# SSL Inspection / TLS Inspection là gì?

**SSL Inspection** (còn gọi là **SSL/TLS Decryption**) là quá trình “bóc” lớp, giải mã traffic HTTPS để firwall/IDS có thể xem nội dung bên trong, phân tích và áp dụng rule trên Layer 7 (tầng ứng dụng). Ta có thể hiểu khi traffic là HTTPS thì nội dung payload ( như URL, file, content,…) đã được mã hóa dẫn đến tình trạng IDS/IPS không thể đọc được nội dung, chỉ thấy tiêu đề TCP/IP (chỉ hoạt động tốt ở layer 3/4).

=> **SSL inspection** giúp "bóc" lớp TLS ra để **xem được nội dung thật bên trong**, ví dụ:

* URL truy cập thật “HTTPS” ( Ví dụ bạn truy cập một trang web qua HTTPS: <https://facebook.com/messages>.) Vì HTTPS đã **mã hóa** nên firewall hoặc IDS/IPS như Suricata **chỉ nhìn thấy:**

| **Thông tin** | **Có thấy được không?** |
| --- | --- |
| IP đích (157.240.22.35) | ✅ Có |
| Port đích (443) | ✅ Có |
| Domain (facebook.com) qua SNI (TLS handshake) | ✅ Có (nếu không bị ẩn) |
| Đường dẫn /messages | ❌ KHÔNG THẤY |
| Nội dung bạn gửi hoặc nhận | ❌ KHÔNG THẤY |
| File bạn tải về | ❌ KHÔNG THẤY |

👉 Tức là firewall **biết bạn truy cập facebook.com**, nhưng **không biết bạn đang làm gì ở trong đó**.

**Khi có SSL Inspection / TLS Decryption**

Lúc này firewall đóng vai "người trung gian", giải mã nội dung HTTPS, và **bắt đầu thấy được nhiều hơn**:

| **Thông tin** | **Có thấy được không?** |
| --- | --- |
| URL đầy đủ: https://facebook.com/messages | ✅ CÓ |
| Dữ liệu POST/GET (ví dụ nội dung chat) | ✅ Có |
| File tải về (PDF, exe, zip...) | ✅ Có |
| Cookie, Token | ✅ Có |

👉 Lúc này firewall biết rõ **bạn truy cập đến đường dẫn cụ thể nào**, chứ không chỉ là domain tổng.

* File đang tải về ( ví dụ một máy client trong mạng nội bộ (LAN) đang mở trình duyệt và truy cập vào một trang web độc hại như: <https://malware.example.com/injector.exe>)

Bình thường (KHÔNG có SSL Inspection):

* Traffic đi qua firewall (hoặc IDS) là **HTTPS đã mã hóa**.
* Suricata hay firewall **chỉ thấy IP đích và port (443)**, **không biết URL hoặc nội dung file tải về**.

👉 IDS/IPS chỉ thấy "client đang gửi HTTPS đến 203.0.113.55:443", hết!

Có **SSL Inspection**:

* Firewall sẽ **giải mã traffic HTTPS** giữa client ↔ server.
* Lúc đó:
  + Thấy rõ client đang GET file .exe từ URL nào.
  + Có thể scan file này bằng: Antivirus, threat emulation, hash signature matching hoặc chỉ đơn giản là Policy sẽ chặn file thực thi từ internet.
  + Cuối cùng, nếu file nguy hiểm: Drop connection, hiển thị cảnh báo cho người dùng.

👉 Firewall lúc này giống như đang "đọc được toàn bộ nội dung" như khi bạn truy cập HTTP.

* Dữ liệu POST/GET ( Khi client gửi yêu cầu đến một web server:

| **Loại request** | **Nội dung** |
| --- | --- |
| **GET** | Lấy dữ liệu từ server (VD: truy cập https://site.com/data.json) |
| **POST** | Gửi dữ liệu lên server (form đăng nhập, file upload, payload...) |

* Trong **HTTP thường** (không mã hóa), ta có thể **thấy hết** URL, headers, cookies, body, v.v.
* Nhưng trong **HTTPS**, toàn bộ nội dung đó sẽ được mã hóa.
* Tên miền trong TLS SNI (Bạn truy cập https://facebook.com – client (trình duyệt) gửi facebook.com trong SNI để server biết đưa chứng chỉ phù hợp).

**DS/IPS sử dụng SNI để làm gì?**

* **Block theo domain** (VD: chặn \*.evilsite.com)
* **Alert nếu truy cập C2 domain**
* **Ghi log truy cập TLS**
* **Gán rule phức tạp hơn** (nếu SNI = abc.com và IP = X thì...)

**SNI Encrypted? (ESNI / ECH)**

* TLS 1.3 + DNS over HTTPS (DoH) sẽ dần **mã hóa luôn cả SNI** → gọi là **ECH (Encrypted Client Hello)**.
* ➡️ Lúc đó thì IDS/IPS như Suricata **cũng mù luôn cái domain** nếu không có MITM proxy.

**Vị trí của SNI trong TLS handshake:**

* Diễn ra trước khi mã hóa.
* Gửi qua cổng 443 như thường lệ.
* Chỉ là **metadata** trong packet – IDS/IPS **có thể đọc được** vì chưa bị mã hóa.

💡 Vậy dù nội dung GET/POST bị mã hóa, IDS/IPS vẫn có thể **biết bạn đang truy cập tên miền nào**.

# HTTPS Decryption vs SSL Inspection

| **Tiêu chí** | **HTTPS Decryption** | **SSL Inspection** |
| --- | --- | --- |
| **Định nghĩa** | Quá trình **giải mã** kết nối HTTPS để lấy nội dung plaintext (HTTP) | Quá trình **giám sát & phân tích nội dung đã giải mã**, thường **sau HTTPS Decryption** |
| **Mục đích chính** | Chỉ để **giải mã SSL/TLS** (có thể để logging hoặc redirect) | Giải mã rồi dùng để **phân tích an ninh**, **lọc nội dung**, **IDS/IPS** |
| **Liên quan đến CA** | Cần cài CA giả lập để MITM kết nối | Cũng cần cài CA, nhưng đi kèm với **phân tích Layer 7** |
| **Cách triển khai phổ biến** | Dùng mitmproxy, sslsplit, squid,... | Dùng firewall như **Checkpoint**, **Fortinet**, **Palo Alto**, hoặc tích hợp với Suricata, Snort,... |
| **Liên quan đến IDS/IPS** | Không luôn | Có! Phân tích signature HTTP, payload,... |
| **Kết quả** | Nhận được HTTP plaintext | Nhận HTTP và thực hiện **policy**, **alert**, **drop**,... |

**Tóm gọn dễ hiểu**

* **HTTPS Decryption**: chỉ là **mở khóa TLS**.
* **SSL Inspection**: là **mở xong thì soi nội dung** để tìm virus, lệnh lạ, shell, v.v.

Nói cách khác: *Decryption* là “mở phong bì thư” còn *Inspection* là “đọc nội dung thư và báo nếu có vấn đề”

**Ví dụ thực tế**

| **Tình huống** | **HTTPS Decryption** | **SSL Inspection** |
| --- | --- | --- |
| Bạn dùng mitmproxy chỉ để xem HTTP | Có | Không (chưa phân tích gì) |
| Bạn dùng mitmproxy → gửi tới Suricata | Có | Có (đây là SSL inspection rồi) |
| Checkpoint firewall bật tính năng “SSL Inspection” | Có decryption | Và inspection Layer 7 |

**Vì sao OPNsense không gọi là SSL Inspection?**

Vì OPNsense mặc định không giải mã SSL (chỉ chặn/detect dựa trên IP, SNI), còn nếu bạn kết hợp Suricata thì nó vẫn chưa có cơ chế **MITM + CA** như các firewall thương mại. Do đó, bạn phải **thêm mitmproxy hoặc sslsplit vào** để **tự tạo mô hình SSL inspection thủ công**.

# Cơ chế hoạt động thế nào? (Man-in-the-middle)

SSL inspection thường hoạt động như một dạng **proxy man-in-the-middle (MITM)**: **Client ↔️ [Firewall] ↔️ Server**

Mục tiêu: "Mở khóa" nội dung bên trong các gói HTTPS để kiểm tra malware, DLP, IPS/IDS, lọc URL, v.v..

1. Client gửi HTTPS request đến server (ví dụ <https://malicious.com>)
2. Firewall chặn giữa, **giả lập chứng chỉ của server đó** → client nghĩ mình đang nói chuyện với server.
3. Firewall giải mã traffic, **scan nội dung**, apply rule.
4. Nếu OK, firewall **tạo lại phiên TLS** với server thật → truyền tiếp dữ liệu.

📌 Yêu cầu: Firewall phải **cài CA giả lập trên máy client** để tránh lỗi HTTPS.

Các cơ chế hoạt động chi tiết:

**1. Passive SSL/TLS Inspection (không giải mã)**

* Không thực hiện MITM.
* Chỉ nhìn thấy metadata: IP, port, protocol, TLS version, SNI (tên miền), certificate info.
* KHÔNG thấy dữ liệu POST/GET, file, path, cookie,...
* IDS như Suricata (trên OPNsense) hoạt động ở kiểu này nếu không có SSL inspection.

**Ưu điểm:** Không cần cài CA giả, không gây lỗi.

**Nhược:** Không phân tích sâu, không quét được mã độc trong HTTPS.

**2. Active SSL Inspection (Full SSL Proxy / MITM Proxy)**

Đây là loại inspection thường chỉ thấy ở NGFW hoặc UTM:

**Quá trình như sau:**

Client 🔒➡️ [Firewall/Proxy MITM] 🔒➡️ Server thật

**Chi tiết:**

1. **Client gửi HTTPS request** đến server (VD: https://facebook.com).
2. **Firewall "chặn" giữa**, không gửi đi ngay.
3. **Firewall tạo chứng chỉ giả mạo** (gọi là *on-the-fly certificate*) cho facebook.com bằng CA nội bộ của firewall.
4. **Client nhận cert giả**, nhưng vì **đã cài CA của firewall vào máy**, nó **tin tưởng** và **thiết lập phiên TLS với firewall**.
5. Firewall giải mã traffic, kiểm tra:
   * URL
   * File download
   * Request body (POST data)
   * Script, cookie, ...
6. Nếu không có gì độc, **firewall mở một phiên TLS mới với server thật**, truyền tiếp dữ liệu.

**Mô hình hình dung đơn giản:**

[Client] ← TLS (fake cert) → [Firewall] ← TLS (real cert) → [Server]

* Firewall là **man-in-the-middle chính hiệu**.
* Nó phải giả mạo **mỗi domain** client truy cập, nhưng CA được cài vào client nên không báo lỗi SSL.

**Yêu cầu kỹ thuật:**

| **Thành phần** | **Vai trò** |
| --- | --- |
| Internal CA | Cấp cert giả cho mỗi domain |
| CA Trust | CA phải được import vào máy client (hoặc toàn bộ AD nếu trong enterprise) |
| Firewall SSL Proxy | Phải thông minh, hỗ trợ generate cert on-the-fly |
| Deep Packet Inspection | IDS/AV/DLP engine phải xử lý được nội dung HTTPS đã giải mã |

**Một số tính năng firewall thế hệ mới kết hợp SSL Inspection:**

| **Tính năng** | **Mô tả** |
| --- | --- |
| URL filtering | Chặn theo full URL/path, không chỉ domain |
| Malware scanning | Quét file tải về từ HTTPS |
| Data Loss Prevention | Phát hiện rò rỉ dữ liệu nhạy cảm (VD: mã số tài khoản, mật khẩu...) |
| App Identification | Nhận dạng ứng dụng thật sự ẩn bên trong HTTPS |
| User-Based Policy | Xác định người dùng dựa trên cert và AD/LDAP |

**Các dòng firewall phổ biến hỗ trợ đầy đủ SSL Inspection:**

| **Vendor** | **Tên tính năng** | **Ghi chú** |
| --- | --- | --- |
| Fortinet | **Deep Inspection** | Có thể chọn chế độ partial hoặc full |
| Palo Alto | **SSL Forward Proxy** | Rất mạnh, hỗ trợ cả TLS 1.3 downgrade |
| Sophos | **Decrypt & Scan HTTPS** | Dễ cài với AD |
| Check Point | **HTTPS Inspection** | Kết hợp Threat Emulation |
| Cisco ASA / Firepower | **SSL Decryption** | Tùy model, tùy license |

# Vì sao Suricata trong OPNsense không làm được điều này mặc định?

**1. Suricata là IDS/IPS, không phải Proxy**

Suricata là một công cụ **phân tích gói mạng (deep packet inspection)**, nhưng **nó không phải là proxy** hay MITM agent.

* Suricata hoạt động kiểu: "Gói nào đến thì tôi nhìn".
* **Không chặn, không giả lập chứng chỉ, không đứng giữa như firewall proxy**.

Vì vậy:

* Với **HTTP (plaintext)** → Suricata thấy tất cả: URL, headers, POST data.
* Với **HTTPS (TLS)** → Suricata **chỉ thấy được:**
  + IP nguồn/đích
  + Port
  + SNI (tên miền trong TLS handshake)
  + TLS version
  + Certificate details (không có nội dung payload)

→ **Không thấy nội dung GET/POST, headers, cookie, file download…**

**2. Không có cơ chế SSL MITM**

Suricata **không có khả năng tạo phiên TLS giả (cert giả)** để "đứng giữa" như một proxy MITM:

| **Firewall Proxy (NGFW)** | **Suricata** |
| --- | --- |
| Tạo cert giả động (CA) | Không có |
| Tham gia thiết lập TLS | Không có |
| Có thể giải mã nội dung HTTPS | Không thể |

Muốn làm MITM, cần có component như: **Squid proxy với SSL bump**, **nghproxy**, hoặc **SSL forward proxy như của Palo Alto/FortiGate**.

**3. Suricata phụ thuộc vào giải mã từ bên ngoài**

Cách duy nhất để Suricata phân tích nội dung HTTPS là:

* **Kết hợp với một công cụ MITM (proxy)** như:
  + Squid proxy có SSL bump
  + HAProxy hoặc nginx reverse proxy (nếu HTTPS terminates tại đó)
* Khi đó Suricata có thể "sniff" traffic **sau khi đã được giải mã**.

Ví dụ:

Client → HTTPS → [Squid/HAProxy - giải mã TLS] → HTTP → Suricata

**4. OPNsense không kích hoạt SSL Bump mặc định**

* OPNsense có thể tích hợp với Squid proxy → nhưng **SSL inspection (SSL bump)** phải **bật thủ công**.
* Suricata trên OPNsense sẽ hoạt động tốt hơn nếu:
  + HTTP đi qua proxy
  + Proxy bật SSL bump
  + Traffic đã được giải mã

# Các NGFW như Check Point có gì hơn?

So với OPNsense + Suricata (miễn phí, mã nguồn mở), các **NGFW thương mại như Check Point, Palo Alto, FortiGate, Sophos, v.v.** có nhiều điểm **vượt trội về SSL Inspection, khả năng phân tích layer 7, và tích hợp bảo mật toàn diện**.

**Check Point NGFW vs OPNsense + Suricata**

| **Tính năng** | **Check Point NGFW** | **OPNsense + Suricata** |
| --- | --- | --- |
| **SSL Inspection (TLS Decryption)** | Có sẵn, toàn diện | Phải cài thêm Squid proxy, cấu hình thủ công |
| **Inline TLS MITM Proxy** | Có, tích hợp CA, auto cert | Không có |
| **Ứng dụng nhận diện (App Control)** | Phân tích Layer 7, định danh app (Facebook, Telegram...) | Không mạnh, phụ thuộc vào port/SNI |
| **Data Loss Prevention (DLP)** | Có | Không có |
| **Antivirus/Antimalware Engine** | Tích hợp sẵn engine Sandblast (AI, Cloud sandbox) | Không có, cần tích hợp ngoài |
| **Threat Intelligence Feed** | Có sẵn, auto cập nhật từ Check Point | Suricata có ET Pro, cần đăng ký |
| **Security Policy theo User (Identity Awareness)** | Có: theo username AD, MAC, IP | Không tích hợp AD sẵn |
| **Log & Report chi tiết + Dashboard chuyên sâu** | Có giao diện quản lý tập trung (SmartConsole, Infinity) | Phải tích hợp thêm (ELK, Grafana...) |
| **Tự động phản hồi (SOAR-lite)** | Có: tự động chặn, gửi alert, script hành động | Không có |
| **Hỗ trợ kỹ thuật & bảo hành** | Có (Hợp đồng support, RMA...) | Không có support chính thức |
| **Hiệu năng xử lý L7** | Hardware optimize, ASIC/NPU | Tùy vào cấu hình máy host |

**Về mặt kỹ thuật: Check Point làm được gì với SSL?**

* **Giải mã mọi HTTPS traffic** dựa trên policy (có thể bypass theo domain/port)
* **Dò quét nội dung file, POST, cookie, tải về…**
* **Chặn URL theo phân loại (URL filtering)** như “adult, malware, anonymizer…”
* **Bảo vệ chống exploit/phishing** trên HTTPS
* **Khả năng “cắm vào AD”** để biết user nào truy cập gì

**Nói dễ hiểu: NGFW như Check Point là "all-in-one security platform":** Nó không chỉ là firewall, mà là **“Security Gateway” toàn diện** — vừa là:

* Firewall + NAT
* Proxy + SSL Inspection
* IPS/IDS + AV + DLP
* App control + Identity control
* Sandboxing + Threat Intelligence + Auto Remediation

**Nhưng... Đổi lại?**

| **Nhược điểm của Check Point** |
| --- |
| **Giá cực cao** (license theo năm, từng tính năng) |
| **Cấu hình phức tạp** (SmartDashboard, policy layer) |
| Phụ thuộc phần cứng/VM appliance |
| Khó tùy biến so với mã nguồn mở |

**Kết luận:**

* Nếu bạn làm **doanh nghiệp lớn**, cần khả năng phân tích L7 mạnh + SSL inspection tốt → **Check Point là lựa chọn an toàn, đáng tin cậy**.
* Nếu bạn **tự build giải pháp open-source**, ưu tiên linh hoạt, học hỏi sâu về cơ chế hoạt động → **OPNsense + Suricata** vẫn rất mạnh, nhưng cần **tích hợp thêm** như proxy, threat intel, ELK stack...

# Một số lựa chọn nếu muốn OPNsense hỗ trợ HTTPS inspection

Nếu muốn OPNsense có khả năng **HTTPS inspection** (SSL/TLS decryption), thì dưới đây là một số lựa chọn phổ biến có thể triển khai để **giải mã HTTPS traffic** cho IDS/IPS như Suricata hoạt động hiệu quả hơn ở Layer 7:

**1. Tích hợp Squid Proxy (Transparent hoặc Explicit)**

**Squid Proxy có thể:**

* Làm **MITM proxy**, giải mã HTTPS
* **Cài CA giả lập** → client không bị lỗi HTTPS
* Lưu log HTTP/HTTPS (URL, POST/GET, Host header)
* Cho phép Suricata bắt traffic sau khi đã giải mã

**Cách tích hợp:**

1. Vào Services → Web Proxy → General bật Squid
2. Bật chế độ **Transparent Proxy** + HTTPS Interception
3. Tạo CA nội bộ trong OPNsense (System → Trust → Authorities)
4. Cài CA đó lên các máy client (thường qua GPO nếu có AD)
5. Cấu hình Suricata để phân tích trên interface có decrypted traffic (hoặc qua TAP/mirroring)

👉 **Lưu ý:** Squid cần cấu hình rất kỹ phần HTTPS Bump (SSL-Bump), tránh lỗi handshake.

**2. Dùng nginx hoặc mitmproxy làm SSL proxy**

Nếu không dùng Squid, bạn có thể dùng:

* [mitmproxy](https://mitmproxy.org/) – công cụ dòng lệnh, mạnh mẽ cho phân tích HTTPS
* nginx (reverse proxy) để terminate SSL

Kịch bản:  
Client → mitmproxy (giải mã) → gửi tiếp traffic → Suricata + forward → Server

**3. Tách Layer 7 sang máy chuyên biệt**

Một giải pháp khác là:

* Cho OPNsense xử lý routing + IDS/IPS L3/L4
* Dùng một VM riêng (hoặc container) chạy proxy SSL như:
  + Squid
  + mitmproxy
  + SSLsplit

→ Traffic ra ngoài sẽ đi qua proxy → giải mã → Suricata nhận được nội dung đầy đủ → phát hiện và chặn hiệu quả hơn

**4. Cài thêm plugin hỗ trợ:**

* os-web-proxy-useracl – Quản lý ACL cho proxy
* os-squid – Plugin chính cho Squid proxy
* os-web-proxy-sso – Tích hợp xác thực (SSO) nếu dùng AD

**Cần nhớ:**

| **Điều kiện** | **Ghi chú** |
| --- | --- |
| Cài CA lên client | Bắt buộc để tránh lỗi chứng chỉ |
| Chọn lọc traffic cần giải mã | Có thể bypass ngân hàng, gov site... |
| Proxy có thể ảnh hưởng performance | Cần tài nguyên đủ mạnh |
| Suricata nên hoạt động sau khi traffic đã được giải mã | Tăng hiệu quả phát hiện |

**Có cần toàn bộ HTTPS phải bị inspect không?**

Không nhất thiết. Có thể:

* Chỉ giải mã traffic từ/to các domain cụ thể (bằng ACL)
* Chặn/cho qua theo SNI, User-Agent, IP reputation trước khi giải mã