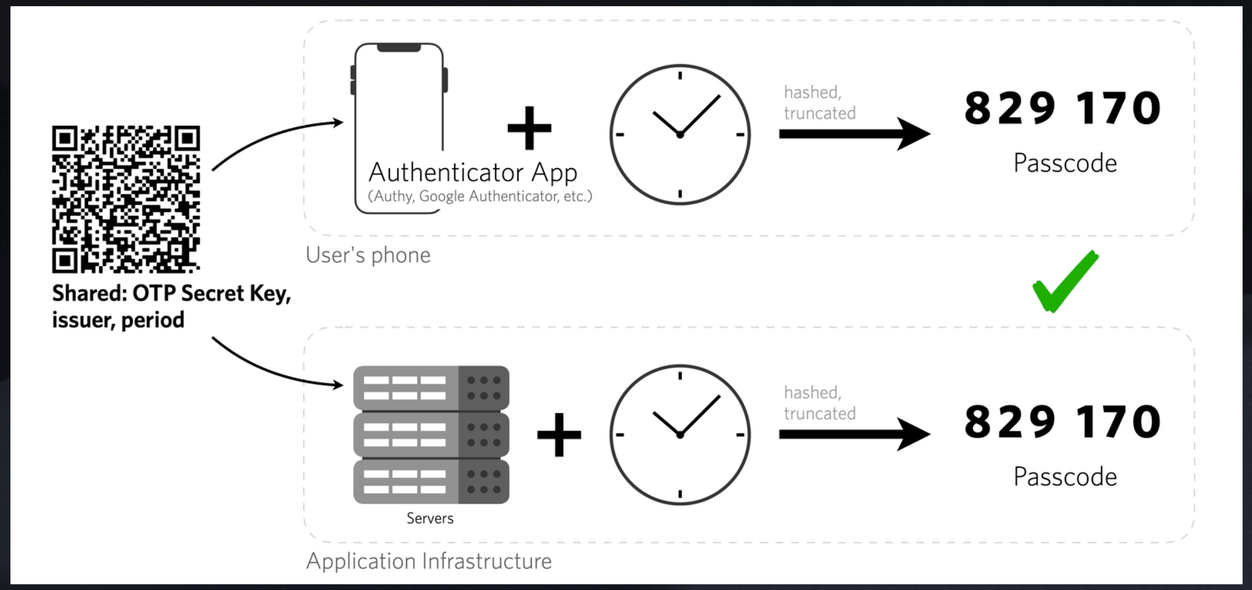
Email

- Similar to SMS
- Server will generate a unique code and email it to you. Then wait for a response.
- Similar issues with SMS verirification, but not to the same level. Lower likelihood of being compromised.
- That said, if your password is the same for your email as the app you're logging into, this becomes only 1 factor of authentication.
- Bad. Don't.

Common MFA techniques

TOTP

- Time-based One Time Password code
- What you usually do with Microsoft Authenticator or Google Authenticator or Yubico Authenticator or....
- Based on a shared secret and the time
- It works offline! - For things like internal private networks.
- https://mfa-v1.quoccacorp.com



So, after all that, what's the best way to store passwords?

Don’t.

Use SSO

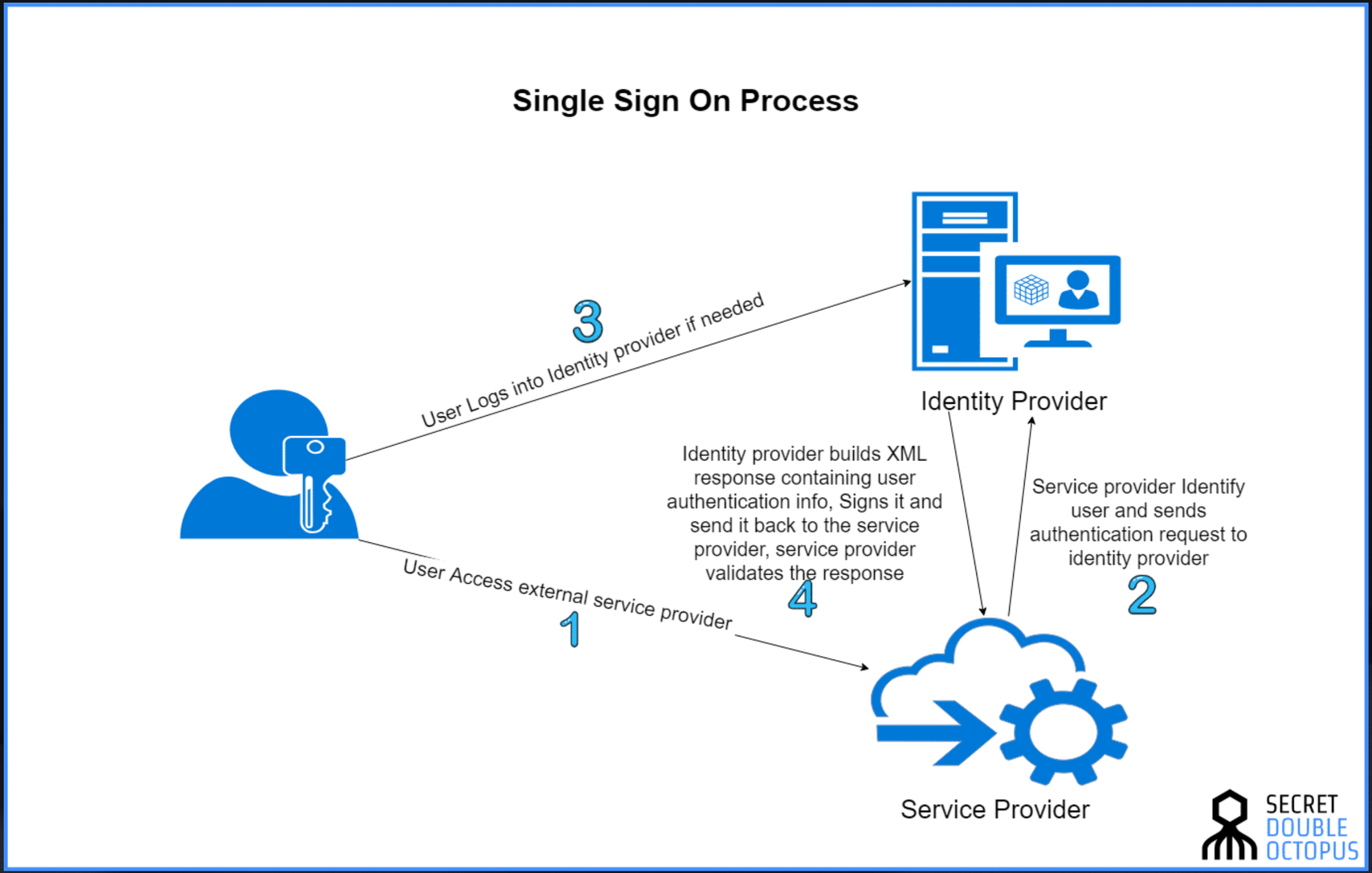
Single Sign On

- Make someone like Google or Microsoft store the passwords.
- When someone wants to log in, direct them to that service provider and get that provider to verify
- You need to trust the other provider.
- Yes I'm showing my age with the picture

SSO High Overview

This is SAML

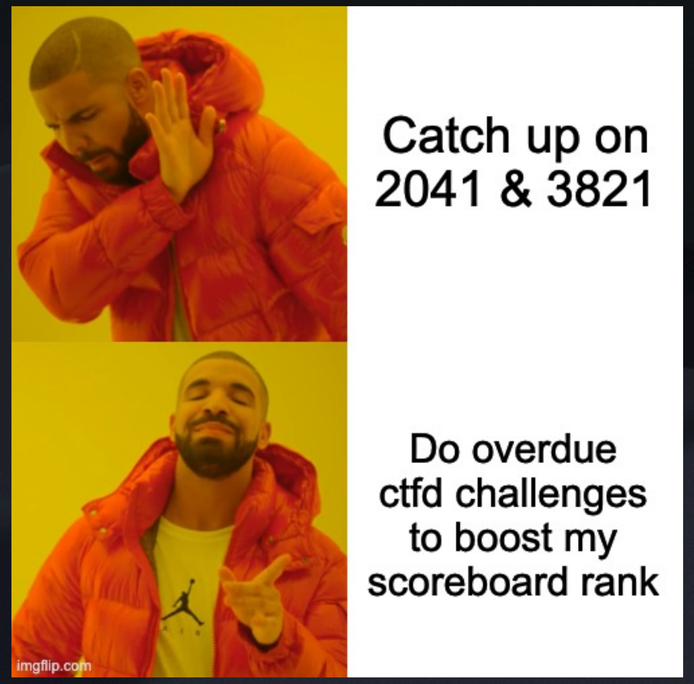
- Yunsar will talk about this in extended. The base course only needs to know that SSO exists and while it's not perfect, it's much better than explaining to your customers why you were storing plaintext passwords.



Things to Look For

Authentication Vulnerabilities

- Check default passwords lmao. admin:admin will often work. Or admin:password. Look up the service and see if there's a default password.
- Things are often built with secret keys. Just because it's secret doesn't mean it's good.
- Other programs sign their tokens (something we'll talk about next week). Make sure to check that the signature is valid.

Other Things of Note

- Use passkeys if you can. Hardware tokens can't be brute forced over the internet (in theory)
- Check if your password is leaked through haveibeenpwned.
- Make a judgement on whether a password manager is right for you. Single point of failure but often means other applications won't be compromised as you'll have stronger passwords.