Transport-Layer Security Comprehensive Exam Notes

Table of Contents

TCP/UDP Protocols

Core Concepts

Port Numbers: Used to identify specific applications on computers. Each application gets a unique port number that acts as an address within the computer.

Maximum Transmission Unit (MTU): Maximum size allowed for data packets. If data exceeds MTU, it's broken into smaller segments.

TCP (Transmission Control Protocol)

Key Features

- **Connection-oriented**: Requires handshake before data transmission
- **Reliable delivery**: Uses checksums and acknowledgments (ACKs)
- **Ordered delivery**: Sequence numbers ensure proper ordering
- **Flow control**: Sliding window protocol manages transmission rate

TCP Segment Structure

- **Sequence Number (32 bits)**: Index of first byte in segment
- **Acknowledgment Number (32 bits)**: Next expected byte from sender
- **Flags (6 bits)**: Control flags (SYN, ACK, FIN, etc.)

Three-Way Handshake

1. Client → Server: SYN
2. Server → Client: SYN-ACK
3. Client → Server: ACK

TCP Components

1. **Handshake**: Establishes connection parameters
2. **Data Flow**: Manages reliable data transfer
3. **Termination**: Cleanly ends connection

UDP (User Datagram Protocol)

Key Features

- **Connectionless**: No handshake required
- **Fast and efficient**: Low latency, minimal overhead
- **No reliability guarantees**: No ACKs or retransmissions
- **Broadcast/Multicast support**: Single packet to multiple recipients

UDP Segment Structure

- **Source Port (16 bits)**: Sender's port number
- **Destination Port (16 bits)**: Receiver's port number
- **Length (16 bits)**: Total segment length
- **Checksum (16 bits)**: Error checking (no recovery)

Attack Vectors

TCP SYN Flooding

- **Attack**: Attacker sends many SYN requests but never completes handshake
- **Impact**: Server resources exhausted by half-open connections
- **Mitigation**:
  - SYN Cookies: Encode information in SYN-ACK, allocate resources only after ACK
  - TCP Filtering: Drop suspicious SYN packets

UDP Amplification Attack

- **Process**:
  1. Attacker sends small UDP packets to servers
  2. Spoofs victim's IP address
  3. Servers respond with larger payloads to victim
  4. Victim overwhelmed with amplified traffic

- **Example**: Memcached attack (2018) - 1.7 Tbps peak, 50,000x amplification
- **Mitigation**:
  - Server-side: Rate limiting, response filtering, disable unnecessary UDP services
  - Victim-side: Firewalls, limit UDP communication

TLS Security Protocol

Background

- **Purpose**: Encrypts TCP payload to ensure confidentiality, integrity, and authenticity
- **Evolution**: SSL 1.0 → SSL 2.0 (1995) → SSL 3.0 (1996) → TLS 1.0 (1999) → TLS 1.1 (2006) → TLS 1.2 (2008) → TLS 1.3 (2018)

TLS Architecture

Four Main Protocols

1. **Handshake Protocol**: Establishes secure connection, negotiates algorithms, exchanges keys
2. **Record Protocol**: Provides confidentiality and integrity through encryption
3. **Alert Protocol**: Communicates errors and warnings
4. **Change Cipher Spec Protocol**: Signals transition to new cipher suite

Record Protocol

Header Structure

- **Content Type**: Type of payload (Handshake, Application Data, Alert)
- **Version**: TLS protocol version
- **Length**: Data length in record

Processing Steps (After Handshake)

1. Fragment data into manageable chunks
2. Compress (optional)
3. Add Message Authentication Code (MAC)
4. Encrypt using symmetric key
5. Add record header

Handshake Protocol (TLS 1.2)

Message Flow

1. **ClientHello**: Protocol version, client nonce, preferred cipher suites
2. **ServerHello**: Chosen version, server nonce, chosen cipher suite
3. **Certificate**: Server's X.509 certificate (for authentication)
4. **ServerKeyExchange**: Additional key exchange data (if needed, e.g., Diffie-Hellman)
5. **CertificateRequest**: Request for client authentication (optional)
6. **ServerHelloDone**: Server finished sending handshake messages
7. **ClientCertificate**: Client's certificate (if requested)
8. **ClientKeyExchange**: Client's key exchange data
9. **CertificateVerify**: Client signs all previous handshake messages
10. **Finished**: Encrypted confirmation of handshake completion

Key Generation Process

- **Pre-master secret** (48 bytes) → **Master secret** → **Shared keys**
- Master secret = PRF(pre_master_secret, 'master secret', client_random||server_random)
- Shared keys = PRF(master_secret, 'key expansion', server_random||client_random)

Cipher Suites

Components

- **Key Exchange Algorithm**: How keys are exchanged (RSA, ECDHE, etc.)
- **Authentication Algorithm**: How parties authenticate each other
- **Symmetric Encryption Algorithm**: Data encryption method
- **Message Authentication Code (MAC)**: Integrity verification

Examples

- **Key Exchange**: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- **Key Transport**: TLS_RSA_WITH_AES_128_CBC_SHA

Change Cipher Spec Protocol

- Single message with value 1
- Notifies that subsequent records will use new cipher suite
- Sent by both client and server

Alert Protocol

- **Structure**: 2 bytes (alert level + alert description)
- **Warning**: Non-critical issues, connection may continue
- **Fatal**: Critical errors, immediate connection termination

Applications

HTTPS

- HTTP + TLS
- Uses port 443 (instead of 80)
- Encrypts URLs, content, form data, cookies, headers
- Domain name still visible in TCP layer

VPN Considerations

- **SSH vs TLS**: Both require app rewriting
- **SSH**: More manual client configuration, port forwarding
- **TLS**: Better integration, minimal client configuration

TLS Summary

Advantages:

- Strong cryptographic protections (Confidentiality, Integrity, Authentication)
- Widely supported

Disadvantages:

- Performance overhead
- Certificate management complexity
- Complex configuration requirements

DTLS Security Protocol

Background

- **Purpose**: TLS-like security for UDP applications
- **Need**: Real-time applications (video conferencing, gaming, streaming) prefer UDP's low latency

UDP Application Requirements

- **Real-time Communications**: Video calls need minimal delay
- **Live Streaming**: Minor packet loss acceptable vs. retransmission delays
- **Online Gaming**: Fast updates more important than guaranteed delivery

DTLS Design Challenges

- **Unreliable Nature**: Must handle packet loss, reordering, duplication
- **Replay Protection**: Guard against packet replay attacks
- **Out-of-Order Protection**: Handle packets arriving out of sequence
- **Real-Time Constraints**: Optimize for efficiency in retransmissions

DTLS Protocol Differences from TLS

Key Addition: HelloVerifyRequest (Cookie)

- **Purpose**: Prevent DoS attacks
- **Process**: Server responds with cookie, remains stateless
- **Requirement**: Client must return cookie in subsequent message
- **Why needed**: UDP has no connection establishment like TCP

Handshake Flow

1. ClientHello
2. HelloVerifyRequest (with cookie)
3. ClientHello (with cookie)
4. ServerHello
5. [Standard TLS handshake messages continue]

DTLS vs TLS Comparison

- **Similarities**: Same cryptographic goals and most handshake steps
- **Key Difference**: Cookie mechanism for DoS protection
- **Additional Considerations**: Packet loss handling, reordering, replay protection

---

QUIC Security Protocol

Background

- **Developer**: Google (2012), IETF standardized (RFC 9000, May 2020)
- **Goal**: Combine reliability, speed, and security
- **Motivation**: Overcome limitations of TCP+TLS and UDP+DTLS combinations

Protocol Comparison Matrix

Protocol	Reliable	Fast	Secure
TCP	YES	NO	NO
UDP	NO	YES	NO
TLS	YES	NO	YES
DTLS	NO	YES	YES
QUIC	YES	YES	YES

QUIC Architecture

Similar to TLS with four main components:

1. **Handshake Protocol**: Establishes secure communication
2. **Change CipherSpec Protocol**: Signals encryption activation
3. **Alert Protocol**: Error handling and connection closure
4. **Application Data Protocol**: Encrypted data transmission

Key QUIC Innovations

1. Optimized Handshake

- **Early Data Transmission**: Server can start sending before handshake completion
- **Integrated Key Exchange**: ClientHello includes key exchange parameters
- **Reduced Round Trips**: Eliminates separate encryption parameter establishment

2. Multiplexing Without Head-of-Line Blocking

- **TCP Problem**: Lost packet blocks entire stream
- **QUIC Solution**: Multiple independent streams in single connection
- **Benefit**: Lost packet in one stream doesn't affect others

3. Flexible Congestion Control

- **TCP-like Control**: Supports traditional congestion control
- **UDP-like Speed**: Can bypass strict TCP rules when appropriate

- **Adaptive**: Balances efficiency with congestion prevention

4. Built-in Security

- **Integrated Encryption**: Security built into protocol, not layered
- **Always Encrypted**: All QUIC packets are encrypted
- **Forward Secrecy**: Supports perfect forward secrecy

QUIC Advantages

- **Lower Latency**: Reduced handshake overhead
- **Better Performance**: No head-of-line blocking
- **Enhanced Security**: Built-in encryption
- **Improved Reliability**: TCP-like acknowledgments with UDP speed

QUIC Disadvantages

- **DoS Vulnerability**: ClientHello doesn't authenticate client identity
- **Resource Intensive**: ServerHello requires more server resources
- **Attack Vector**: Multiple ClientHello messages can overwhelm server

---

Key Exam Topics Summary

Protocol Characteristics

- **TCP**: Reliable, ordered, connection-oriented, slower
- **UDP**: Fast, connectionless, unreliable, efficient
- **TLS**: Secure TCP, complex handshake, certificate-based
- **DTLS**: Secure UDP, cookie-based DoS protection
- **QUIC**: Combines all benefits, modern solution

Security Attacks

- **TCP SYN Flooding**: Half-open connections exhaust resources
- **UDP Amplification**: Small requests trigger large responses
- **TLS**: Certificate management, performance overhead
- **DTLS**: DoS protection via cookies
- **QUIC**: ClientHello flooding potential

Handshake Processes

- **TCP**: 3-way handshake (SYN, SYN-ACK, ACK)
- **TLS**: 10-step process with certificate exchange
- **DTLS**: TLS + cookie verification
- **QUIC**: Optimized with early data transmission

Key Applications

- **TCP**: Web browsing, file transfers, email
- **UDP**: Gaming, streaming, real-time communication
- **TLS**: HTTPS, secure web communication
- **DTLS**: VPN, secure real-time applications
- **QUIC**: Modern web applications, HTTP/3

Performance Considerations

- **Latency**: UDP < QUIC < TCP < TLS < DTLS
- **Reliability**: UDP < DTLS < TCP = TLS = QUIC
- **Security**: TCP = UDP < DTLS = TLS = QUIC
- **Complexity**: UDP < TCP < DTLS < TLS < QUIC