Review

Protocol

- Is a set of rules
- Is a standard set of rules that different applications will follow to interact
- HTTP - reacts to whatever you send to it

DNS and IPs

If you want to find where a domain is whois lookup can get information about who registered that domain or IP

URI

Is an identifier
URN - gives a name
URL - gives the name

- The format of a URL - follows scheme, host and path as the structure

HTTP methods

- Get - requests information
- Post - asks the server to do something with what we send it

- If you see delete start playing with it in scope though
    - Deletes from database without the security checks
    - Put adds things to the database without the security checks/looking at it and generating a response - just does it

Ffuf - faster fuzzer for pentesting

Login sessions and session tokens

When you send login details - send to database for checking, does it match - then sends a response

Hashing

- This is not encryption - was something we did over in 6441
- For this course we are using Sha1 or Md5 so just bear that in mind
- There are many that we have in the world

MD5 is broken - is cryptographically broken

Ways to break hashes
- Collision attack - needed so a weak password doesn't let us log in to an account protected by a strong password
    - 2 things that form the same hash
- Preimage attack - needed so that we can't guess the password by looking at the hash
    - Don't want someone to look at a hash and work out what the password is
- Second preimage attack
- Length extension attacks

If a hash is broken it means that it is vulnerable to these types of attacks - we want someone to take longer to brute force to get into accessing your password

You should be hashing things on the server side because
- Can just match hashes
- Stealing the hash on the client side is effectively the same as stealing the password

Attacks against passwords

- Rainbow table - table of plaintext to hashes that have already been computed
    - You already have the hashes, just checking the hashes to know what the plaintext is
- Dictionary attack - spamming of plaintext passwords
    - Going through a list of things that are likely going to be passwords
- Bruteforce is going a,b,c,e,etc
    - Guaranteed to hit something using this but it just takes a long time
- Credential stuffing - you have a password for a username, throw it everywhere and see where it logs in

- Rubber Hose cryptography - idea is that you just ask people or force them to tell you to get the password

SALT - would be stored in plaintext
Username | SALT | hash

- What the server does - we get a username and password combination
    - First look up username
    - Get the salt for the user (unique thing for them - randomly generated, is added on top of the password)
    - Hash password and salt
    - Compare
- Adds a bit of complexity to login

NIST guidelines - the guidelines for how to set passwords
- Lots of organizations don't implement these

Back to logins: man in the middle
- Customer makes a request, a rogue DNS points to an attacker's server, attacker relays the requests to a real website sending their own request
    - Can steal passwords in this form
    - Can get access to people's credentials this way
- Solution to man in the middle is TLS
    - Is asymmetric encryption
    - Is a piece of mathematically brilliance
    - Can generate 2 keys - keep private key private and share your public key
    - Anything that is encrypted with public key can be decrypted with our private key
    - This is a protocol for setting up an encryption key
    - Is not perfect - but if there is a man in the middle makes things a little easier
    - But if you get the private keys this kind of isn't that useful anymore
- Mtls just means that it goes both ways
- The master key is a symmetric encryption but the exchange is asymmetric encryption

PKI
- We have certificates that are stored on the browser
- How do we verify that the certificate for the site is legit - we go up the chain and ask
    - We just trust the certificate authority
    - At some point we trust something along the chain of verifying certificates

Burpsuite can man in the middle though so what is happening there
- Burp installed its own certificate
- When we access a website - we're connecting to burp and burp is using their certificate

- So everytime you connect to a website through burp we are trusting that burp won't do anything or have anything malicious

MFA

- Multiple factors
- Factors are:
    - Something you know
    - Something you have
    - Something you are
- Having multiple factors means more than one attack path
    - You would have the same attack path to get the info for a pin and password but not for biometrics and password
    - You would need to go through multiple attack paths to get this
- Not perfect but better to have in case someone guesses your password

Session Management

- Skill set around managing cookies, sessions and when they expire
- Cookies
    - Little tokens that give some form of validations of who you are
    - They can store anything
    - Can store anything
    - Can take multiple formats
    - They are little files on your computer - they are stored somewhere on your device
- Cookie formats - plaintext, base64, jwt, python flask token
    - Base64 is enciphering - can just be deciphered
    - If you see == or = after a bunch of random letters - chuck it in a base64 decoder - might just be a username

Demo
If we want to force send

- Turn burp intercept on
- Hit submit and burp opens up asking if we want to forward it
- Here we can bypass client side protections
- Now when we go to cookies - under the lock button we get name, email info on the side
- Session tokens or cookies are ways of storing a token - if the cookie is valid you are logged in as that person
    - The server doesn't check any redirection path
    - It just checks cookie, sees the cookie is good and gives you information based on that

Http only - in the cookie means that - means that javascript does not work on this cookie

- Only way to interact with the cookie here is directly or over post requests

Secure - means that do not issue this cookie over anything that is unencrypted - HTTPS only

Cookie attacks

XSS - cross site scripting - javascript to send arbitrary code there

Man in the middle

Steal them directly - they are on your computer - stored in hidden files