Day 2

Exploitation Analyst

What is Network Security Control?

Network security control refers to the technical and administrative safeguards implemented to protect the integrity, confidentiality, and availability of data and network resources. These controls help prevent, detect, and respond to threats targeting network infrastructure.

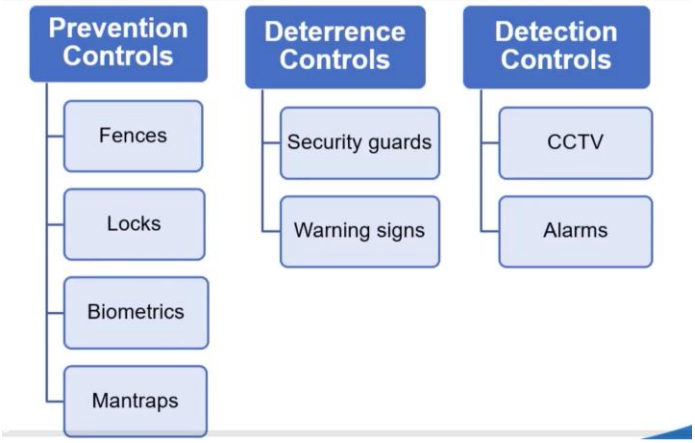
They include firewalls, intrusion detection systems (IDS), access control lists, VPNs, encryption, and network segmentation. Network security controls are tools and rules to protect networks from unauthorized access and threats. They form the core of defensive strategies in securing network communications and systems.

Network security controls are categorized into three main types to address different aspects of protection:

- **Physical Controls**: Protect the physical infrastructure (e.g., locked server rooms, CCTV, biometric access).
- **Technical Controls**: Use technology to protect systems (e.g., firewalls, encryption, IDS/IPS).
- **Administrative Controls**: Define policies and procedures (e.g., security training, access management policies, incident response plans).

Physical Security Controls

These secure the physical components of the network like servers, routers, and data centers. Examples include surveillance cameras, biometric locks, and restricted facility access.



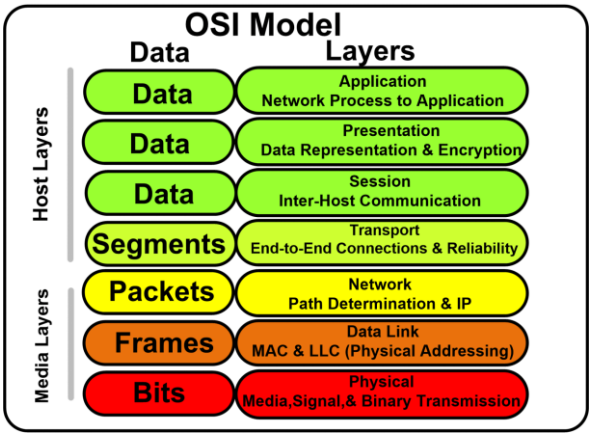
Technical Security Controls

These are software or hardware tools that enforce security policies and protect data flow. Examples include firewalls, intrusion detection systems, and data encryption.

Administrative Security Controls

These involve policies, procedures, and training to guide secure user behaviour and system usage. Examples include access control policies, security awareness programs, and incident response plans.

What is OSI Model?



Layer	Function (One Line)	Important Info (Interview)
7. Application	Interfaces directly with user applications for network services.	Protocols: HTTP, FTP, SMTP; user-level interaction.
6. Presentation	Translates, encrypts, or compresses data for the application layer.	Handles data format (e.g., JPEG, SSL/TLS).
5. Session	Manages sessions and controls dialog between devices.	Maintains, establishes, and terminates sessions (e.g., NetBIOS, RPC).
4. Transport	Ensures reliable data transfer with error checking and flow control.	TCP (reliable), UDP (faster); port numbers (e.g., 80, 443).
3. Network	Routes data packets between different networks.	IP addressing, routing, devices like routers; protocols: IP, ICMP.
2. Data Link	Provides node-to-node data transfer and handles MAC addressing.	MAC addresses, switches, frames, protocols like ARP, Ethernet.
1. Physical	Transmits raw bits over physical media like cables or radio.	Cables, NICs, hubs; bit-level transmission (0s and 1s).

What is Network Security Protocols?

Network security protocols are standardized rules and procedures designed to secure data communication over networks. They ensure confidentiality, integrity, authentication, and secure transmission between devices.

Examples include SSL/TLS (secure web traffic), IPsec (secure IP communication), HTTPS (secure HTTP), SSH (secure remote login), and WPA2 (secure wireless communication).

Protocol	Purpose / Use Case
SSL/TLS	Secures web traffic (HTTPS), email, and VoIP by encrypting data in transit.
IPSec	Provides secure IP communication through tunneling, encryption, and authentication.
HTTPS	Secure version of HTTP; encrypts website communication using SSL/TLS.
SSH	Secure remote login and command execution on remote systems.
Kerberos	Ticket-based authentication used in enterprise environments (e.g., Active Directory).
SNMPv3	Secure network device monitoring and management with encryption and authentication.
WPA2/WPA3	Encrypts wireless communication for Wi-Fi networks; WPA3 is the latest standard.
RADIUS	Centralized authentication system for remote users and devices.

Why so many different types of protocols exist?

Different network security protocols exist because each one is designed to address specific security needs across different layers, systems, and use cases in a network environment. No single protocol can handle all aspects of security efficiently.

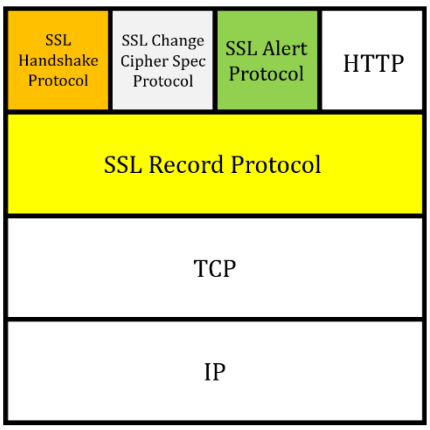
SSL – Secure Socket Layer:

SSL is a network security protocol that ensures confidentiality, integrity, and authentication between a client and a server. It operates between the Application layer (Layer 7) and the Transport layer (Layer 4) in the OSI model.

How SSL Works:

When the application layer sends data, SSL encrypts it and adds a secure header before handing it to the transport layer. On the receiving side, the header is removed, the data is decrypted, and then passed up to the application layer securely.

SSL is not a single operation but a suite of protocols working together to ensure secure data transmission. Which protocols?



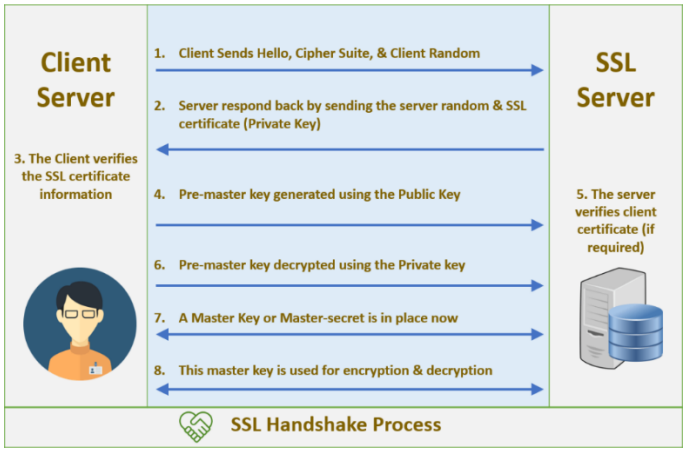
Key protocols within SSL include:

- **Handshake Protocol**: Establishes a secure connection, negotiates cryptographic algorithms, and exchanges keys. [important] (it ensures authenticity)
- **Record Protocol**: Handles data encryption, integrity checks, and secure transmission of messages. [important] (it ensures confidentiality, integrity)
- **Alert Protocol**: Sends error or warning messages during the session (e.g., handshake failure). (it handles any error messages)
- **Change Cipher Spec Protocol**: Signals the transition to a newly agreed set of cryptographic parameters. (it changes the pending state to the current state)

SSL handshake protocol:

Step-by-Step in Easy Words:

1. **Client Hello**: The client sends a message saying “Hello, I want to talk securely,” and shares supported encryption methods and a random number.
2. **Server Hello**: The server replies, “Okay, let’s use this encryption method,” and sends its digital certificate (which contains its public key) and another random number.
3. **Certificate Verification**: The client checks if the server’s certificate is valid and trustworthy (e.g., issued by a trusted Certificate Authority).
4. **Session Key Generation**: The client creates a unique **session key** (used for actual encryption) and encrypts it using the server’s public key.
5. **Key Exchange**: The encrypted session key is sent to the server, which then decrypts it using its private key.
6. **Secure Communication Begins**: Both now have the same session key and start securely exchanging data.



SSL record protocol:

The SSL Record Protocol is responsible for securely packaging and transmitting data after the SSL Handshake is complete. It ensures that the actual communication (like sending web content, login info, etc.) remains confidential and tamper-proof.

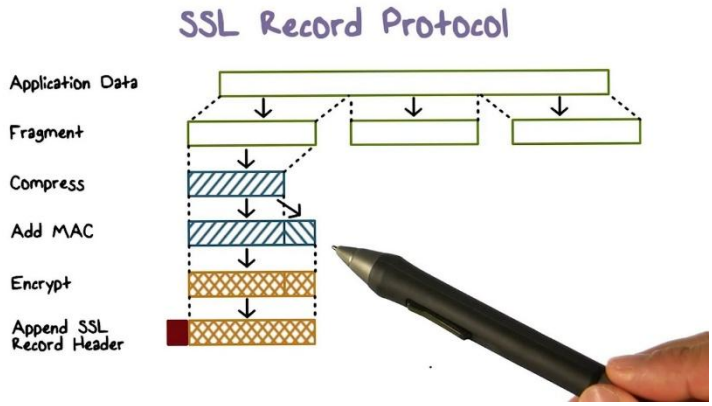
In Simple Words:

Once the handshake is done and both sides have a shared session key, the Record Protocol takes over to protect application data being sent.

Here's what it does:

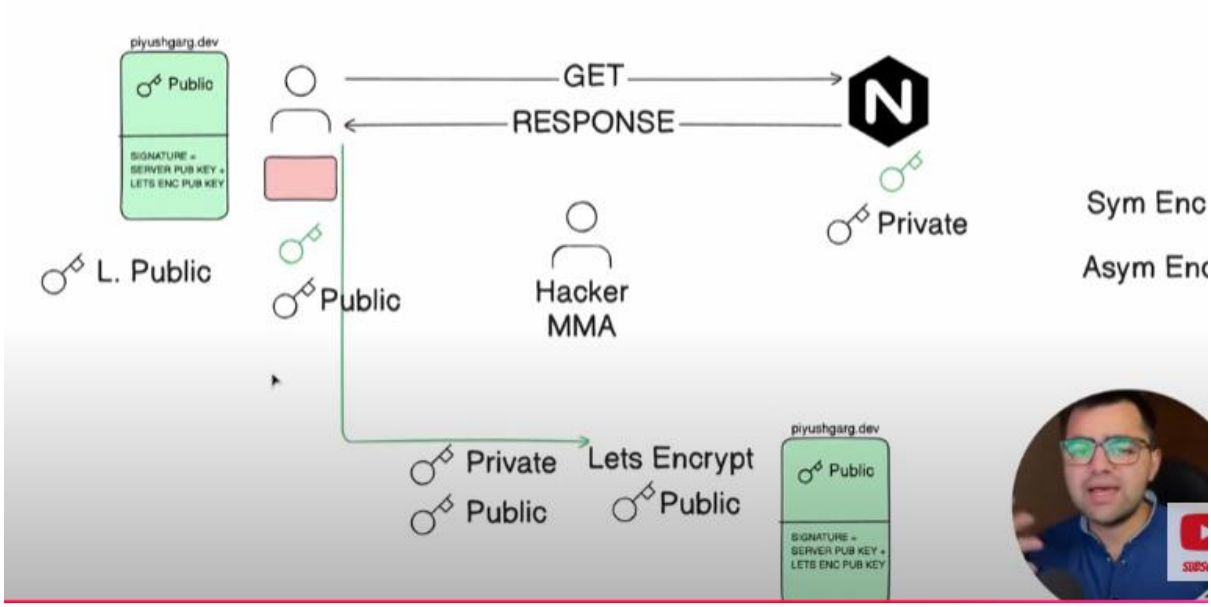
1. **Fragmentation**: Breaks large data into manageable blocks.
2. **Compression** *(optional)*: Compresses the data to reduce size.
3. **MAC (Message Authentication Code)**: Adds a code to verify the data hasn't been altered.
4. **Encryption**: Encrypts the data and the MAC using the session key.
5. **Header Addition**: Attaches a small header indicating what type of data it is and its length.
6. **Transmission**: Sends the encrypted packet to the receiver.

On the receiving end, the process is reversed — the record is decrypted, MAC is verified, and the data is reassembled and passed to the application.



How Digital Certificates Work in SSL:

1. Certificate Creation by the Server and CA:
    - The server generates a public-private key pair.
    - It sends its public key and domain name to a Certificate Authority (CA).
    - The CA verifies the server's identity and then creates a digital certificate, which includes the server's public key and domain name, digitally signed using the CA's private key.
2. What's Inside the Certificate?
    - Server's public key
    - Domain name
    - Certificate expiration date
    - Signature created by the CA: Sign(Hash of certificate info) using CA's private key
3. Handshake and Verification by Client:
    - The server sends its certificate to the client during the SSL handshake.
    - The client already has a list of trusted CA public keys (built into browsers/OS).
    - The client uses the CA's public key to verify the CA's signature on the certificate.
        - This proves the certificate wasn't forged or tampered with.
    - If valid, the client trusts the server's public key from the certificate.
4. Secure Key Exchange:
    - The client then generates a symmetric session key, encrypts it using the server's public key, and sends it to the server.
    - The server decrypts it using its private key. Now both share the same symmetric key.



Advantages of SSL Certificates

1. **Encryption**: SSL ensures all data transferred between client and server is encrypted, protecting it from eavesdropping.
2. **Authentication**: SSL certificates verify the identity of a website, helping users trust that they're communicating with the legitimate server.
3. **Data Integrity**: Ensures data isn't tampered with during transmission.
4. **User Trust & SEO**: Browsers show a padlock for SSL-enabled sites, increasing trust, and Google ranks HTTPS sites higher.
5. **Compliance**: Required for PCI DSS, GDPR, HIPAA, and other data protection regulations.

Disadvantages of SSL Certificates

1. **Cost**: Certificates from trusted CAs can be expensive, especially for wildcard or EV certificates.
2. **Performance Overhead**: SSL encryption/decryption can slightly impact server performance.
3. **Maintenance**: Certificates expire and need renewal, requiring proper management to avoid downtime.
4. **False Sense of Security**: SSL only secures data in transit, not the website's internal security or storage.

What is the Self-signed SSL certificate?

A self-signed certificate is an SSL certificate that is signed by the server itself, not by a trusted Certificate Authority (CA). It provides encryption but does not guarantee trust, because the identity is not verified by an external authority.

Steps to Create a Self-Signed Certificate

1. Generate Key Pair
    a. The server generates a public-private key pair.
2. Create Certificate Signing Request (CSR) *(optional)*
    a. The server fills in details like domain name, organization, and public key.
    b. Usually used when submitting to a CA, but can also be used for self-signing.
3. Self-Sign the Certificate
    a. The server uses its own private key to sign the certificate.
    b. This creates a digital signature over the certificate content.
4. Install the Certificate
    a. The self-signed certificate is installed on the server for SSL communication.
    b. The server sends this certificate to clients during the SSL handshake.

Advantages of SSL Self-Signed Certificates

1. **Free**: No cost involved since you don't need a third-party Certificate Authority (CA).
2. **Quick Setup**: Can be generated instantly without waiting for CA validation.
3. **Useful for Internal Use**: Suitable for development, testing, or internal tools where public trust isn't needed.
4. **Full Control**: You manage the entire certificate lifecycle yourself.

Disadvantages of SSL Self-Signed Certificates

1. **Not Trusted by Browsers**: Browsers will show warnings since no CA has verified the identity.
2. **Vulnerable to MITM Attacks**: Without trusted verification, attackers can impersonate the server.
3. **Poor User Experience**: Users may avoid sites with security warnings.
4. **No Identity Validation**: There's no third-party assurance that the server is legitimate.

How SSL Certificates Can Be Bypassed by a Hacker

While SSL is designed to secure communication, attackers can **bypass or undermine it** through certain **tactics** — mostly when users or systems fail to verify the certificate properly.

Common SSL Bypass Techniques:

1. Man-in-the-Middle (MITM) with Fake Certificate
    a. The attacker intercepts traffic and presents a fake SSL certificate to the client.
    b. If the user ignores browser warnings, the attacker can decrypt, read, or alter data.
2. Compromised Certificate Authority (CA)
    a. If a CA is hacked or malicious, it can issue fake but trusted certificates, enabling MITM attacks that go undetected.
3. Using Self-Signed Certificates
    a. Attackers set up a fake site with a self-signed certificate. If the client accepts it blindly, the connection can be hijacked.
4. SSL Stripping Attack
    a. The attacker downgrades the HTTPS connection to HTTP, tricking the user into sending data over an insecure channel.
5. Client Misconfiguration
    a. If the system or application doesn't properly verify certificates, attackers can exploit this by presenting invalid or spoofed certificates.
6. Phishing with Lookalike Domains
    a. A malicious site uses a valid SSL certificate but on a domain name that looks similar to a trusted one (e.g., go0gle.com), tricking users.

How SSL Certificates Work

SSL certificates ensure that information passed between systems can never be read or intercepted.

1	A browser tries to connect to a website that is SSL-secured.	
2	The web server's identification is requested by the browser.	
3	The web server delivers a copy of its SSL certificate to the browser.	
4	The browser verifies if it trusts the SSL certificate or not. If it does, the web server receives a signal.	
5	A digitally signed grant is then returned by the web server to initiate an SSL encrypted session.	
6	The web server and the browser or server exchange encrypted data.