Network & Web Security Study Notes

Compiled from your notes (Q&A; + simple examples).

Name: Vaibhav Saini

Core Web & Crypto Concepts

1. What is a three-way handshake?

The TCP three-way handshake establishes a connection between client and server. In other words, a three-way handshake is how two computers start talking on the internet.

SYN: Client sends SYN packet with random sequence number.
SYN-ACK: Server responds with SYN-ACK, acknowledging the client's sequence number and sending its own.
ACK: Client acknowledges the server's sequence number.

This ensures both parties are ready to communicate and agree on initial sequence numbers for reliable data transmission.

Example:
Step 1. You say hello. Your computer sends a message that means, "I want to talk."
Step 2. The other computer replies. It says, "I heard you and I am ready."
Step 3. You confirm. You say, "Great, let us start."

After these three steps, both computers agree the connection is real and ready. Then they start sending data, like web pages or messages.

This happens in TCP, the main system that moves data on the internet.

2. How do cookies work?

Cookies are small data pieces stored on the client by the server via the Set-Cookie HTTP header. The browser automatically sends cookies back to the server with subsequent requests to the same domain. They're used for session management, personalization, and tracking. Security attributes include HttpOnly (prevents JavaScript access), Secure (HTTPS only), and SameSite (CSRF protection).

Example:
1. First visit (getting your "stamp"):
- You visit Amazon.com and log in.
- Amazon says: "Cool, you're logged in! Here's a special cookie to remember you."
- Your browser saves this cookie.

2. Coming back (showing your "stamp"):
- You come back later.
- Your browser automatically sends Amazon the cookie.

- Amazon reads it and says: "Oh, it's you again! You're still logged in."

What's actually in a cookie?
It's a small piece of text that might say:
- "This is user #12345"
- "They last visited on January 11"
- "They have 3 items in their shopping cart"

Why websites use cookies:
- Remember you're logged in
- Shopping carts
- Preferences (like dark mode)

3. How do sessions work?

Sessions maintain state across HTTP requests. The server creates a unique session ID upon user login, stores session data server-side, and sends the session ID to the client (typically via a cookie). On subsequent requests, the client sends the session ID, allowing the server to retrieve associated session data. Sessions expire after timeout or logout.

4. Explain how OAuth works.

OAuth 2.0 is an authorization framework allowing third-party applications limited access to user resources without exposing credentials:

- User initiates login with third-party app
- App redirects to authorization server
- User authenticates and grants permissions
- Authorization server issues authorization code
- App exchanges code for access token
- App uses access token to access protected resources

Example:
You want to let Spotify see your Facebook friends. Instead of giving Spotify your Facebook password, you tell Facebook: "Let Spotify see my friends list, but nothing else." Facebook gives Spotify a special permission slip (token). Spotify can see your friends, but can't post as you or change your password.

5. Explain how JWT works.

JSON Web Tokens (JWT) are compact, self-contained tokens for secure information transmission.

Structure: Header.Payload.Signature
- Header: algorithm and token type
- Payload: claims (user data, expiration)
- Signature: HMAC or RSA signature verifying integrity

JWTs are stateless: the server doesn't store session data. The signature ensures tokens haven't been tampered with. Commonly used for API authentication and single sign-on.

Example:
Like a movie ticket that has your name, the movie, the time, and a special hologram that can't be faked. JWT contains info about you plus a signature that proves it's real.

6. What is a public key infrastructure flow?

PKI manages digital certificates and public-key encryption.

Certificate Authority (CA) issues a certificate:
- Certificate contains the entity's public key and identity info
- CA signs it

Verification process:
1. Entity presents certificate
2. Verifier checks CA signature
3. Validates certificate status (CRL/OCSP)
4. Confirms identity and validity period

Analogical example:
Like a school ID card with the principal's signature: people trust it because the principal signed it.

7. Difference between symmetric and asymmetric encryption.

Symmetric: same key for encryption and decryption. Fast and efficient for large data (AES, DES, 3DES). Main challenge is secure key distribution.

Asymmetric: uses a public-private key pair. Public key encrypts, private key decrypts. Slower but solves key distribution (RSA, ECC). Used for key exchange and digital signatures.

Hybrid approach: use asymmetric to exchange a symmetric key, then use symmetric for bulk data.

Simple examples:
- Symmetric = one house key locks and unlocks.
- Asymmetric = mailbox: anyone can drop mail in (public key), only you can open it (private key).

8. Describe SSL/TLS handshake.

SSL/TLS handshake establishes a secure connection:

- ClientHello: supported cipher suites, TLS version
- ServerHello: selected cipher, certificate
- Certificate verification: client validates server certificate
- Key exchange: client and server establish shared secrets
- Session keys generated
- Finished messages: both verify handshake

Example:
Two kids agree on a secret way to talk, verify it, then use a shared secret code for the conversation.

9. How does HMAC work?

HMAC (Hash-based Message Authentication Code) provides message integrity and authenticity.

```
HMAC(K, m) = H((K' XOR opad) || H((K' XOR ipad) || m))

Where:
- K is secret key
- m is message
- H is hash function
- opad/ipad are padding constants
```

It combines a secret key with the message before hashing, preventing length extension attacks that can affect simple hash(key || message) constructions.

10. Why HMAC is designed that way?

HMAC's design addresses security concerns:
- Nested hashing helps prevent length extension attacks
- Key padding ensures the key influences the whole hash process
- Two-pass construction (inner and outer hash) adds safety
- Separates key and message to resist certain cryptanalytic attacks

Example:
Like a special double-seal technique that is hard to fake or move without breaking.

11. Authentication vs authorization?

Authentication = "Who are you?" (verifying identity).
Authorization = "What can you do?" (permissions after identity is known).

Example:
You log into Netflix (authentication). Your profile can only watch kids shows (authorization).

12. Diffie-Hellman vs RSA?

Diffie-Hellman: key exchange protocol. Two parties establish a shared secret over an insecure channel without sending the secret itself. Often provides forward secrecy (with ephemeral DH).

RSA: asymmetric encryption and digital signatures. Can encrypt data and sign messages. Does not automatically provide forward secrecy unless used in a way that supports it.

DH is mainly for key agreement; RSA is for encryption and signatures.

13. How does Kerberos work?

Kerberos is a network authentication protocol using tickets:
- AS request: client requests Ticket Granting Ticket (TGT)
- AS response: receives TGT encrypted with user's password hash
- TGS request: client presents TGT, requests service ticket
- TGS response: receives service ticket
- AP request: presents service ticket to application server
- Access granted: server validates ticket

Uses symmetric encryption, time-based tickets, and mutual authentication. Common in Active Directory.

14. Compress vs encrypt - which first and why?

Compress first, then encrypt.
- Compression finds patterns; encryption removes patterns
- Encrypted data looks random and doesn't compress well
- Compressing encrypted data wastes resources

Example:
Fold clothes first (compress), then lock the suitcase (encrypt).

15. How do I authenticate you and know you sent the message?

Use digital signatures:
- Sender hashes the message
- Encrypts the hash with their private key (signature)
- Sends message + signature
- Receiver decrypts signature with sender's public key
- Receiver hashes the received message and compares

If they match, the message is authentic and unmodified. You also need a verified public key (via PKI/certificates).

16. Should you encrypt all data at rest?

Not necessarily all data. Encrypt when:
- PII, financial data, health records
- passwords, API keys, secrets
- regulated or sensitive business data

You may skip for:
- public information
- non-sensitive logs
- cases where performance/cost outweighs risk (still evaluate carefully)

Balance security needs with performance impact, key management, and compliance.

17. What is Perfect Forward Secrecy (PFS)?

PFS ensures past session keys stay safe even if the server's private key is compromised later. Achieved using ephemeral key exchange (DHE/ECDHE). Each session uses a unique key that isn't stored, so recorded traffic can't be decrypted later with a stolen long-term key.

Network Level and Logging

Common ports involving security: risks and mitigations

Common ports:
- 20/21 (FTP): unencrypted; use SFTP/FTPS
- 22 (SSH): brute force; use keys, rate limiting, fail2ban
- 23 (Telnet): unencrypted; replace with SSH
- 25 (SMTP): spam relay; require auth, use 587 for submission
- 53 (DNS): tunneling, amplification; use DNSSEC, rate limiting
- 80 (HTTP): unencrypted; redirect to HTTPS
- 443 (HTTPS): still app attacks; use WAF, proper TLS config
- 3306 (MySQL): exposed DB; bind to localhost/private, firewall
- 3389 (RDP): brute force; use VPN, strong passwords, NLA
- 8080/8443 (alt HTTP/HTTPS): often forgotten; monitor like 80/443

DNS port:
Port 53 for DNS, using both UDP (queries) and TCP (zone transfers, large responses).

Describe HTTPS and how it is used.

HTTPS is HTTP over TLS/SSL and provides:
- Encryption (confidentiality)
- Authentication (server identity via certificates)
- Integrity (prevents tampering)

Used to protect credentials, financial transactions, and sensitive data in transit.

Difference between HTTPS and SSL.

HTTPS is a protocol (HTTP over TLS/SSL).
SSL is an old/deprecated encryption protocol. Modern HTTPS uses TLS (TLS 1.2 and TLS 1.3). People still say "SSL" but usually mean TLS.

How does threat modeling work?

Threat modeling is a systematic way to identify and reduce risk:
1. Identify assets
2. Identify threats (e.g., STRIDE: Spoofing, Tampering, Repudiation, Information Disclosure, DoS, Elevation of Privilege)
3. Identify vulnerabilities
4. Assess risk (likelihood x impact)
5. Define mitigations
6. Validate and repeat (iterative)

Frameworks: STRIDE, PASTA, DREAD, Attack Trees.

What is a subnet and how is it useful in security?

A subnet is a logical subdivision of an IP network. Security benefits:
- Segmentation (isolate sensitive systems)

- Smaller blast radius
- Access control between subnets (firewalls, ACLs)
- Easier monitoring
- Compliance separation (DMZ vs app vs DB vs management).

What is a subnet mask?

A subnet mask determines which part of an IP address is the network and which part is the host.
Example: 192.168.1.0/24 with mask 255.255.255.0.
Network: 192.168.1
Hosts: 1-254 usable.

Explain what traceroute is.

Traceroute maps the path packets take using TTL (Time To Live). It sends packets with increasing TTL:
- TTL=1 expires at first router (ICMP Time Exceeded)
- TTL=2 expires at second router
- continues until destination reached

Shows each hop and latency. Useful for diagnosing routing issues.

Explain TCP/IP concepts.

TCP/IP is the protocol suite for internet communication:
- IP: addressing and routing packets
- TCP: reliable, ordered, error-checked delivery
- UDP: connectionless, faster, no guarantees
- ICMP: errors and diagnostics

Common 4 layers: Network Access, Internet, Transport, Application.

What is OSI model?

Seven layers:
1. Physical
2. Data Link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

Mnemonic: Please Do Not Throw Sausage Pizza Away.

Router vs switch?

Router (Layer 3): routes between networks using IP, uses routing tables, connects LANs to WANs, often does NAT/firewalling.
Switch (Layer 2): connects devices within a network using MAC addresses, forwards frames based on MAC table, fast local traffic.

TCP vs UDP - which is more secure and why?

Neither is inherently "more secure." Security depends on what runs on top (TLS, app auth, etc.).
TCP has connection tracking which can reduce some spoofing. UDP is easier to spoof, but can still be secured (e.g., DTLS, app-layer checks).

IPsec Phase 1 vs Phase 2?

Phase 1 (IKE SA): builds a secure authenticated channel to negotiate.
- chooses crypto algorithms
- authenticates peers
- establishes IKE Security Association

Phase 2 (IPsec SA): negotiates the actual tunnel parameters and protected traffic.
Phase 1 protects Phase 2 negotiation; Phase 2 protects data.

Biggest AWS security vulnerabilities?

Common issues:
- Misconfigured S3 buckets (public access)
- Overly permissive IAM policies
- Exposed access keys (in repos)
- Missing encryption at rest/in transit
- Open security groups (0.0.0.0/0)
- No MFA
- No logging/monitoring (CloudTrail/GuardDuty off)
- Unpatched instances
- Public snapshots (EBS/RDS)
- Shared responsibility misunderstandings

How do web certificates for HTTPS work?

Certificates (X.509) prove server identity:
- CA issues cert after verification
- Cert includes domain, public key, validity, CA signature
- Browser checks trust chain, expiry, domain match, revocation (CRL/OCSP)
- Prevents MITM by verifying identity before encryption.

Purpose of TLS?

TLS provides:
- Confidentiality (encryption)
- Integrity (MAC)
- Authentication (certificates, optionally client certs)

Used for HTTPS, email, VPNs, and more.

Is ARP UDP or TCP?

Neither. ARP is Layer 2 (Data Link) and maps IP addresses to MAC addresses on the local network.

What information is added at each OSI layer?

Encapsulation example:
- L7 Application: app data (HTTP request)
- L6 Presentation: encoding/encryption/compression metadata
- L5 Session: session identifiers
- L4 Transport: ports, sequence numbers, checksums (TCP/UDP headers)
- L3 Network: IP addresses, TTL, protocol (IP header)
- L2 Data Link: MAC addresses, frame check sequence (Ethernet header/trailer)
- L1 Physical: bits/signals on the wire

What is a firewall? How does it work? In cloud?

A firewall controls traffic based on rules.
- Packet filtering: checks IP/port/protocol
- Stateful inspection: tracks connections
- App-layer firewalls: deep inspection

Cloud examples:
- AWS Security Groups (stateful, instance-level)
- Network ACLs (stateless, subnet-level)
- WAF (app-layer, OWASP Top 10 protection)
- Virtual appliances (traditional firewalls as VMs)

Difference between IDS and IPS?

IDS: detects and alerts; does not block (out-of-band).
IPS: inline; can block/drop traffic; risk of false positives blocking legit traffic.

How do you harden a system?

Common hardening steps:
- remove unnecessary services/apps
- patch regularly
- strong authentication (MFA, SSH keys)
- least privilege
- firewall rules
- remove/disable default accounts
- audit logging
- encryption
- follow baselines (CIS, STIG)
- vulnerability scanning and monitoring
- network segmentation

What would you do if you discovered an infected host?

Incident response:
- contain/isolate host
- identify scope and malware type
- preserve evidence (memory/disk images)
- eradicate (remove malware / rebuild)
- recover from clean backup

- monitor for reinfection
- document and do lessons learned

Approach to memory dump analysis?

Memory forensics steps:
- confirm OS/profile
- list processes and suspicious parent/child
- network connections and listening ports
- check injection/hollowing/DLLs
- persistence artifacts
- extract/dump suspicious processes
- correlate with disk/logs

Tools: Volatility, Rekall, Redline, YARA rules

How would you detect a DDoS attack?

Indicators:
- sudden traffic spike
- geographic anomalies
- high latency/timeouts
- resource exhaustion (CPU/memory/bandwidth)
- many failed connections (SYN floods)
- repeated patterns (same user agents/requests)

Use baselines + monitoring tools (NetFlow/sFlow, WAF/CDN alerts, SIEM).

How does the kernel know which function to call for the user?

Through system calls:
- program invokes a syscall (via libc wrapper)
- syscall number placed in a register
- CPU switches to kernel mode via syscall instruction/interrupt
- kernel uses syscall table to map number to function pointer
- executes and returns to user mode with result

Reverse-engineering a custom protocol packet?

Approach:
- capture traffic (Wireshark/tcpdump)
- find fixed vs variable fields, magic bytes, length fields
- compare multiple samples
- guess header/payload structure, checksums
- test hypotheses by modifying packets and observing responses
- detect encryption via high entropy
- document protocol flow/state machine

Tools: Wireshark, scapy, 010 Editor

Differentiate XSS from CSRF.

XSS: injects malicious script into a page; runs in victim browser; can steal cookies/session tokens. Mitigate with output encoding, input validation, CSP.

CSRF: tricks an authenticated user into making unwanted state-changing requests. Mitigate with CSRF tokens, SameSite cookies, Origin/Referer checks.

What is SSRF?

Server-Side Request Forgery: attacker makes the server send requests to unintended locations (internal services, cloud metadata like 169.254.169.254, etc.).
Mitigation: allowlist URLs, block private IP ranges, disable risky protocols, validate/sanitize URLs, segment networks.

Padlock icon in browser - how is it generated?

Browser shows the lock when:
- TLS handshake succeeds
- certificate validates (trusted CA chain, not expired, domain match, not revoked)
- page is loaded without blocked mixed content (or appropriate secure indicators)

What is Same Origin Policy and CORS?

SOP: browser rule that scripts from one origin cannot access data from another origin.
Origin = scheme + domain + port.

CORS: server-controlled way to relax SOP using headers like:
- Access-Control-Allow-Origin
- Access-Control-Allow-Methods
- Access-Control-Allow-Headers

Preflight OPTIONS used for complex requests.

Secure a MongoDB database?

Key steps:
- enable authentication and RBAC
- use TLS for connections
- encrypt at rest
- bind to localhost/private interfaces; firewall
- patch regularly
- enable audit logs
- backups (encrypted, tested restores)
- disable unused features (like server-side JS if not needed)

Secure PostgreSQL?

Key steps:
- use scram-sha-256 (or md5) rather than trust
- strict pg_hba.conf rules
- least privilege roles, avoid superuser
- require TLS
- listen only where needed + firewall

- patch regularly
- audit logging (pgaudit)
- optional row-level security
- encrypted backups, test restore

DB exfiltrated: SHA256 (single round) with static salt - what now?

Risk: high. Single-round SHA256 is fast to crack, and a static salt is weak.
Immediate actions:
- force password reset for affected users
- monitor for credential stuffing
- investigate and close the breach vector, review logs
Changes:
- move to bcrypt/scrypt/Argon2 with per-user random salts
- use key stretching (iterations)
- consider adding a pepper (app-level secret)
- implement MFA and improve detection/response