

DNS & DNSSEC

Domain Name System

- Chức năng chính: ánh xạ tên miền (www.hcmus.edu.vn) ↔ địa chỉ IP (172.29.1.16).
- Quy trình phân giải tên miền:
 - Client gửi request đến **Local DNS resolver**.
 - Nếu không biết, resolver hỏi **Root DNS server** → server cấp cao hơn (vd: **.vn** → **edu.vn** → **hcmus.edu.vn**) → đến khi tìm ra IP thật.
- Ví dụ: **www.hcmus.edu.vn** → Local DNS → Root **.vn** → **edu.vn** → **hcmus.edu.vn** → trả IP.

DNS Root Name Servers

- **Root servers** quản lý **top-level domains** (.com, .vn, .org, ...).
- **Authoritative servers**: quản lý domain con.
- **Local resolvers**: hỏi authoritative servers nếu chưa có trong cache.
- Sự kiện lịch sử: **06/02/2007**, nhiều root DNS servers bị **DDoS attack** → chứng minh DNS là mục tiêu quan trọng.

DNS Caching

- **Mục đích**: tăng tốc độ phân giải tên miền, giảm tải root servers.
- **Cơ chế**:
 - Cache kết quả phân giải (IP address).
 - Cache cả **NS records** cho domain.
 - Cache cả **negative responses** (tên miền không tồn tại, ví dụ gõ sai).
- **TTL (Time To Live)**:
 - Chủ sở hữu tên miền quyết định TTL.
 - Dữ liệu hết hạn sau TTL sẽ bị xóa khỏi cache.

DNS “Authentication” cơ bản

- Request chứa một số ngẫu nhiên **TXID 16-bit**.
- Response hợp lệ nếu có **TXID giống request**.
- Vấn đề: TXID quá ngắn (65.536 khả năng) → kẻ tấn công có thể đoán.
- Nếu đoán đúng → response giả sẽ **lưu trong cache trong thời gian TTL** → dẫn đến **DNS cache poisoning**.

Bailiwick Checking (từ 1997)

- Chỉ chấp nhận records nằm **trong cùng domain** với câu hỏi ban đầu.
- Ví dụ: query `foo.com`, chỉ chấp nhận kết quả liên quan đến `foo.com`.
- Giúp giảm nguy cơ nhận dữ liệu giả mạo từ domain khác.

Kaminsky's Exploit (2008)

- **Ý tưởng**: ép resolver hỏi về tên miền **không tồn tại** (vd: `doesnotexist.example.com`).
- Attacker gửi liên tục các phản hồi giả chứa:
 - IP giả cho `example.com`.
 - Cộng với các bản ghi phụ (NS record, A record).
- Nếu attacker đoán đúng TXID → **cache bị đầu độc** với IP giả trong TTL dài.
- Kết quả: tất cả request sau đó đến `example.com` đều bị điều hướng đến server của attacker.

DNS Spoofing

- **Cơ chế**:
 - Client → hỏi `host1.foo.com`.
 - Attacker đoán TXID, trả lời giả (`6.6.6.6`).
 - Nếu thua → attacker chờ TTL hết hạn rồi thử lại với `host2.foo.com`, `host3.foo.com`, ...
- Nếu thắng → **cache poisoned** → redirect toàn bộ traffic.

Exploiting Recursive Resolving

- Nếu resolver không biết → nó hỏi authoritative server.
- Attacker giả mạo authoritative server → đưa địa chỉ giả (`6.6.6.6`) + TTL dài.
- Một khi poisoned → không cần thắng thêm lần nào → toàn bộ future queries đều trở đến attacker.

Pharming

- **Phishing** thường dựa vào domain thật, nhưng attacker có thể bypass bằng cách **đầu độc DNS**.
- Ví dụ:
 - User gõ: `www.paypal.com`.
 - Attacker DNS response: `6.6.6.6`.

- **Dynamic pharming:**
 - Ban đầu trả đúng IP để user tin tưởng.
 - Sau đó trả IP giả chứa script độc hại.
 - Trình duyệt tin rằng script này có **cùng origin** với site thật → rất nguy hiểm.

Giải pháp chống DNS Spoofing

- **TTL dài** cho kết quả hợp lệ → giảm số lần query.
- **Random hóa port + TXID:**
 - Không chỉ dựa vào TXID 16-bit, mà thêm port ngẫu nhiên (16-bit).
 - Tổng cộng 32-bit randomness → khó đoán hơn nhiều.
- **DNSSEC:** giải pháp chuẩn.

DNSSEC (DNS Security Extensions)

- **Mục tiêu:** bảo vệ tính toàn vẹn và xác thực DNS.
- **Cơ chế:**
 - DNS server ký dữ liệu bằng chữ ký số.
 - Client có thể kiểm tra tính hợp lệ bằng **public key** của DNS server.
- **Triển khai:**
 - **PK-DNSSEC:**
 - Dùng public key.
 - Server ký dữ liệu sẵn.
 - Vấn đề: làm sao các server khác biết được public key?
 - **SK-DNSSEC:**
 - Dùng symmetric key (Mã hóa + MAC).
 - Mỗi message chứa **nonce** để chống replay attack.
 - Mỗi node chia sẻ symmetric key với **parent**.
 - Zone root server có public key → hybrid.
- **Ưu điểm:** chặn DNS spoofing, đảm bảo người dùng đến đúng server.


Internet Mail Architecture, DKIM, S/MIME

Internet Mail Architecture

Hệ thống mail chia thành nhiều thành phần, mỗi thành phần đảm nhận một vai trò riêng:

- **MUA (Mail User Agent)**
 - Client mail như Outlook, Thunderbird, Gmail app.
 - Người gửi dùng MUA để tạo, định dạng, gửi mail.
 - Người nhận dùng MUA để đọc, lưu trữ mail.

- **MSA (Mail Submission Agent)**
 - Nhận mail từ MUA của người gửi.
 - Áp dụng chính sách domain, tuân thủ chuẩn Internet.
 - Ví dụ: SMTP server của Gmail, Outlook.
- **MTA (Mail Transfer Agent)**
 - Trung gian vận chuyển mail (hop-to-hop).
 - Ví dụ: Postfix, Sendmail, Microsoft Exchange.
- **MDA (Mail Delivery Agent)**
 - Chuyển mail từ MTA đến hộp thư của người nhận (Message Store).
 - Ví dụ: Procmal, dovecot.
- **MS (Message Store)**
 - Nơi lưu mail đã đến cho người nhận truy cập (POP3, IMAP).

 Nói ngắn gọn:

MUA → MSA → (MTA → MTA ...) → MDA → MS → MUA

S/MIME (Secure/Multipurpose Internet Mail Extensions)

- **Nguồn gốc:** mở rộng từ chuẩn MIME (RFC 822, 2045).
- **Chức năng:**
 - **Mã hóa** (confidentiality).
 - **Ký số** (authentication + integrity).
 - Hỗ trợ multimedia (text, image, audio, video).

Các kiểu dữ liệu S/MIME:

1. **Enveloped Data**
 - Nội dung + khóa phiên được mã hóa bằng khóa công khai của người nhận.
 - Đảm bảo chỉ người nhận có private key mới giải mã được.
2. **Signed Data**
 - Hash nội dung (message digest).
 - Mã hóa digest bằng private key của người gửi → chữ ký số.
 - Người nhận dùng public key để kiểm tra.
3. **Clear-Signed Data**
 - Nội dung để nguyên (ai cũng đọc được).
 - Chỉ chữ ký số là base64-encoded.
 - Người không có S/MIME vẫn xem được nội dung, nhưng không xác thực chữ ký được.
4. **Signed & Enveloped Data**
 - Có thể lồng ghép (encrypt + sign hoặc sign rồi encrypt).
 - Tăng mức độ bảo mật (chứng thực + riêng tư).

S/MIME Certificates

- Dùng **X.509v3 certificates** để chứng thực khóa công khai.
- Chứng chỉ thường do **CA (Certificate Authority)** cấp.
- Giúp đảm bảo người nhận tin rằng public key thực sự thuộc về người gửi.

Điểm yếu của S/MIME:

- Cần cả người gửi và người nhận **đều hỗ trợ S/MIME**.
- Chỉ **ký nội dung mail**, không bảo vệ metadata (vd: tiêu đề, địa chỉ From).
- Khó áp dụng rộng rãi vì đa số user không bật/tương thích.

DKIM (DomainKeys Identified Mail)

- **Định nghĩa:** chuẩn (RFC 4871) cho phép **domain ký số mail**, không cần người dùng thao tác.
- **Cách hoạt động:**
 - Mail server gửi mail → ký bằng private key của domain (chèn chữ ký vào mail header).
 - Người nhận → mail server của họ query DNS của domain gửi để lấy public key (bản ghi TXT chứa key).
 - Dùng public key xác minh chữ ký trong header.
- **Đặc điểm:**
 - Hoàn toàn **trong suốt với người dùng** (MUA không cần hỗ trợ).
 - Áp dụng cho **toàn bộ mail từ domain**.
 - Giúp chứng minh mail thực sự đến từ domain đó, chống giả mạo (spoofing).

DKIM vs S/MIME

- **S/MIME:**
 - End-to-end (người gửi ↔ người nhận).
 - Cần cả 2 bên hỗ trợ.
 - Bảo vệ **nội dung mail**.
- **DKIM:**
 - Domain-to-domain (server ↔ server).
 - Người dùng không cần cài đặt gì.
 - Bảo vệ **nguồn gốc mail**, chống giả mạo "From:".

SSL/TLS

Tổng quan

- **SSL:** giao thức bảo mật được Netscape giới thiệu.
- **TLS:** phiên bản chuẩn hóa của SSL (IETF chuẩn hóa, kế thừa SSL 3.0).

- **Vị trí:** chạy trên TCP, cung cấp **end-to-end security** cho ứng dụng (HTTP, SMTP, FTP, ...).
- **Không phải** là một giao thức đơn, mà là **bộ giao thức nhiều tầng**.

Khái niệm cơ bản: Session & Connection

- **Connection:**
 - Một kênh TCP an toàn (peer-to-peer).
 - Ngắn hạn (transient).
 - Mỗi connection gắn với 1 session.
- **Session:**
 - Một tập các tham số bảo mật (thuật toán, khóa, ...) được tạo trong quá trình handshake.
 - Có thể chia sẻ cho nhiều connection (tái sử dụng để tránh handshake lặp lại).

📌 Hiểu đơn giản:

- Session = “thỏa thuận bảo mật”
- Connection = “kênh dữ liệu an toàn” dựa trên session đó.

SSL Record Protocol

Cung cấp dịch vụ bảo mật cho kết nối SSL/TLS:

- **Confidentiality (bí mật):** dùng symmetric encryption (khóa được sinh từ handshake).
- **Integrity (toàn vẹn):** dùng MAC (Message Authentication Code).

Quy trình:

1. Ứng dụng gửi dữ liệu → Record protocol chia thành khối.
2. Thêm MAC, nén (nếu có), rồi mã hóa.
3. Truyền qua TCP.

Các Sub-Protocols của TLS

a) Handshake Protocol

- **Quan trọng nhất** – nơi thiết lập session.
- Gồm 4 pha:
 1. **Establish security capabilities:**
 - Client gửi danh sách cipher suites hỗ trợ.
 - Server chọn cipher suite.
 2. **Server authentication & key exchange:**

- Server gửi chứng chỉ X.509 (public key).
- Có thể gửi key exchange parameters.
- 3. **Client authentication & key exchange** (tùy chọn):
 - Client gửi chứng chỉ (nếu yêu cầu).
 - Trao đổi key material (vd: premaster secret).
- 4. **Finish**:
 - Tạo master secret → sinh session keys.
 - Xác nhận handshake hoàn tất.

b) Change Cipher Spec Protocol

- Rất nhỏ, chỉ gửi 1 byte giá trị **1**.
- Mục đích: báo rằng từ đây trở đi sẽ dùng cipher suite vừa đàm phán.

c) Alert Protocol

- Gửi thông báo lỗi hoặc cảnh báo.
- Mỗi alert có **2 byte**:
 - Loại (warning/fatal).
 - Mã lỗi.
- Nếu “fatal” → đóng kết nối ngay.

d) Heartbeat Protocol

- Kiểm tra xem kết nối còn “sống” không.
- Gồm 2 loại message: **heartbeat_request** và **heartbeat_response**.

SSL/TLS Attacks

Các dạng tấn công thường gặp:

- **Tấn công vào Handshake Protocol**
 - Ví dụ: downgrade attack (ép client/server dùng cipher yếu).
- **Tấn công vào Record/Application Protocol**
 - **BEAST (2011)**: lợi dụng lỗ hổng trong CBC mode.
 - **CRIME (2012)**: tấn công nén TLS để lộ cookie session.
- **Tấn công vào PKI (Public Key Infrastructure)**
 - Giả mạo hoặc lợi dụng CA yếu để cấp chứng chỉ giả.
 - Nếu client không kiểm chứng đúng → attacker có thể MITM.
- **Khác**:
 - **DoS attack**: flood handshake request để tiêu hao tài nguyên.
 - **Heartbleed (2014)**: lỗi trong OpenSSL Heartbeat → rò rỉ bộ nhớ (private key, session data).

Tóm tắt SSL/TLS

- **Mục tiêu:**
 - Confidentiality (mã hóa dữ liệu).
 - Integrity (MAC chống thay đổi).
 - Authentication (chứng thực server, có thể cả client).
- **Quy trình:**
 - Handshake → sinh khóa, xác thực.
 - Record protocol → mã hóa & toàn vẹn.
 - Các protocol phụ → đảm bảo duy trì trạng thái an toàn.
- **Điểm yếu:**
 - Phụ thuộc PKI (nếu CA bị compromise → nguy hiểm).
 - Các bug triển khai (Heartbleed, downgrade attacks).

IPSec

Tổng quan

- IPSec = tập hợp các chuẩn bảo mật ở **tầng mạng (IP layer)**.
- Có thể triển khai ở:
 - **Router/Firewall**: bảo vệ toàn bộ traffic qua mạng biên.
 - **End system (máy chủ / client)**: bảo vệ traffic giữa các host.

Ưu điểm:

- **Trong suốt với ứng dụng**: không cần thay đổi phần mềm ở tầng ứng dụng.
- **Trong suốt với người dùng**: không cần cấu hình thủ công cho từng user.
- **Quản lý dễ dàng**: cấp phát, thu hồi key theo user/group.
- Hỗ trợ cả:
 - **Site-to-Site VPN** (kết nối 2 LAN).
 - **Remote Access VPN** (người dùng kết nối an toàn từ xa).

Các chức năng chính

IPSec cung cấp 2 dịch vụ chính:

1. **ESP (Encapsulating Security Payload)**
 - Đảm bảo **Confidentiality (mã hóa)** + **Authentication (toàn vẹn/xác thực)**.
 - Sử dụng thuật toán mã hóa đối xứng (AES, 3DES) + MAC (HMAC-SHA2).
 - Thay thế được AH (vì ESP đã có xác thực).
2. **IKEv2 (Internet Key Exchange v2)**
 - Quản lý khóa: sinh, trao đổi, thay đổi key.

- Dùng để thiết lập **Security Associations (SAs)**.
- 3. **AH (Authentication Header)** (hiện ít dùng)
 - Chỉ xác thực & toàn vẹn.
 - Không cung cấp mã hóa.
 - Dần **deprecated** vì ESP đã bao gồm authentication.

Security Association (SA)

- Là khái niệm trung tâm trong IPSec.
- **SA = một mối quan hệ bảo mật một chiều** giữa sender và receiver.
- Mỗi SA định nghĩa:
 - **SPI (Security Parameter Index)**: ID của SA (giá trị 32-bit).
 - **Địa chỉ IP đích**.
 - **Protocol identifier** (ESP hoặc AH).
- Một kết nối IPSec thường cần **2 SA** (song song, mỗi chiều một SA).

Ví dụ:

- Client → Server: SA1 (SPI=1001, ESP, AES).
- Server → Client: SA2 (SPI=1002, ESP, AES).

Chế độ hoạt động

IPSec có 2 chế độ:

1. **Transport Mode**
 - Chỉ mã hóa payload (dữ liệu), giữ nguyên IP header gốc.
 - Dùng cho **end-to-end** (host ↔ host).
2. **Tunnel Mode**
 - Mã hóa toàn bộ IP packet (header + payload).
 - Gói tin mới được bọc thêm header IP mới.
 - Dùng cho **VPN site-to-site** (gateway ↔ gateway).

Cơ chế hoạt động (ESP Example)

1. **Sender:**
 - Dữ liệu gốc → thêm ESP header (SPI + SeqNum).
 - Mã hóa payload bằng key đối xứng.
 - Thêm MAC (HMAC).
 - Đóng gói vào IP packet → gửi đi.
2. **Receiver:**
 - Nhìn vào SPI để tìm đúng SA.
 - Giải mã payload.

- Kiểm tra MAC để đảm bảo toàn vẹn/xác thực.

Kịch bản ứng dụng

- **VPN cho công ty:** nhân viên ở nhà kết nối qua Internet vào mạng công ty an toàn.
- **Kết nối giữa chi nhánh:** 2 văn phòng dùng IPSec tunnel để liên kết LAN.
- **Bảo mật dữ liệu nhạy cảm:** triển khai IPSec trong mạng nội bộ cho ứng dụng đặc biệt.

Tóm tắt IPSec

- **Mục tiêu:** bảo mật toàn diện ở tầng IP.
- **Thành phần chính:** ESP (mã hóa + xác thực), IKE (quản lý key).
- **Cấu trúc:** dựa trên **Security Associations**.
- **Chế độ:** Transport (end-to-end) & Tunnel (VPN).
- **Ưu điểm:** trong suốt, mạnh mẽ, áp dụng rộng rãi trong VPN.