

Cung cấp internet cho 3 máy ảo: 2/3

Mỗi khi đăng nhạp phải mfa: 1/3

Luồn gtraffic qua switch - a việt

fire wall xem luồng trafic ấy, xử lý tn - a viẹt

Log từ firewall, tb mạng nối đến siem, show truy cập từ đâu đến đâu,

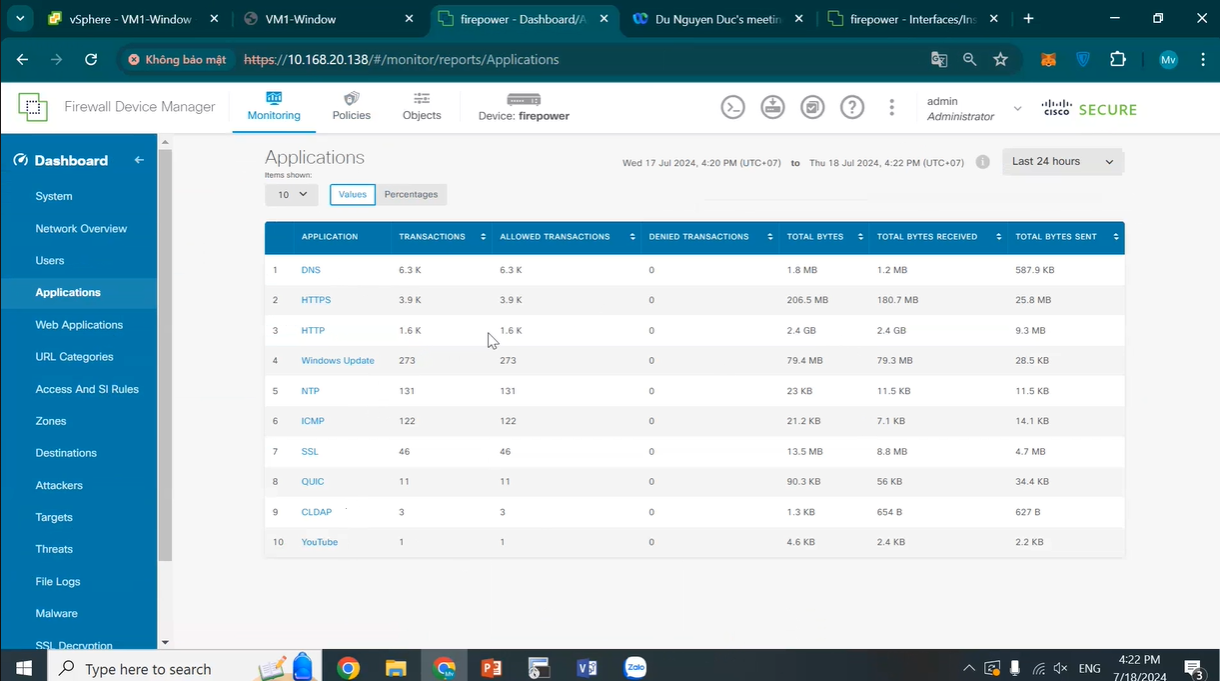
Nghiên cứu firewall, íp, ids

Siem oke onion

1. Cung cấp internet cho vm, đọc log
2. Siem, gửi luồng về siem cả chgy trufbg đến internet

Trong tuần xong duo

Ssl applications bật íp, ids, đứng từ vm vào thử trang web xem có phát hiện ko (vd chặn trang cờ bạc, allow 24h)



Cách xử lý policy (trong slide)

Hiểu firewall: xử lý tt gì trc, luồng tt từ đâu đến đâu

Chuyển siem thành 1 NIC, nhận logs qua ip addr

26/7

# ips ids vs policy

|  | IDS | IPS | fw |
| --- | --- | --- | --- |
| Detection | describes a suspected intrusion once it has taken place and signals an alarm, watches for attacks that originate from within a system | Giống IDS, nhưng được cài đặt giữa luồng traffic -> có thể ngăn chặn trước khi traffic tới nơi | limit access between networks to prevent intrusion and do not signal an attack from inside the network |
|  |  | Encrypted Traffic: Modern IPS solutions can inspect encrypted traffic using techniques like SSL/TLS decryption. Policies alone cannot effectively manage threats within encrypted traffic without such capabilities.  IPS: The IPS, through its threat detection capabilities, can identify the malicious nature of the traffic based on its content and behavior, even if the IP address is not previously known as malicious. | Security Policy: You might have a policy that blocks all traffic from specific IP addresses known to be malicious. However, if a new attack comes from a previously unknown IP or through a compromised legitimate IP, the policy won't block it. |

## IDS có 2 loại:

NIDS:

placed at a strategic point or points within the network to monitor traffic to and from all devices on the network.[[8]](https://en.wikipedia.org/wiki/Intrusion_detection_system#cite_note-8) It performs an analysis of passing traffic on the entire [subnet](https://en.wikipedia.org/wiki/Subnetwork), and matches the traffic that is passed on the subnets to the library of known attacks. Once an attack is identified, or abnormal behavior is sensed, the alert can be sent to the administrator. NIDS function to safeguard every device and the entire network from unauthorized access. NID Systems are also capable of comparing signatures for similar packets to link and drop harmful detected packets which have a signature matching the records in the NIDS.

An example of an NIDS would be installing it on the subnet where firewalls are located in order to see if someone is trying to break into the firewall.

## Giao tiếp vs ht:

* On-line NIDS deals with the network in real time. It analyses the [Ethernet packets](https://en.wikipedia.org/wiki/Ethernet_frame) and applies some rules, to decide if it is an attack or not.
* Off-line NIDS deals with stored data and passes it through some processes to decide if it is an attack or not.

HIDS:

Host intrusion detection systems (HIDS) run on individual hosts or devices on the network. A HIDS monitors the inbound and outbound packets from the device only and will alert the user or administrator if suspicious activity is detected. It takes a snapshot of existing system files and matches it to the previous snapshot. If the critical system files were modified or deleted, an alert is sent to the administrator to investigate. An example of HIDS usage can be seen on mission critical machines, which are not expected to change their configurations.

## Phương pháp phát hiện:

**Signature-based**:

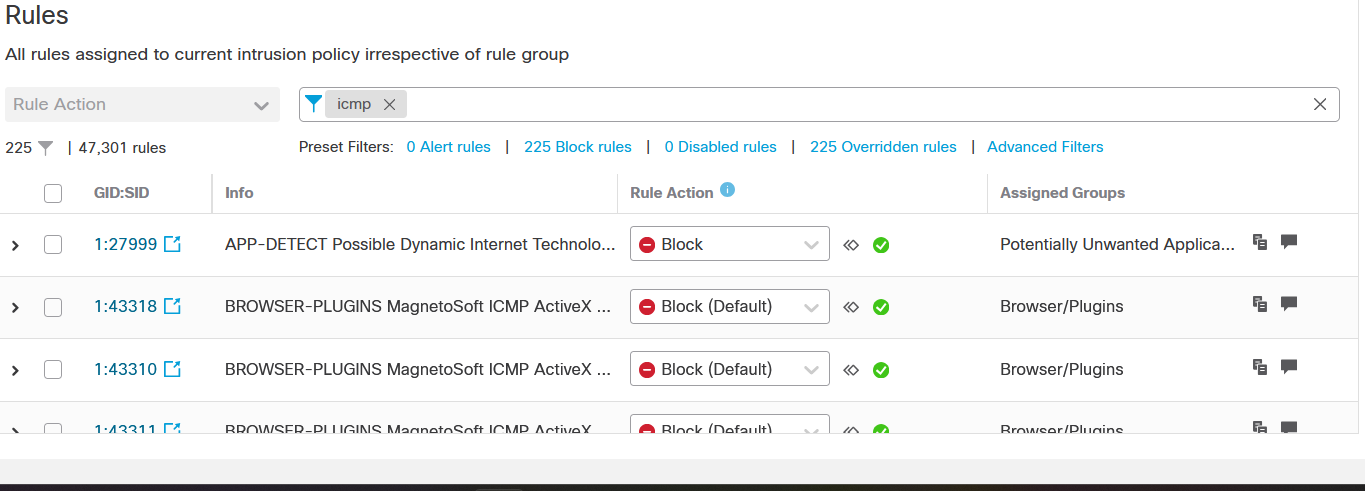
Tìm theo mẫu nhất định, vd chuỗi byte trong luồng traffic, chuỗi instructions của malware.

Phát hiện dễ dàng những tấn công đã biết, khó phát hiện tấn công mới

**Anomaly-based**

Dùng machine learning để tạo mô hình các hành vi tin cậy, so sánh hành vi cần quét với mô hình này.

Phát hiện tấn công mới, nhưng hay gặp false positives. Tốn thời gian



# Dns: để fw chặn đc query của dns, thì fw phải là dns server. Vd block những trang dns

The purpose of a DNS server is to translate what a user types into their browser into something a computer can understand and use to locate a website. In other words, its purpose is to convert a domain name such as www.example.com into an IP address such as 71.232. 101.120.

A DNS firewall works by filtering the traffic that moves along DNS endpoints. This filtration process checks all the traffic using specific rules and policies. If the firewall finds that the traffic violates one of these policies or rules, the web request gets blocked.

DNS blocking works by intercepting this request and checking the domain name against a predefined blocklist or denylist. If the domain name is found on the blocklist, the DNS server responds with a blocked message instead of the IP address, preventing the user from accessing the website.

Câu này có nghĩa là để firewall (FW) có thể chặn được các truy vấn DNS, thì firewall phải đóng vai trò là DNS server. Ví dụ, để chặn các trang web thông qua DNS, firewall cần phải kiểm soát và xử lý các truy vấn DNS. Điều này có nghĩa là tất cả các yêu cầu DNS từ các thiết bị trong mạng sẽ được gửi đến firewall trước, và firewall sẽ quyết định có cho phép truy vấn đó tiếp tục hay không.

Trong thực tế, có một số phương pháp mà firewall có thể sử dụng để chặn các trang web thông qua DNS mà không cần phải trở thành DNS server:

DNS Filtering: Firewall có thể chặn các truy vấn DNS đến các máy chủ DNS cụ thể, hoặc chặn các truy vấn đến các tên miền cụ thể bằng cách sử dụng danh sách đen (blacklist).

DNS Sinkholing: Firewall có thể trả lời các truy vấn DNS cho các tên miền không mong muốn bằng cách cung cấp địa chỉ IP sai, thường là một địa chỉ IP không tồn tại hoặc địa chỉ IP của một server do quản trị viên kiểm soát để theo dõi và quản lý các truy cập.

DNS Inspection: Firewall có thể kiểm tra nội dung của các gói DNS và chặn các truy vấn dựa trên các quy tắc đã được định trước.

Cụ thể hơn, câu này giải thích rằng để chặn các trang web qua DNS, firewall cần phải có khả năng xử lý các truy vấn DNS, tức là cần phải thực hiện vai trò của một DNS server.

# Https: dùng cert để mã hóa

### **TLS 1.2 is slow**

Encryption has always been important online, but historically it was only used for things like logging in or sending credit card information, leaving most other data exposed. There has been a major trend in the last few years towards using HTTPS for all traffic on the Internet. This has the positive effect of protecting more of what we do online from eavesdroppers and [injection attacks](https://blog.cloudflare.com/an-introduction-to-javascript-based-ddos/), but has the downside that new connections get a bit slower.

For a browser and web server to agree on a key, they need to exchange cryptographic data. The exchange, called the “handshake” in TLS, has remained largely unchanged since TLS was standardized in 1999. The handshake requires two additional round-trips between the browser and the server before encrypted data can be sent (or one when resuming a previous connection). The additional cost of the TLS handshake for HTTPS results in a noticeable hit to latency compared to an HTTP alone. This additional delay can negatively impact performance-focused applications.

### **Defining TLS 1.3**

* reducing handshake latency
* encrypting more of the handshake
* improving resiliency to cross-protocol attacks
* removing legacy features

|  | HTTP | HTTPS |
| --- | --- | --- |
| Encryption | Truyền dữ liệu qua HTTP dưới dạng plaintext -> ai cũng có thể đọc đc tt truyền | Mã hóa bằng TLS -> tt đc bảo mật |
| Integrity | Dữ liệu có thể bị thay đổi mà k ai phát hiện | Dùng checksum để xác minh dữ liệu đã không bị thay đổi trong quá trình truyền. Nếu bị thay đổi thì reject dữ liệu. |
| Authentication |  | Dùng chứng thực điện tử đc cấp bởi các CA đáng tin để xác thực danh tính server |

# Root cert vs intermediate cert



Từ trên xuống:

1. Root Certificate. Là các chứng thư điện tử thuộc về CA đã issue. Đc tải sẵn trong các browser, lưu trong trust store. Chứng thư root đc bảo về bởi CA.
2. Intermediate Certificate. Mọc ra từ root. Đứng giữa chứng thư root và chứng thư public. Có ít nhất 1.
3. Server Certificate. Chứng thư được issue cho domain mà ng dùng cần.

|  | Root cert | Intermediate cert |
| --- | --- | --- |
| Vai trò | Nền tảng của trust model | Cung cấp 1 lớp bảo mật bổ sung |
|  | Chứa nhiều gốc trong trusted stores của các trình duyệt phổ biến.  Root CA offline và cấp chứng từ cho intermediate CA, ko cho EE để bảo mật. | Không chứa gốc trong trusted stores. Nối đến CA bên thứ 3.  Được root CA cấp chứng từ, dùng để kí EE và server. Giữa root CA và EE có nhiều intermediate CA. |
| Who signed? | Self-signed | Cross-signed (ko có root trong trust store của browser, nên nối đến root bên thứ 3 khác) |
| Quy trình thu hồi |  | Revoke -> những chứng từ đã issue qua nó sẽ ko đáng tin |
| Thời gian sống | Rất lâuuuuu | Khoảng 2 năm |

# How Does a Browser Trust a Certificate?

Cơ quan cấp chứng chỉ (CA) sử dụng khóa riêng để ký mã hóa tất cả các chứng chỉ mà họ cấp. Chữ ký này đóng vai trò là xác minh không thể thay đổi rằng một CA cụ thể đã cấp chứng chỉ và không có sửa đổi nào sau khi ký. CA khẳng định quyền sở hữu khóa ký này bằng cách sở hữu chứng chỉ gốc, còn được gọi là điểm neo tin cậy, tương ứng với khóa công khai. Việc tạo, quản lý và sử dụng gốc của CA được quản lý và kiểm tra chặt chẽ, tuân thủ các Yêu cầu cơ bản do Diễn đàn CA/B đặt ra.

Tất cả các chứng chỉ bên dưới chứng chỉ gốc đều tin tưởng vào chứng chỉ gốc và sử dụng khóa công khai của chứng chỉ gốc để ký các chứng chỉ khác. Độ tin cậy của chứng chỉ gốc tạo thành cơ sở để xác minh kết nối SSL/TLS của nhiều ứng dụng phần mềm, bao gồm cả trình duyệt. Do giá trị cao của các chứng chỉ gốc này và các rủi ro tiềm ẩn liên quan đến việc xâm phạm chúng, nên chúng hiếm khi được sử dụng để cấp chứng chỉ thực thể cuối. Thay vào đó, các chứng chỉ trung gian thường được sử dụng. Sau đó, các chứng chỉ trung gian này được sử dụng để cấp chứng chỉ cho khách hàng của họ. Chứng chỉ trung gian tạo thành "Chuỗi tin cậy" giữa chứng chỉ thực thể cuối và chứng chỉ gốc.

Tất cả các chứng chỉ bên dưới chứng chỉ gốc đều đặt niềm tin vào chứng chỉ gốc và khóa công khai của chứng chỉ gốc được sử dụng để ký các chứng chỉ khác. Nhiều ứng dụng phần mềm kế thừa độ tin cậy của chứng chỉ gốc này giống như trình duyệt xác minh kết nối SSL/TLS trên cơ sở độ tin cậy của chứng chỉ gốc. Do giá trị của các chứng chỉ gốc này và rủi ro đi kèm khi một chứng chỉ gốc bị xâm phạm, chúng hiếm khi được sử dụng để cấp chứng chỉ thực thể cuối. Thay vào đó, chúng tôi sử dụng các chứng chỉ trung gian. Sau đó, các chứng chỉ trung gian này có thể được sử dụng để cấp chứng chỉ cho khách hàng của họ. Chứng chỉ trung gian đóng vai trò là "Chuỗi tin cậy" giữa chứng chỉ thực thể cuối và chứng chỉ gốc.

Đây là cách thức hoạt động. CA gốc ký gốc trung gian bằng khóa riêng của mình, giúp chứng chỉ này được tin cậy. Sau đó, CA sử dụng khóa riêng của chứng chỉ trung gian để ký và cấp chứng chỉ SSL cho người dùng cuối. Quá trình này có thể được lặp lại nhiều lần, trong đó một gốc trung gian ký một trung gian khác và cuối cùng là ký chứng chỉ thực thể cuối. Các liên kết này, từ gốc đến trung gian đến thực thể cuối, là chuỗi chứng chỉ hoặc chuỗi tin cậy.

Xem xét kỹ hơn chuỗi tin cậy và ghi nhớ định dạng chứng chỉ X.509 v3, đường dẫn chứng nhận ứng viên phải "chuỗi tên" giữa neo tin cậy được công nhận và chứng chỉ mục tiêu (tức là chứng chỉ thực thể cuối). Làm việc từ neo tin cậy đến chứng chỉ mục tiêu, điều này có nghĩa là Tên chủ thể trong một chứng chỉ phải là Tên đơn vị phát hành trong chứng chỉ tiếp theo trong đường dẫn, v.v. Hình ảnh 2 bên dưới giúp minh họa khái niệm này. Trong ví dụ này, đường dẫn bắt đầu bằng chứng chỉ tự ký có chứa khóa công khai của neo tin cậy. Đường dẫn kết thúc bằng chứng chỉ thực thể cuối. Tất cả các chứng chỉ khác trong đường dẫn được gọi là chứng chỉ CA trung gian. Lưu ý rằng mọi chứng chỉ trong chuỗi ngoại trừ chứng chỉ cuối cùng đều là chứng chỉ CA.

Sơ đồ chuỗi tin cậy hoặc đường dẫn chứng nhận

Hình ảnh 2: Chuỗi tin cậy hoặc Đường dẫn chứng nhận

Trình duyệt được cung cấp kèm theo danh sách gốc tin cậy tích hợp, được gọi là kho lưu trữ tin cậy. Kho lưu trữ tin cậy là một tập hợp các chứng chỉ gốc được tin cậy theo mặc định và được duy trì bởi các công ty tạo ra hệ điều hành và trình duyệt web, chẳng hạn như Apple, Microsoft, Mozilla và Google. Mỗi nhà cung cấp có các tiêu chuẩn và yêu cầu riêng đối với chứng chỉ gốc nhưng tất cả đều yêu cầu CA cấp phát phải trải qua một hoặc nhiều cuộc kiểm toán chứng minh độ tin cậy, tính hợp lệ và sự tuân thủ của họ với Yêu cầu cơ bản của Diễn đàn CA/B trước khi chứng chỉ gốc của họ được đưa vào. Để xác minh chứng chỉ, trình duyệt sẽ lấy một chuỗi chứng chỉ, mỗi chứng chỉ đã ký chứng chỉ tiếp theo trong chuỗi, kết nối gốc của CA ký với chứng chỉ của máy chủ.

# Fw đọc dl của ng dùng ra ngoài (để tránh ng dùng gửi tt bí mật). Cách nào fw đọc đc? Data inspection?

## Cách fw phân tích dữ liệu:

### 1. Stateless inspection

Kiểm tra gói tin + So sánh src, dest, port, ứng dụng với bảng = không tuân thủ quy tắc -> bị chặn.

VD: Response packet port lạ -> block

### 2. Stateful inspection

Giống stateless nhưng lưu lại trạng thái, lịch sử.

VD: Response packet port lạ nhưng thuộc về kết nối đã thiết lập -> allow

### 3. Deep Packet Inspection (DPI)

Kiểm tra gói tin + So sánh nội dung gói vs bảng = không tuân thủ quy tắc -> bị chặn. Phân tích dữ liệu ở tất cả các lớp osi.

Tốn kém tài nguyên

### 4. Behavioral analysis

Phân tích hành vi người dùng để phát hiện các hoạt động bất thường.

### 5. Machine learning

Tự động phát hiện các mối đe dọa mới.

Hay gặp False positive.

6. **TLS inspection**

(HTTPS interception hoặc MitM inspection) Dùng để phát hiện các cuộc tấn công qua HTTPS

**B1.** Khi người dùng cố gắng kết nối đến một trang web HTTPS, yêu cầu này sẽ được chuyển qua firewall. FW chặn traffic và thực hiện TLS handshake với trang web đích, sau đó tạo ra một kết nối TLS mới với thiết bị của người dùng

**B2**. Giải mã và kiểm tra

**B3**. Nếu không có vấn đề -> FW mã hóa lại + sử dụng chứng chỉ gốc đã cài đặt để tạo ra một chứng chỉ tạm thời + gửi lại cho user

**B4**. Root cert của FW đã được cài sẵn vào máy -> browser tin tưởng cert FW tạo ra và nhận

**Cài đặt root cert trên thiết bị người dùng**

**B1.** Firewall hoặc thiết bị bảo mật sẽ tạo ra một Root cert và cài đặt chứng chỉ này trên tất cả các thiết bị trong mạng. Điều này thường được thực hiện trên các máy tính công ty hoặc các thiết bị do công ty cấp phát.

**B2.** Chứng chỉ gốc này được thêm vào danh sách các chứng chỉ đáng tin cậy trên thiết bị người dùng, cho phép firewall có thể thực hiện TLS inspection.

AV scanning can be either proxy-based or stream-based, in which case the firewall examines files block by block. You can also set up your firewalls to send suspicious files to a sandbox, or to an external antivirus platform (Virustotal-like).

URL filtering relies on the fact that HTTPS doesn't usually hide the domain and, if TLS inspection is enabled, it also examines the actual URL in the payload. Firewalls then keep a database which classifies URLs in categories (travel, gambling, health, cryptomining,...). This is also useful for compliance reasons. Example: you don't want to carry out TLS inspection when people access health-related websites.

## Cách fw đọc dữ liệu:

### Inline Deployment: Đứng giữa đường truyền từ máy đến internet

### Network Tap: 1 thiết bị hardware sao chép một bản sao của lưu lượng mạng và gửi đến tường lửa.

### Port Mirroring (SPAN): Một switch sao chép lưu lượng từ một port hoặc VLAN cụ thể đến port của tường lửa.

## Tiến trình phân tích

TLS inspection (HTTPS interception hoặc MitM inspection) dùng để phát hiện các cuộc tấn công qua HTTPS

**B1.** Khi người dùng cố gắng kết nối đến một trang web HTTPS, yêu cầu này sẽ được chuyển qua firewall. FW chặn traffic và thực hiện TLS handshake với trang web đích, sau đó tạo ra một kết nối TLS mới với thiết bị của người dùng

**B2**. Giải mã và kiểm tra

**B3**. Nếu không có vấn đề -> FW mã hóa lại + sử dụng chứng chỉ gốc đã cài đặt để tạo ra một chứng chỉ tạm thời + gửi lại cho user

**B4**. Root cert của FW đã được cài sẵn vào máy -> browser tin tưởng cert FW tạo ra và nhận

**Cài đặt root cert trên thiết bị người dùng**

**B1.** Firewall hoặc thiết bị bảo mật sẽ tạo ra một Root cert và cài đặt chứng chỉ này trên tất cả các thiết bị trong mạng. Điều này thường được thực hiện trên các máy tính công ty hoặc các thiết bị do công ty cấp phát.

**B2.** Chứng chỉ gốc này được thêm vào danh sách các chứng chỉ đáng tin cậy trên thiết bị người dùng, cho phép firewall có thể thực hiện TLS inspection.

## Phân loại

### Hardware firewalls: checks the data coming in from the various parts of the internet and verifies that it is safe.

Dùng packet filtering kiểm tra từng gói tin xem nguồn + vị trí.

So sánh dữ liệu vs bảng cho phép -> nếu cần thì bỏ

### Software firewalls: inspect data that goes in and out of the device.

Phần filter giống hardware

Cài đặt lên 1 máy trong mạng, chỉ bảo vệ máy í

# Flows and Sessions

Flows and Sessions are the basic building blocks of IP communication. A **Flow** is a stream of packets between two hosts that maintains key IP header information. The network device is able to determine how to handle a packet based on 3 key components:

* Source IP
* Destination IP
* Protocol

The network device is programmed to look at the IP Header, and will then decode the protocol to look for additional headers, which I usually call **components**. As an example, ICMP contains these additional components:

* ICMP Type
* ICMP Code

TCP/UDP - by far the most common protocols used on the Internet - include the following:

* Source Port
* Destination Port

The information from the IP Header, and the components pulled from decoding the protocol makes up the information that we call a **5-Tuple** - named that way from the 5 pieces of information that is extracted from each packet. When two hosts communicate to each other, they will use this 5-Tuple to maintain their communication. These messages look like this:

* Flow 1 (from Host A):
  + Source IP: 192.168.10.1 (Host A)
  + Destination IP: 192.168.20.1 (Host B)
  + Protocol: 6 (TCP)
  + Source Port: 1025 (Some random ephemeral port)
  + Destination Port: 80 (HTTP)
* Flow 2 (from Host B):
  + Source IP: 192.168.20.1 (Host B)
  + Destination IP IP: 192.168.10.1 (Host A)
  + Protocol: 6 (TCP)
  + Source Port: 80
  + Destination Port: 1025

These two flows become a **Session**.

The most basic network device can track flows individually and filter traffic based on them, but they are unaware of a full session. This is why these filters are called *stateless firewalls*, and are applied to interfaces as opposed to other contexts such as zones. They're very high-performing, often at or very near line-rate of the interface. But what if you wanted to track the entire session between two hosts? This is where modern firewalls come into the picture.

A modern firewall can track these sessions inside a table called a **Session Table**, and apply policies based on the entire session as opposed to a single flow. This is called **Stateful Packet Inspection** (SPI). So modern firewalls are called *stateful firewalls* for this very reason. SPI makes filtering traffic much simpler but this comes at a cost - namely, memory. The firewall has a finite amount of memory it can use to store information about network flows/sessions in its session table. So the stateful firewall employs different mechanisms to preserve as much memory as possible in its table.

This includes things such as tracking the TCP handshake (SYN, FIN, and ACK; FIN and RST packets) to aggressively remove invalid/incomplete sessions between two hosts, or session timeouts where the stateful firewall will remove session table entries if there is no traffic between two hosts in an existing session for a certain period of time. This is to prevent session table exhaustion or denial of service attacks on the stateful firewall itself.

## Certificate Inspection (SSL/TLS)

For encrypted connections (e.g., HTTPS), firewalls can use SSL/TLS inspection to decrypt and inspect the traffic:

* **Certificate Verification**: Firewalls can inspect SSL/TLS certificates to ensure they are valid and issued by trusted Certificate Authorities (CAs).
* **Decryption and Inspection**: Firewalls can decrypt SSL/TLS traffic, inspect the contents, and then re-encrypt it before sending it to the destination. This requires the firewall to act as a man-in-the-middle (MITM) proxy.

### **User and Entity Behavior Analytics (UEBA)**

Advanced firewalls use UEBA to analyze the behavior of users and devices:

**Behavior Baselines**: They establish baselines for normal behavior and detect deviations that may indicate compromised accounts or insider threats.

**Machine Learning**: They employ machine learning algorithms to improve the accuracy of threat detection and reduce false positives.

để firewall có thể đọc và kiểm tra dữ liệu của người dùng khi gửi ra ngoài (ví dụ, để tránh người dùng gửi thông tin bí mật), một kỹ thuật thường được sử dụng là SSL/TLS inspection, hay còn gọi là HTTPS interception. Đây là cách hoạt động của quá trình này:

### **SSL/TLS Inspection (HTTPS Interception)**

* **Cài đặt Chứng chỉ Gốc trên Thiết bị Người Dùng**:
  + **Chứng chỉ Gốc (Root Certificate)**: Firewall hoặc thiết bị bảo mật sẽ tạo ra một chứng chỉ gốc (Root Certificate) và cài đặt chứng chỉ này trên tất cả các thiết bị trong mạng. Điều này thường được thực hiện trên các máy tính công ty hoặc các thiết bị do công ty cấp phát.
  + **Trust Anchor**: Chứng chỉ gốc này được thêm vào danh sách các chứng chỉ đáng tin cậy trên thiết bị người dùng, cho phép firewall có thể thực hiện SSL/TLS inspection.
* **Firewall Hoạt Động như Man-in-the-Middle (MITM)**:
  + **Intercept và Decrypt**: Khi người dùng cố gắng kết nối đến một trang web HTTPS, yêu cầu này sẽ được chuyển qua firewall. Firewall sẽ chặn kết nối này và thực hiện SSL/TLS handshake với trang web đích, sau đó tạo ra một kết nối SSL/TLS mới với thiết bị của người dùng.
  + **Certificate Generation**: Firewall sử dụng chứng chỉ gốc đã cài đặt để tạo ra một chứng chỉ tạm thời cho trang web mà người dùng đang cố gắng truy cập. Thiết bị người dùng sẽ tin tưởng chứng chỉ này vì chứng chỉ gốc đã được cài đặt sẵn và được tin cậy.
* **Kiểm Tra Dữ Liệu**:
  + **Decrypt và Inspect**: Firewall giải mã dữ liệu HTTPS, cho phép nó đọc và kiểm tra nội dung dữ liệu để phát hiện và ngăn chặn bất kỳ thông tin bí mật hoặc vi phạm chính sách nào.
  + **Re-encrypt và Forward**: Sau khi kiểm tra, firewall sẽ mã hóa lại dữ liệu và gửi nó đến trang web đích. Tương tự, phản hồi từ trang web sẽ được firewall giải mã, kiểm tra, và mã hóa lại trước khi gửi đến người dùng.
* **Logging và Monitoring**:
  + **Ghi nhật ký**: Firewall có thể ghi lại các nhật ký về các kết nối và dữ liệu đã kiểm tra để cung cấp thông tin chi tiết về hoạt động mạng và hỗ trợ cho việc giám sát và báo cáo.
  + **Cảnh báo**: Firewall có thể thiết lập cảnh báo khi phát hiện thông tin nhạy cảm hoặc vi phạm chính sách.

### **Lợi Ích và Hạn Chế**

**Lợi Ích**:

* **Bảo Mật Tốt Hơn**: Phát hiện và ngăn chặn thông tin nhạy cảm bị rò rỉ ra ngoài.
* **Kiểm Soát Chính Sách**: Đảm bảo rằng người dùng tuân thủ các chính sách bảo mật của công ty.

**Hạn Chế**:

**Quyền Riêng Tư**: Người dùng có thể cảm thấy quyền riêng tư bị xâm phạm khi dữ liệu của họ bị kiểm tra.

**Hiệu Suất**: Quá trình giải mã và mã hóa lại dữ liệu có thể làm giảm hiệu suất mạng.

# Trao đổi khóa: công khai, chia sẻ

## 2 loại mã hóa:

* **Mã hóa đối xứng**

Mã hóa và giải mã cùng key

Trao đổi khóa khó -> dùng bất đối xứng khi trao đổi, sau đó dùng đối xứng để giao tiếp

* **Mã hóa bất đối xứng**

Mã hóa + Giải mã 2 key khác nhau

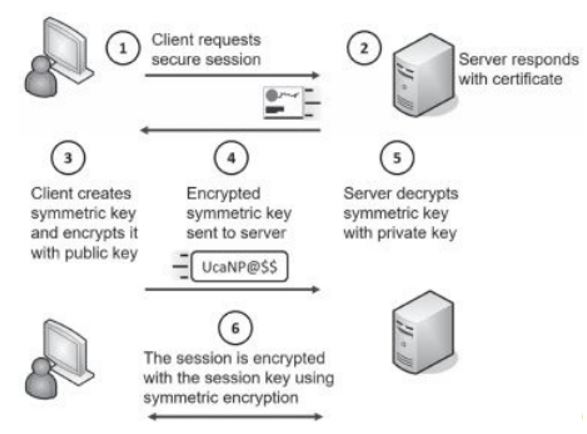
* Tạo cặp khóa = Khóa công khai mã + khóa bí mật mở
* Trao đổi khóa công khai

Ứng dụng: HTTPS, (phần lớn các) blockchain

|  | **Mã hóa đối xứng** | **Mã hóa bất đối xứng** |
| --- | --- | --- |
| Tốc độ | Nhanh (do key ngắn hơn + chỉ dùng 1 key) | Chậm, cần nhiều bộ nhớ hơn |
| Bảo mật | Quá trình vận chuyển khóa có thể gặp nhiều rủi ro | Chỉ chia sẻ khóa công khai -> ít rủi ro |
| Trao đổi khóa | Dùng mã hóa bất đối xứng  Dùng kênh khác | DHE, PSK |
| Ứng dụng | Dùng khi ưu tiên tốc độ hơn bảo mật hoặc khi cần mã hóa 1 lượng dữ liệu lớn  VD:   * Lưu trữ dữ liệu * Các hệ thống giới hạn tài nguyên như IoT | Ưu tiên bảo mật hơn tốc độ hoặc khi yêu cầu xác thực danh tính  VD:   * Chữ kí số * PKI * Email * Trao đổi khóa |

## Kết hợp dùng cả 2:

### **TLS**:



Quá trình trao đổi khóa:

1. Client request server, server gửi response chứa cert. Trong cert có public key của server.
2. Client tạo khóa đối xứng (session key) và mã hóa bất đối xứng bằng public key của server. Client gửi session key cho server + server giải mã bằng private key

Quá trình truyền tin:

Dùng session key (khóa đối xứng) để mã hóa và giải mã

### **Chat**:

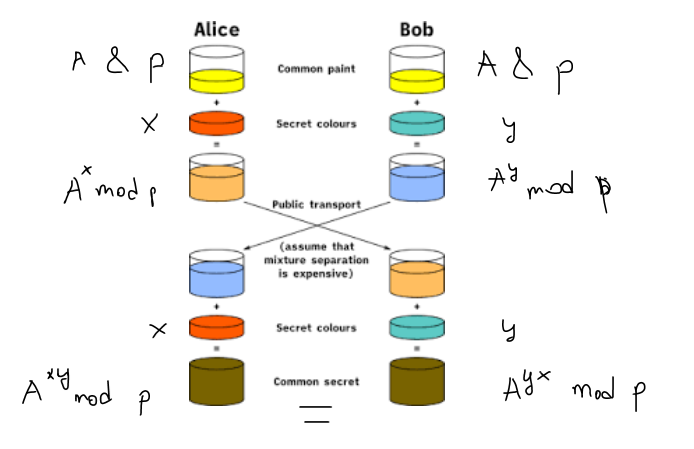
Dùng bất đối xứng xác minh danh tính 2 bên

Trong quá trình gửi tin dùng đối xứng

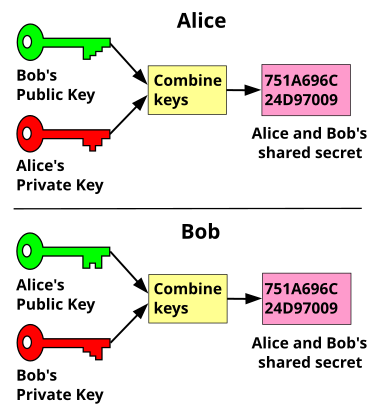
## Các phương pháp trao đổi khóa

### 1. TLS 1.0–1.2

* **Diffie–Hellman:**

****

1. Alice + Bob cùng tạo cặp khóa
2. Kết hợp: khóa công khai đối phương + khóa bí mật của mình = shared secret,
3. Gửi share secret đi



1. Kết hợp: share secret nhận được + khóa bí mật của mình = khóa cần trao đổi

Điểm yếu của TLS <1.3: Non-forward secret connection -> replay attacks

* **RSA**
* **PSK**

### 2. TLS 1.3

Thuật toán trao đổi khóa tạm thời -> cặp khóa mới được tạo cho mỗi lần trao đổi.

* **Diffie–Hellman:**

Trong Diffie–Hellman cũ, 2 bên trao đổi dùng lại khóa bí mật tạo trong bước 1 -> secret không đổi mỗi lần trao đổi khóa. Trong Diffie-Hellman khóa tạm thời (DHE), mỗi lần trao đổi tạo 1 cặp khóa mới, sau khi trao đổi hủy khóa ->

- Nếu private key bị lộ thì chỉ 1 kết nối bị lộ

- Nếu secret bị lộ thì không thể giải mã được do khóa bí mật đã bị hủy

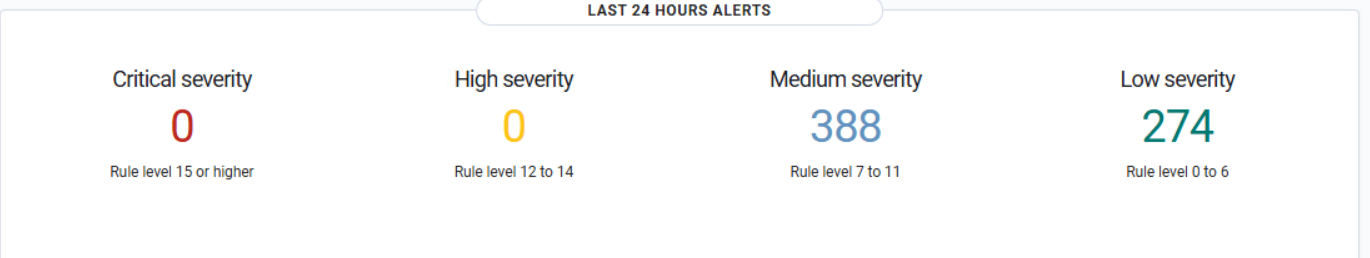
* **RSA:**

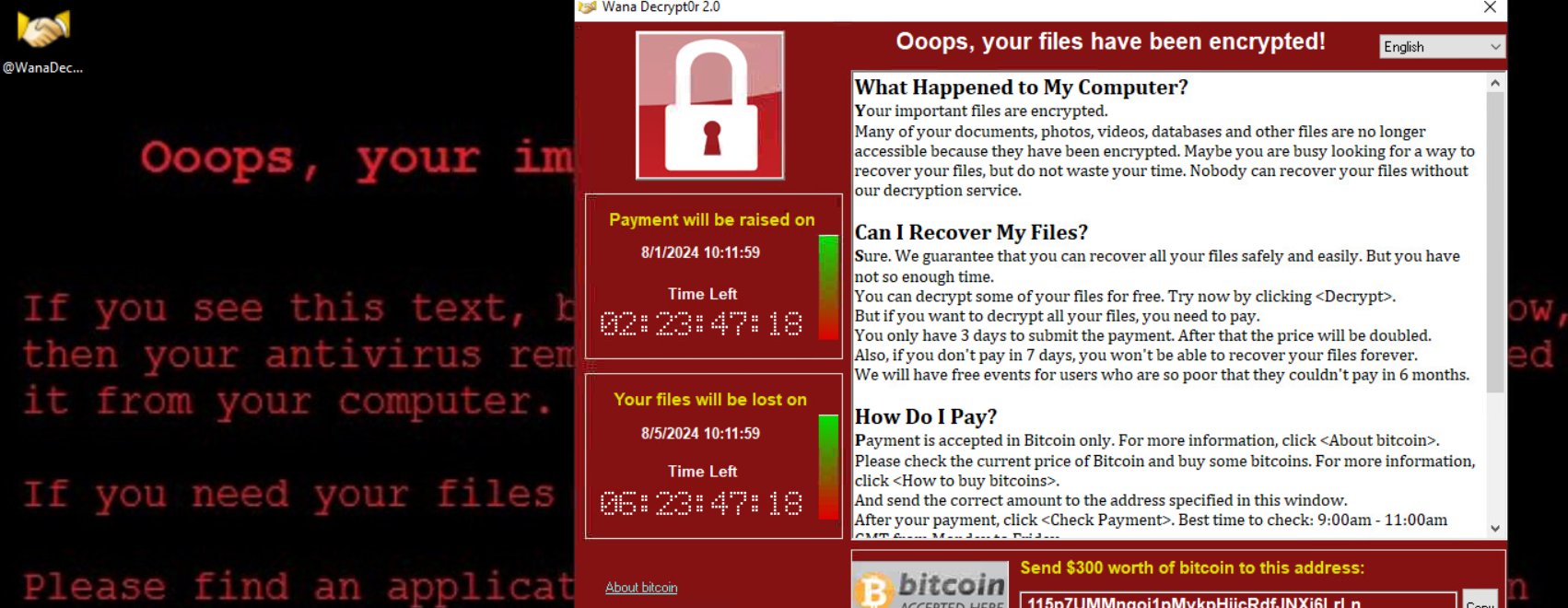
RSA mã hóa quá lâu, không thích hợp dùng trao đổi khóa tạm thời -> bỏ

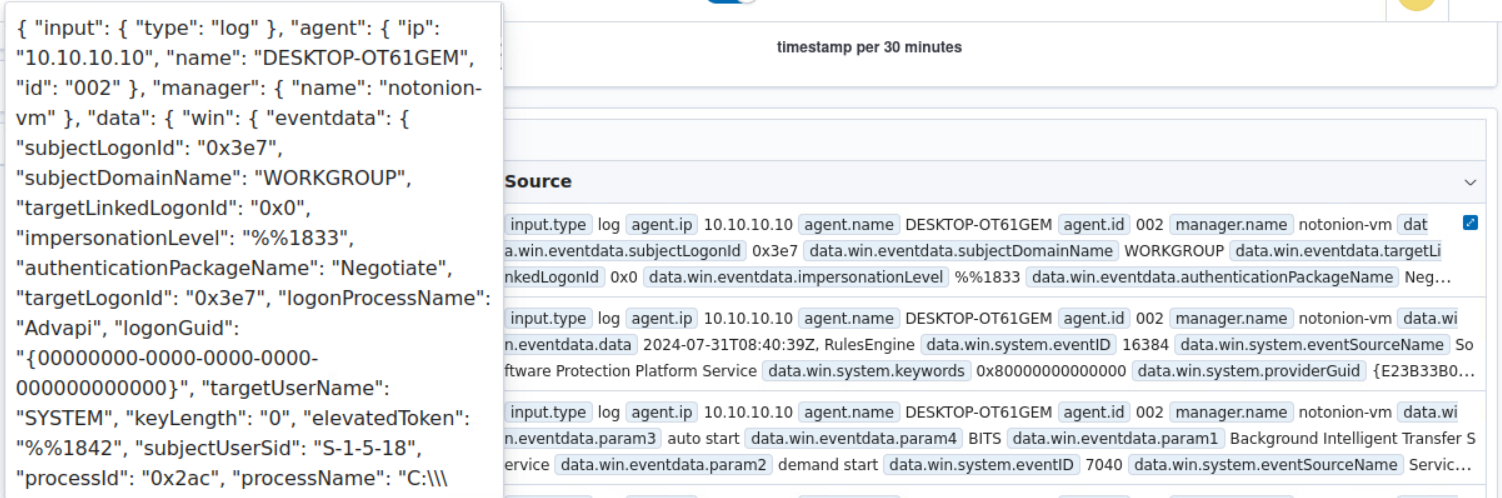
* **PSK:**

Vẫn được sử dụng, có thêm lựa chọn kết hợp với DHE

# Nghiên cứu thêm 1 SIEM và so sánh







# Wireshark để đọc gói tin

1. Định hướng cty: triển khai các tb

Nexus os có wireshark nhúng trong ấy 🙂

⅛

Xu hướng cloud security: cài agent trên máy, doanh nghiệp k cần ddaafu tư phần cứng, chỉ cần mua license

# 1. MFA-Duo

## 1. Định nghĩa MFA

MFA (Multi-Factor Authentication) - hay còn gọi là đa yếu tố xác thực - là một phương pháp bảo mật mà yêu cầu người dùng phải cung cấp nhiều hơn một yế hyu tố để xác minh danh tính trước khi được phép truy cập vào hệ thống hay dịch vụ. Mỗi yếu tố có thể là một trong các loại sau:

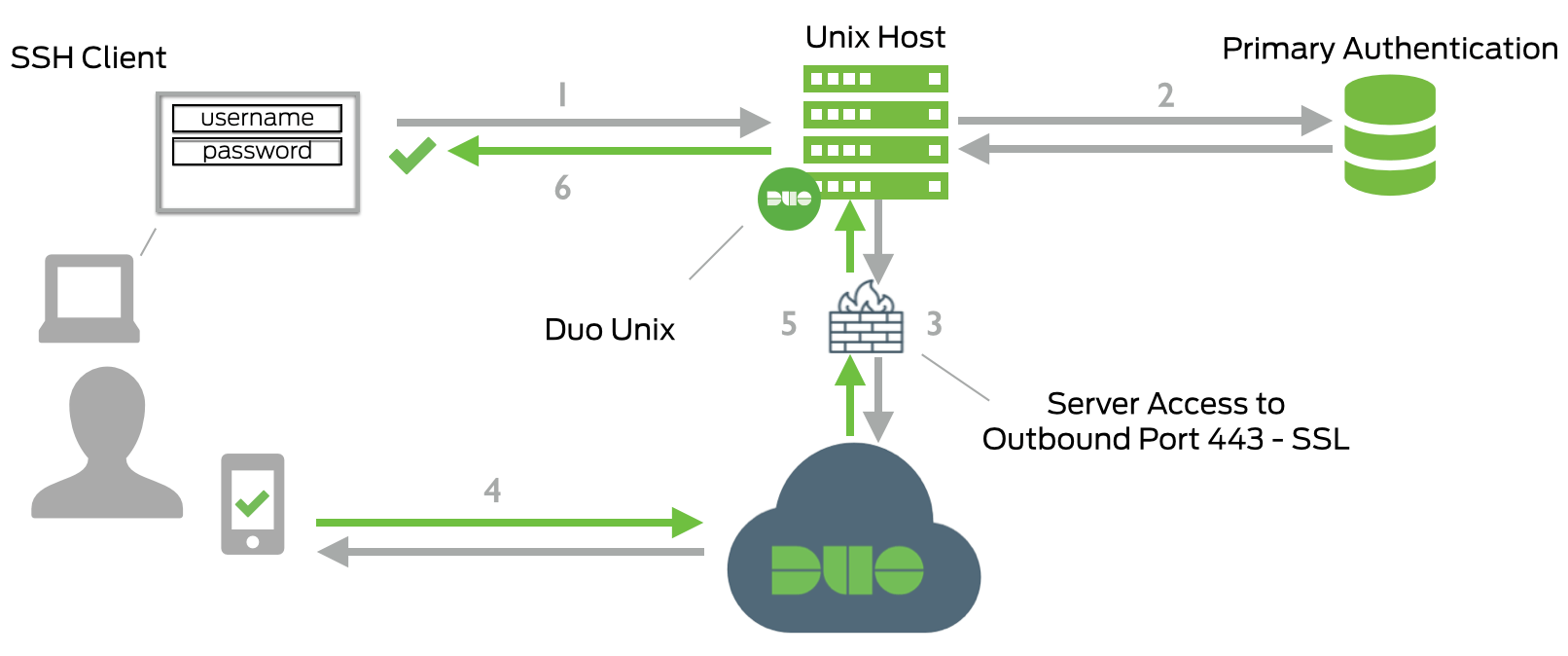
1. **Yếu tố kiến thức**: Mật khẩu, mã PIN.
2. **Yếu tố sở hữu**: Thẻ token sinh mã OTP (One-Time Password).
3. **Yếu tố biến thể**: Xác thực bằng dấu vân tay, nhận diện khuôn mặt.
4. **Yếu tố vị trí**: Xác thực dựa trên vị trí địa lý của người dùng.

Duo là một trong những nhà cung cấp giải pháp MFA phổ biến, cung cấp các phương thức xác thực như:

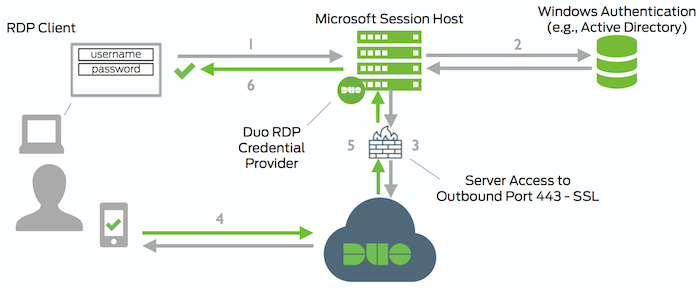
* **Duo Push**: Gửi yêu cầu xác thực đến thiết bị di động của người dùng.
* **Mã OTP (One-Time Passcode)**: Sinh mã OTP để người dùng nhập vào.
* **Gọi điện thoại**: Gọi điện để xác thực qua âm thanh.
* **SMS**: Gửi mã xác thực qua tin nhắn văn bản.

Quá trình MFA giúp tăng cường bảo mật bằng cách đảm bảo rằng người dùng phải có thêm yếu tố xác thực bên cạnh mật khẩu cơ bản để truy cập vào hệ thống, bảo vệ dữ liệu quan trọng và ngăn chặn các cuộc tấn công truy cập trái phép.

## 2. Tổng quan Duo



Ảnh 1.1: Các bước xác thực trên Linux



Ảnh 1.2: Các bước xác thực trên Windows, giống gần hết Linux

1. Kết nối RDP, đăng nhập
2. Xác thực bằng tđn, mk
3. Duo Windows thiết lập kết nối tới Duo Security qua TCP port 443
4. Xác thực lần 2 bằng push, security code, sms
5. Duo Windows Logon nhận phản hồi từ Duo server
6. Đăng nhập thành công

3 tp chính: Integration Key, Secret Key, và API Hostname

* Integration Key (khóa tích hợp): Khi gửi API request đến Duo, Integration Key đc dùng để xác định th bị nào gửi
* Secret Key: Ký API request để xác thực, giống password
* API Hostname: domain name của dịch vụ Duo mà xử lý API request. Khi ht gửi API request đến Duo, ht gửi request đến API hostname đã setup. Hostname chuyển hướng tiếp.

Cách hoạt động

Tạo API request: gồm IKey, timestamp, kí bằng secret key

Gửi cho server Duo bằng API hostname.

Server Duo xác thực người gửi = IKey, xác thực chữ ký = secret key + process = cách gửi push đến đt, tạo passcode,...

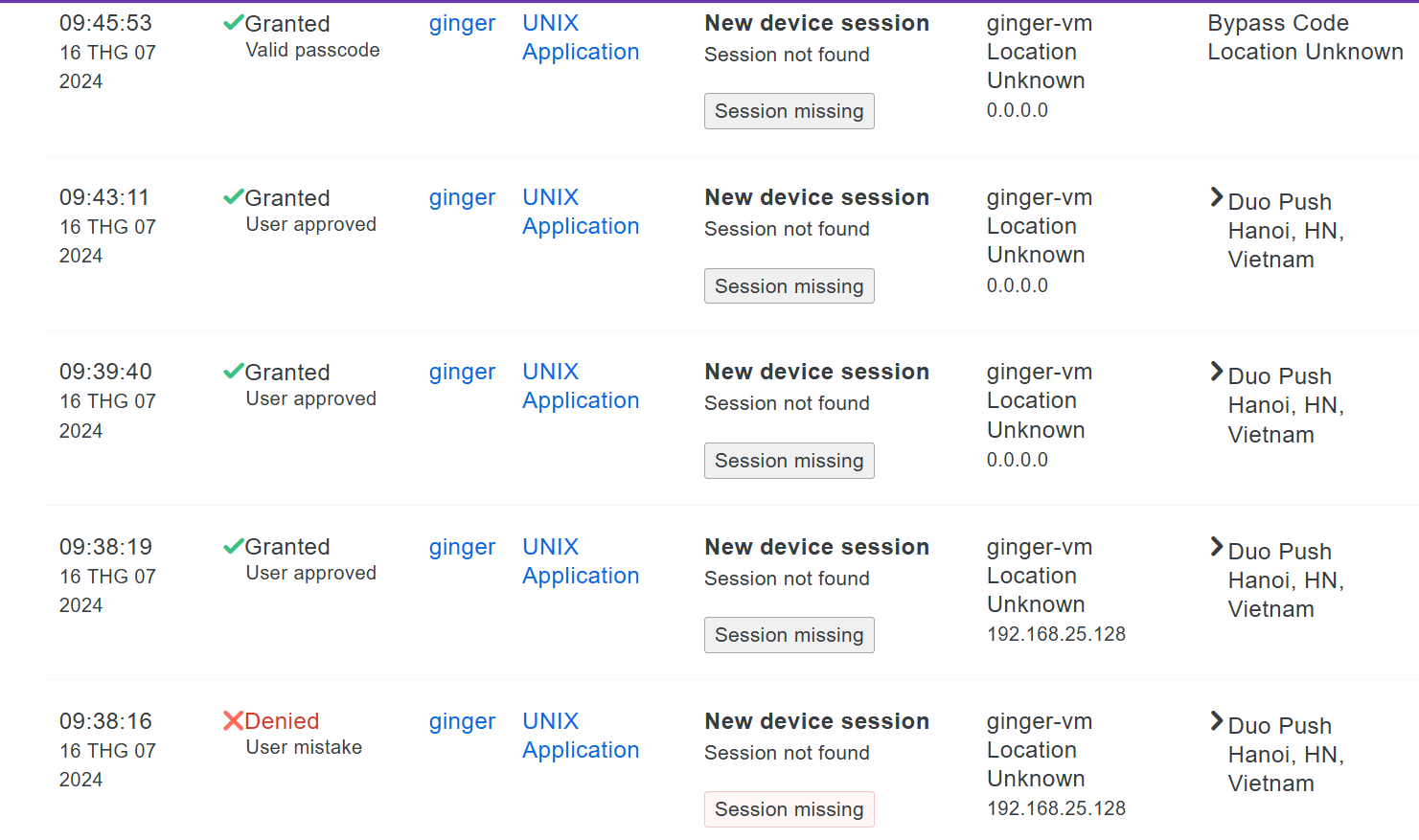
Phản hồi tới th bị = thành công or thất bại

* Khóa mã hóa: Duo sử dụng mã hóa không đối xứng để bảo vệ thông tin liên lạc giữa thiết bị của người dùng và dịch vụ đám mây của Duo. Khóa riêng được lưu trữ an toàn trên máy chủ của Duo, trong khi khóa công khai được phân phối khi cần.
* Thông tin phiên: Bao gồm thông tin chi tiết về nỗ lực đăng nhập, chẳng hạn như tên người dùng, thông tin thiết bị, tên ứng dụng và dấu thời gian yêu cầu.
* Trạng thái phê duyệt: Việc người dùng chấp thuận hoặc từ chối thông báo đẩy được gửi đến máy chủ của Duo để xử lý yêu cầu xác thực.

Cân nhắc về bảo mật

* Đăng ký thiết bị: Thiết bị được sử dụng để xác thực Duo phải được đăng ký với dịch vụ của Duo. Điều này đảm bảo rằng chỉ những thiết bị đã biết và được ủy quyền mới có thể được sử dụng cho MFA.
* Kênh an toàn: Mọi thông tin liên lạc giữa ứng dụng, dịch vụ đám mây của Duo và thiết bị của người dùng đều được mã hóa bằng các giao thức mật mã mạnh (ví dụ: TLS).
* Giảm thiểu việc tiết lộ dữ liệu: Duo giảm thiểu việc tiết lộ thông tin nhạy cảm bằng cách không truyền mật khẩu chính và đảm bảo dữ liệu phê duyệt của người dùng được mã hóa.
* Quy trình xác thực đa yếu tố của Duo Security được thiết kế để cung cấp khả năng bảo vệ mạnh mẽ chống lại truy cập trái phép đồng thời đảm bảo dữ liệu và thông tin liên lạc của người dùng vẫn an toàn.

Admin có thể: Quản lý t bị, policy dựa theo vai trò, groups, khi nào cần MFA, kiểm tra log khi cần



Duo check điện thoại:

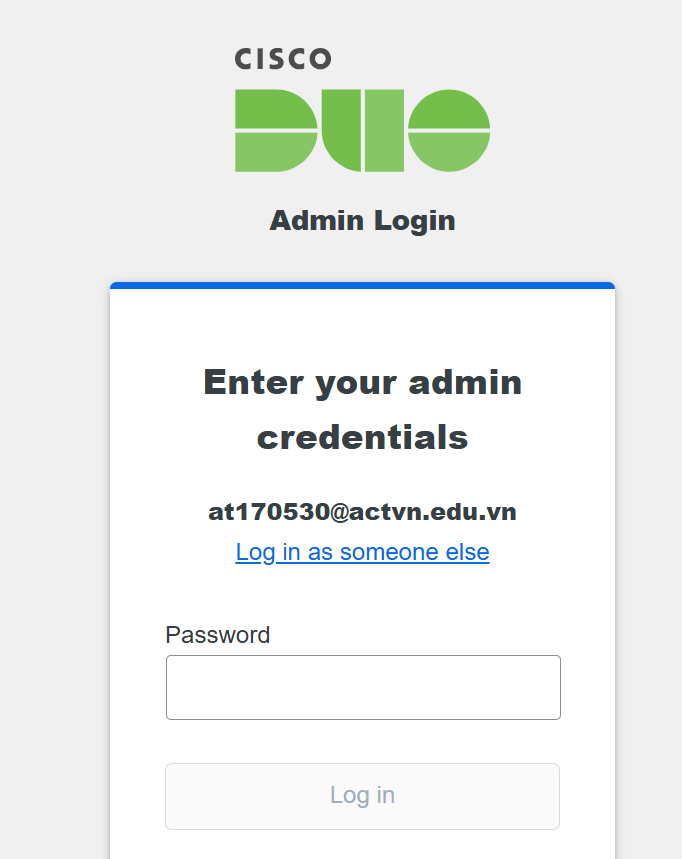
* Tampered: đã crack các kiểu
* Lock screen
* Biometrics

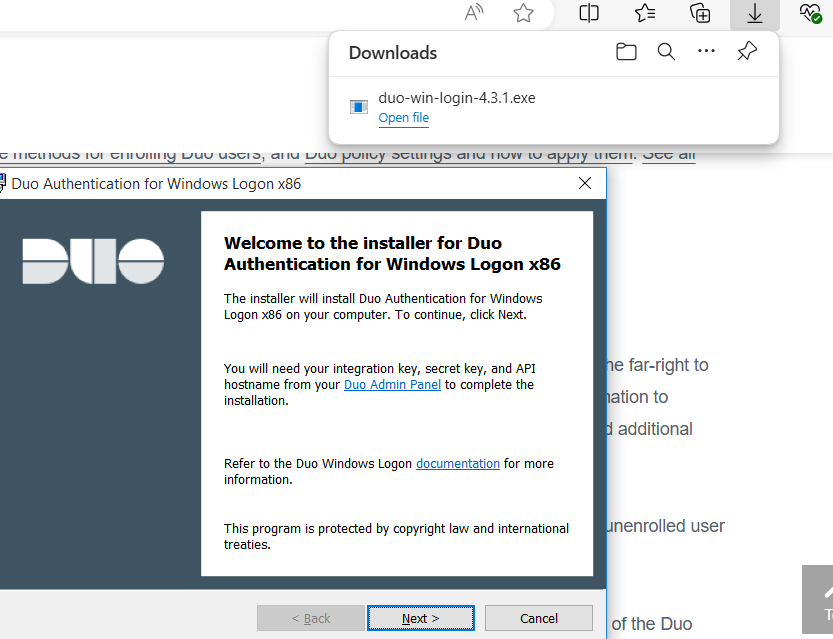
## Thực hành:

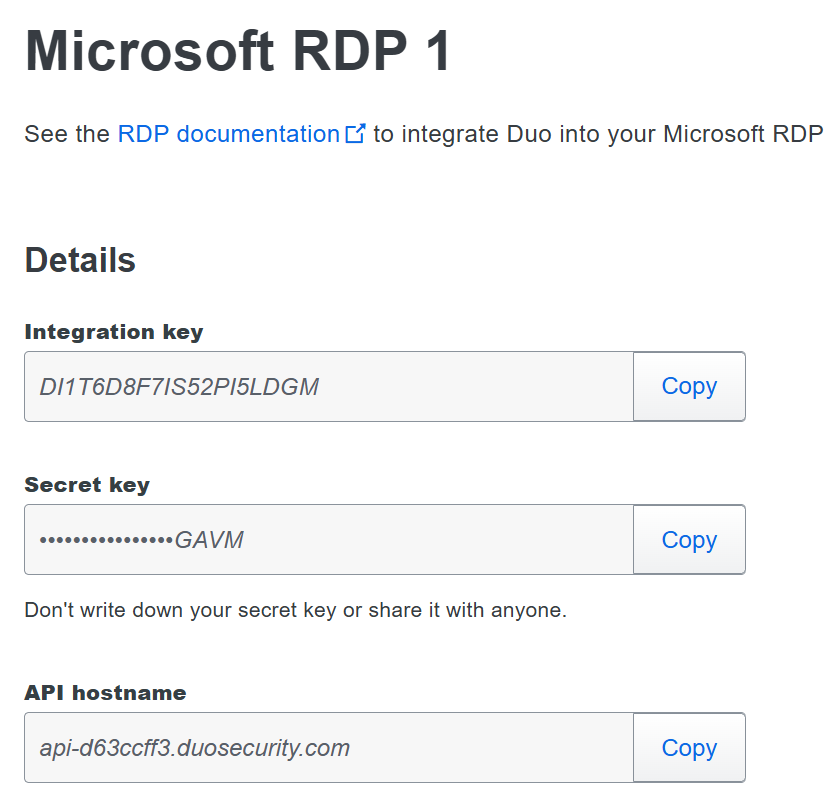
Mô hình:

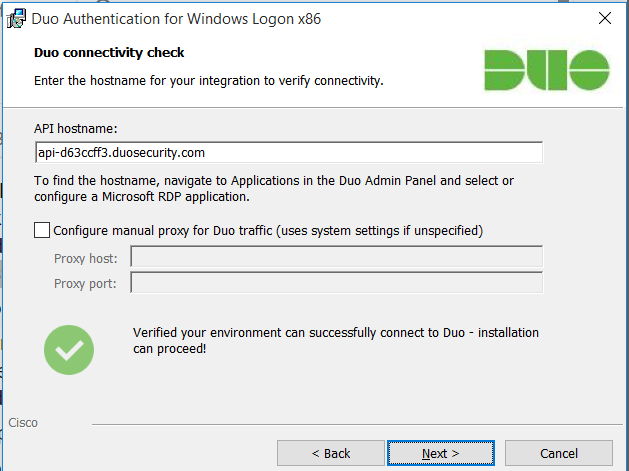
Dự kiến kết quả:

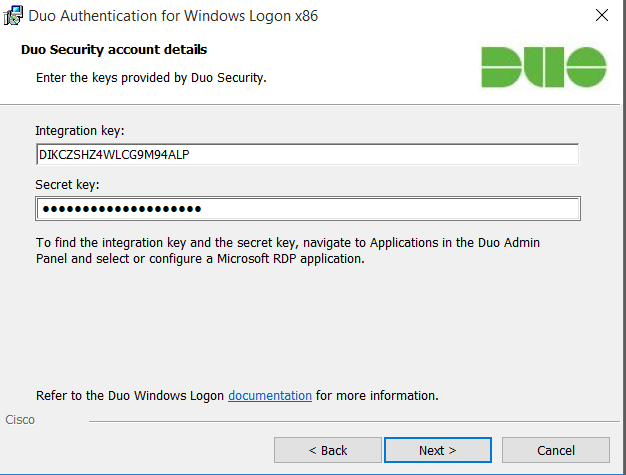
Các bước:

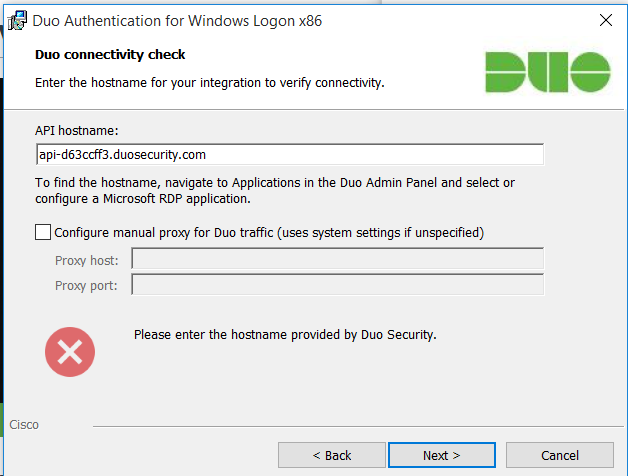


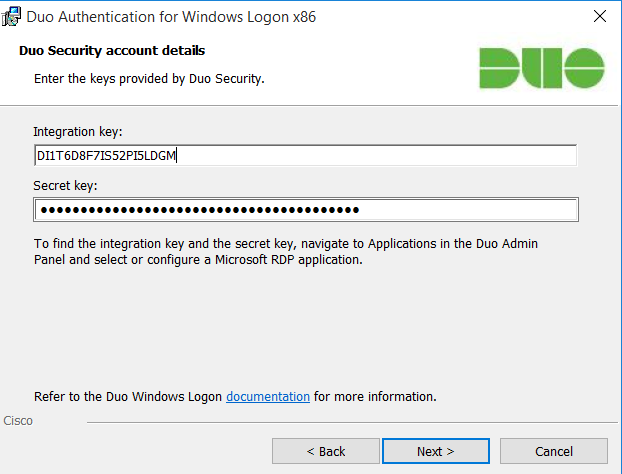






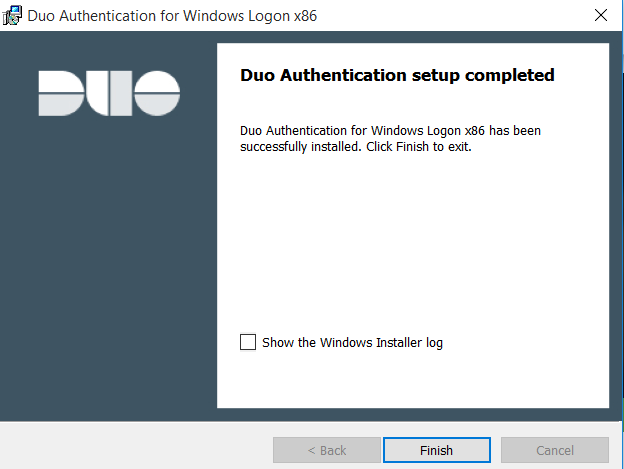


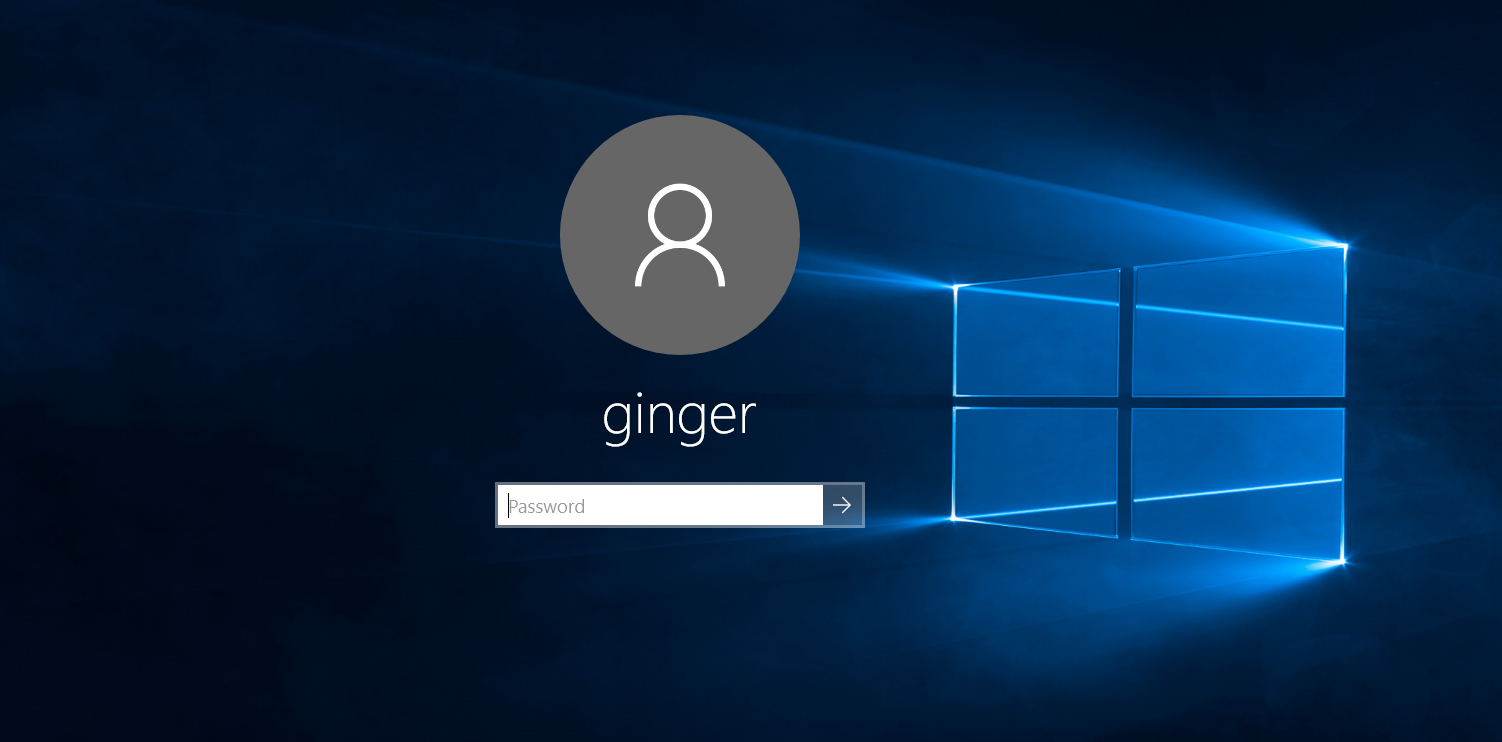


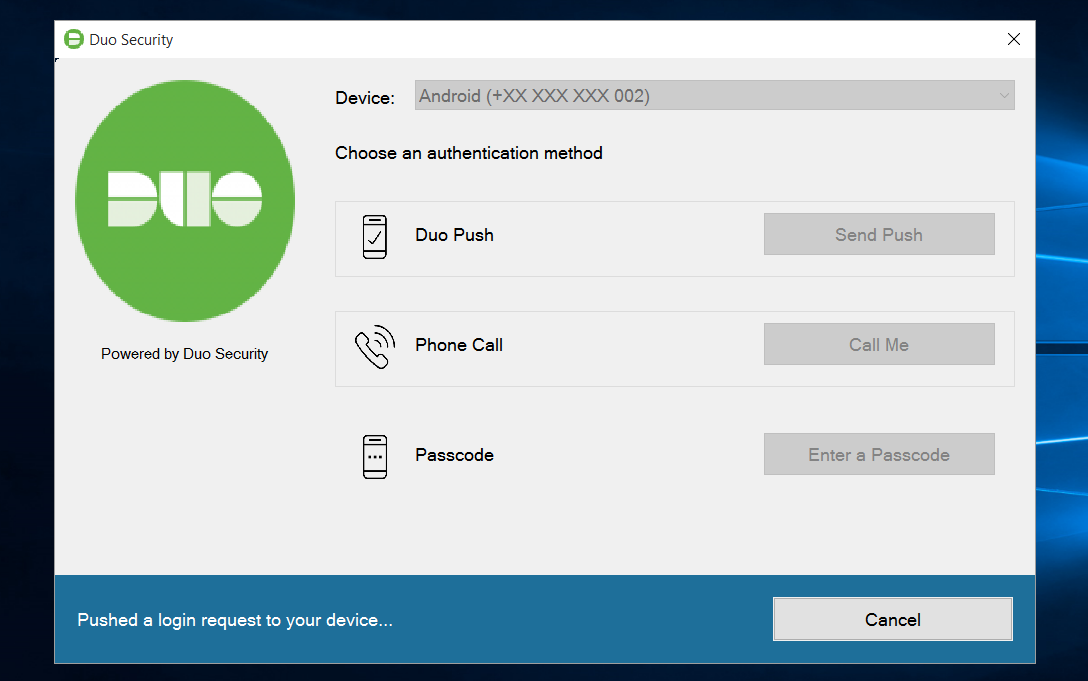




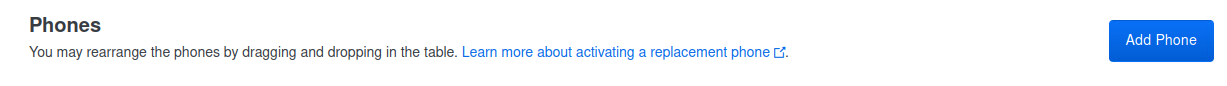
Tick the first option, otherwise we cant access when offline

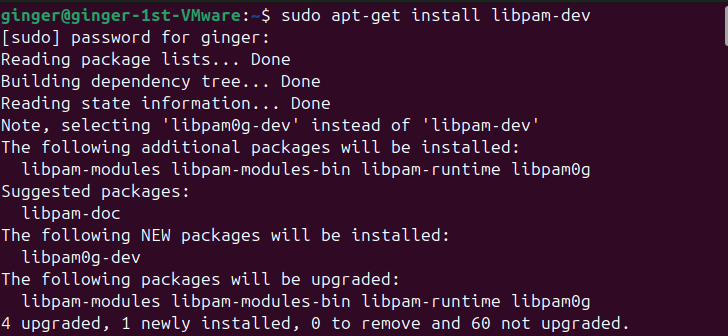


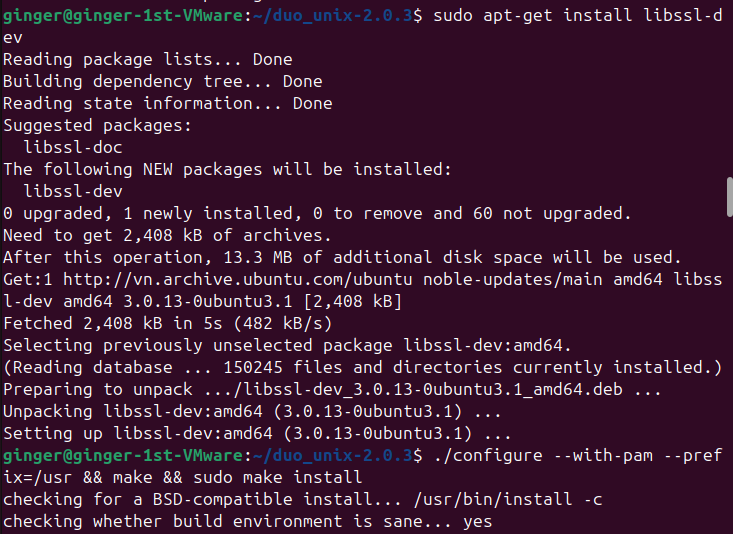


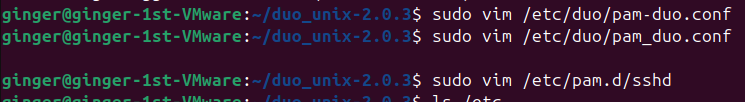


### Linux:

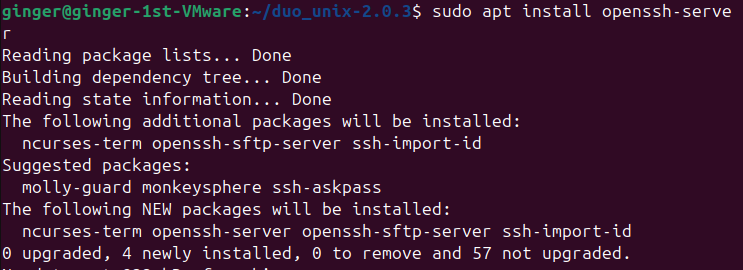






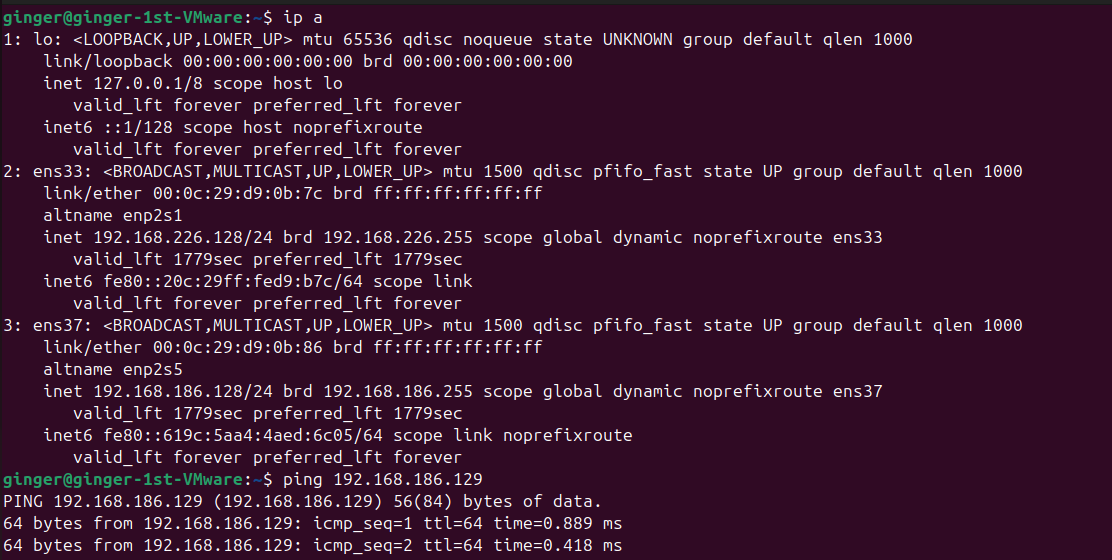


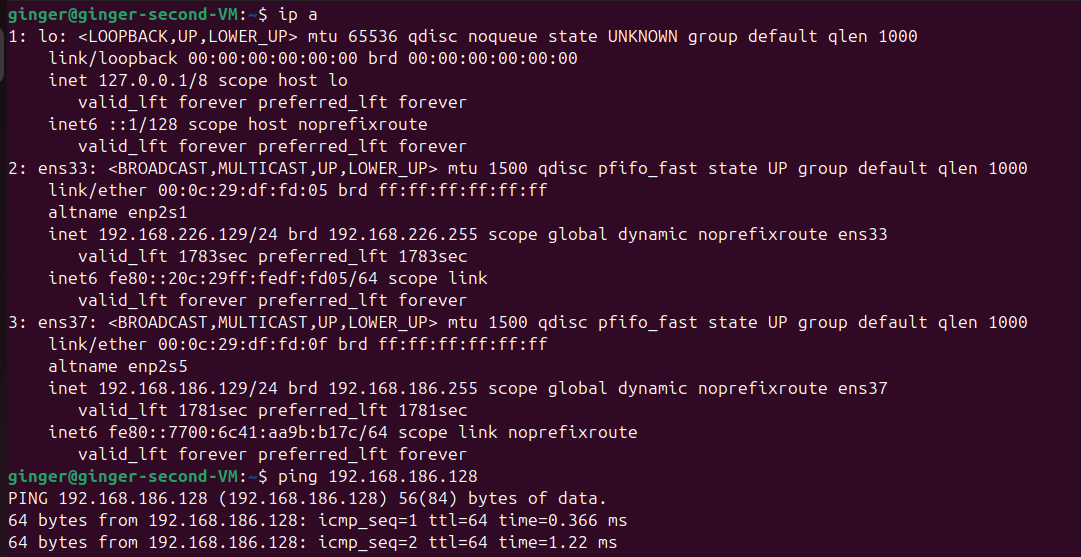


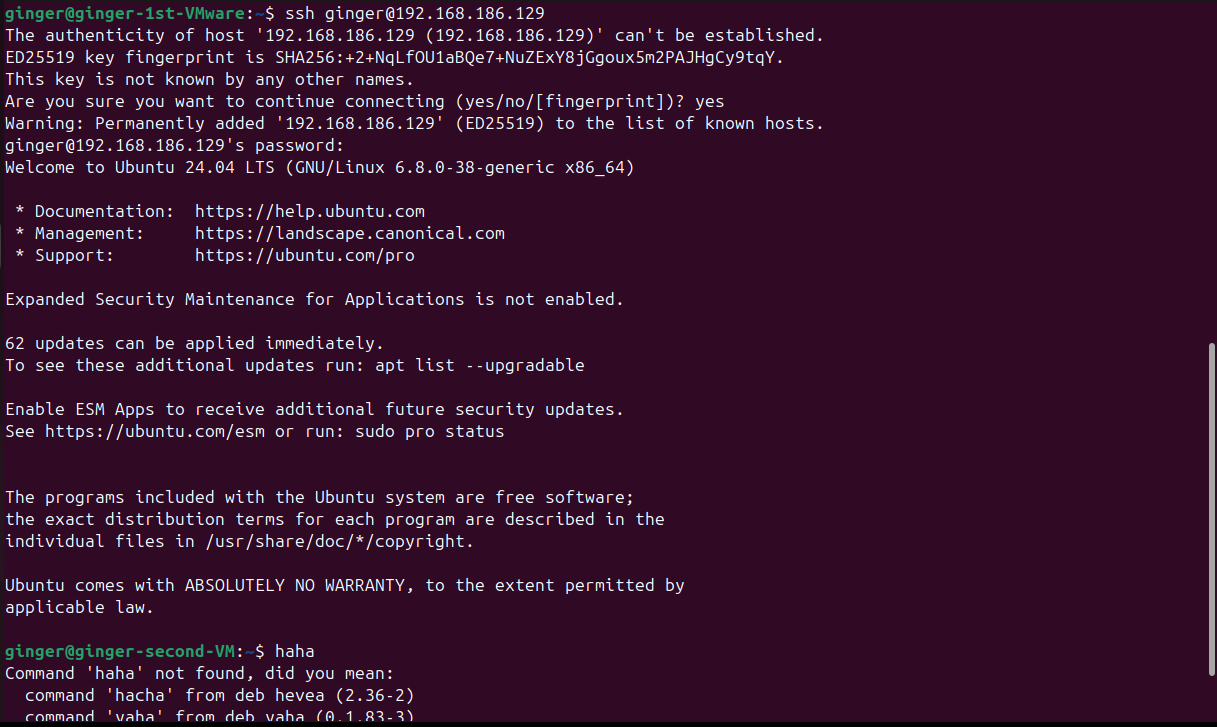












# 2. SIEM

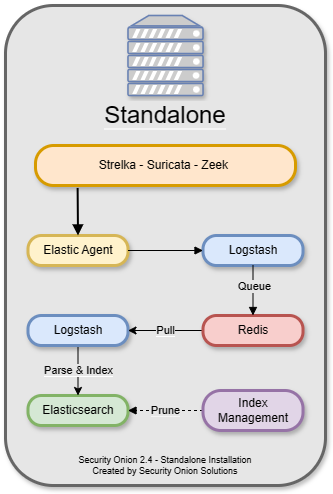
Security information and event management (SIEM) là giải pháp phần mềm tổng hợp dữ liệu nhật ký và sự kiện, thông tin tình báo về mối đe dọa và cảnh báo bảo mật để cung cấp thông tin chi tiết hữu ích về các sự kiện bảo mật tiềm ẩn.

Công việc chính:

* Collecting: nhặt logs từ khắp nơi về (hosts or net devices)
* Normalizing
* Analyzing: theo rules
* Alerting: theo rules

Thành phần:

1. Indexer: xử lý log forwarder nhặt về và normalize thành cặp field + value (ip-192.168.1.1, port-53,..)
2. Forwarder: các agent cài tg tb. Vd cần logs từ win + linux thì cài forwarder vào 2 cái í. Nhặt web server logs, syslog / kernel log, httpd
3. Search head: enable user to search



As we know, each network component can have one or more log sources generating different logs. One example could be setting up Sysmon along with Windows Event logs to have better visibility of Windows Endpoint. Có thể chia nguồn log thành 2 phần:

**1) Host-Centric Log Sources**

Đây là các nguồn nhật ký ghi lại các sự kiện xảy ra trong hoặc liên quan đến máy chủ. Một số nguồn tạo Host-Centric Log là Windows Event logs, Sysmon, Osquery, v.v. VD:

* Người dùng mở file
* Người dùng xác thực
* File exe chạy
* 1 tiến trình thêm/sửa/xóa giá trị registry
* Khởi động Powershell

**2) Network-Centric Log Sources**

Network-related logs được tạo ra khi các máy chủ giao tiếp với nhau hoặc truy cập internet để truy cập một trang web. Một số giao thức dựa trên mạng là SSH, VPN, HTTP/s, FTP, v.v. VD:

* Kết nối SSH
* Chạy 1 file qua FTP
* Web traffic
* Người dùng kết nối tới máy chủ qua VPN.
* Chia sẻ file qua mạng

Mỗi thiết bị trong mạng đều tạo ra một loại nhật ký nào đó bất cứ khi nào có hoạt động được thực hiện trên thiết bị đó, chẳng hạn như người dùng truy cập trang web, kết nối với SSH, đăng nhập vào máy trạm của mình, v.v. Một số thiết bị phổ biến có trong môi trường mạng được thảo luận dưới đây:

Windows Machine

Windows ghi lại mọi sự kiện có thể xem qua Event Viewer. Nó gán một ID duy nhất cho từng loại hoạt động nhật ký, giúp dễ dàng kiểm tra và theo dõi. Các nhật ký này từ tất cả các điểm cuối của Windows được chuyển tiếp đến giải pháp SIEM để giám sát và hiển thị tốt hơn.

Linux Workstation

Hệ điều hành Linux lưu trữ tất cả các nhật ký liên quan, chẳng hạn như sự kiện, lỗi, cảnh báo, v.v., sau đó được đưa vào SIEM để giám sát liên tục. Một số vị trí phổ biến mà Linux lưu trữ nhật ký là:

* /var/log/httpd: Chứa HTTP Request / Response và error logs.
* /var/log/auth.log và /var/log/secure: Lưu trữ nhật ký xác thực.
* /var/log/kern: Tệp này lưu trữ các sự kiện liên quan đến kernel.

Web Server

It is important to keep an eye on all the requests/responses coming in and out of the webserver for any potential web attack attempt. In Linux, common locations to write all apache related logs are /var/log/apache or /var/log/httpd.

Here is an example of Apache Logs:

| 192.168.21.200 - - [21/March/2022:10:17:10 -0300] "GET /cgi-bin/try/ HTTP/1.0" 200 3395  127.0.0.1 - - [21/March/2022:10:22:04 -0300] "GET / HTTP/1.0" 200 2216 |
| --- |

Log Ingestion

All these logs provide a wealth of information and can help in identifying security issues. Each SIEM solution has its own way of ingesting the logs. Some common methods used by these SIEM solutions are explained below:

1) Agent / Forwarder: These SIEM solutions provide a lightweight tool called an agent (forwarder by Splunk) that gets installed in the Endpoint. It is configured to capture all the important logs and send them to the SIEM server.

2) Syslog: Syslog is a widely used protocol to collect data from various systems like web servers, databases, etc., are sent real-time data to the centralized destination.

3) Manual Upload: Some SIEM solutions, like Splunk, ELK, etc., allow users to ingest offline data for quick analysis. Once the data is ingested, it is normalized and made available for analysis.

4) Port-Forwarding: SIEM solutions can also be configured to listen on a certain port, and then the endpoints forward the data to the SIEM instance on the listening port.

SIEM is one major component of a Security Operations Center (SOC) ecosystem, as illustrated below. SIEM starts by collecting logs and examining if any event/flow has matched the condition set in the rule or crossed a certain threshold

Some of the common capabilities of SIEM are:

* Correlation between events from different log sources.
* Provide visibility on both Host-centric and Network-centric activities.
* Allow analysts to investigate the latest threats and timely responses.
* Hunt for threats that are not detected by the rules in place.

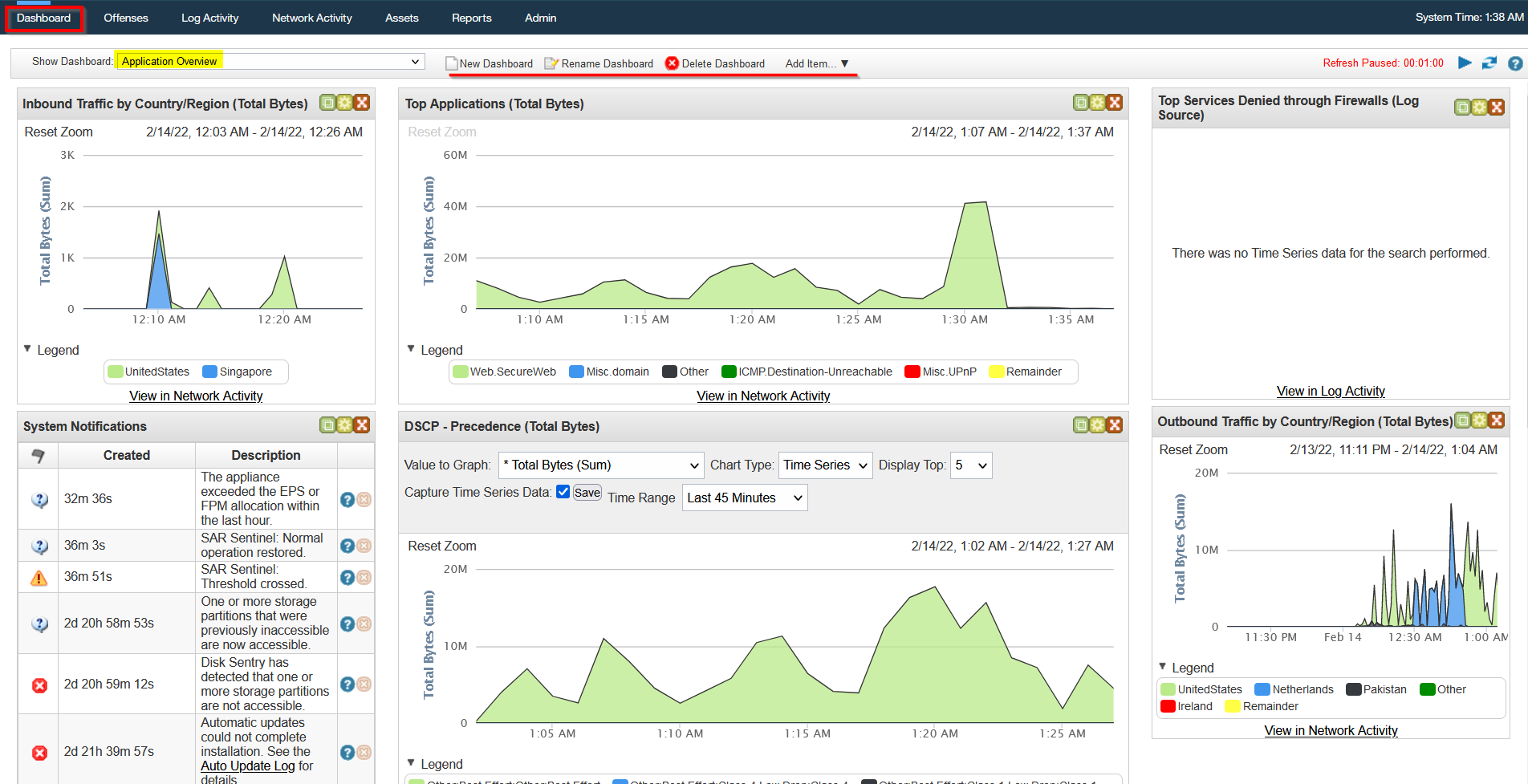
SIEM tool gets all the security-related logs ingested through agents, port forwarding, etc. Once the logs are ingested, SIEM looks for unwanted behavior or suspicious pattern within the logs with the help of the conditions set in the rules by the analysts. If the condition is met, a rule gets triggered, and the incident is investigated.

Dashboard

Dashboards are the most important components of any SIEM. SIEM presents the data for analysis after being normalized and ingested. The summary of these analyses is presented in the form of actionable insights with the help of multiple dashboards. Each SIEM solution comes with some default dashboards and provides an option for custom Dashboard creation. Some of the information that can be found in a dashboard are:

* Alert Highlights
* System Notification
* Health Alert
* List of Failed Login Attempts
* Events Ingested Count
* Rules triggered
* Top Domains Visited

An example of a Default dashboard in Qradar SIEM is shown below:



Correlation Rules

Correlation rules play an important role in the timely detection of threats allowing analysts to take action on time. Correlation rules are pretty much logical expressions set to be triggered. A few examples of correlation rules are:

* If a User gets 5 failed Login Attempts in 10 seconds - Raise an alert for Multiple Failed Login Attempts
* If login is successful after multiple failed login attempts - Raise an alert for Successful Login After multiple Login Attempts
* A rule is set to alert every time a user plugs in a USB (Useful if USB is restricted as per the company policy)
* If outbound traffic is > 25 MB - Raise an alert to potential Data exfiltration Attempt (Usually, it depends on the company policy)

How a correlation rule is created

To explain how the rule works, consider the following Eventlog use cases:

**Use-Case 1:**

Adversaries tend to remove the logs during the post-exploitation phase to remove their tracks. A unique Event ID **104** is logged every time a user tries to remove or clear event logs. To create a rule based on this activity, we can set the condition as follows:

**Rule:** If the Log source is WinEventLog **AND** EventID is **104** - Trigger an alert Event Log Cleared

**Use-Case 2:** Adversaries use commands like **whoami** after the exploitation/privilege escalation phase. The following Fields will be helpful to include in the rule.

* Log source: Identify the log source capturing the event logs
* Event ID: which Event ID is associated with Process Execution activity? In this case, event id 4688 will be helpful.
* NewProcessName: which process name will be helpful to include in the rule?

**Rule:** If Log Source is WinEventLog **AND** EventCode is **4688,** and NewProcessName contains **whoami,** then Trigger an ALERT WHOAMI command Execution DETECTED

In the previous task, the importance of field-value pairs was discussed. Correlation rules keep an eye on the values of certain fields to get triggered. That is the reason why it is important to have normalized logs ingested.

Alert Investigation

When monitoring SIEM, analysts spend most of their time on dashboards as it displays various key details about the network in a very summarized way. Once an alert is triggered, the events/flows associated with the alert are examined, and the rule is checked to see which conditions are met. Based on the investigation, the analyst determines if it's a True or False positive. Some of the actions that are performed after the analysis are:

* Alert is False Alarm. It may require tuning the rule to avoid similar False positives from occurring again.
* Alert is True Positive. Perform further investigation.
* Contact the asset owner to inquire about the activity.
* Suspicious activity is confirmed. Isolate the infected host.
* Block the suspicious IP.

### 2.

Core Pipeline: Elastic Agent [Fleet | Forward] –> Logstash [Manager] –> Redis [Manager]

Logs: Zeek, Suricata, syslog

Logo

2.4

**TABLE OF CONTENTS**

* [About](https://docs.securityonion.net/en/2.4/about.html)
* [Introduction](https://docs.securityonion.net/en/2.4/introduction.html)
* [License](https://docs.securityonion.net/en/2.4/license.html)
* [First Time Users](https://docs.securityonion.net/en/2.4/first-time-users.html)
* **[Getting Started](https://docs.securityonion.net/en/2.4/getting-started.html)**
  + [Best Practices](https://docs.securityonion.net/en/2.4/best-practices.html)
  + **[Architecture](https://docs.securityonion.net/en/2.4/architecture.html#)**
    - [Import](https://docs.securityonion.net/en/2.4/architecture.html#import)
    - [Evaluation](https://docs.securityonion.net/en/2.4/architecture.html#evaluation)
    - [Standalone](https://docs.securityonion.net/en/2.4/architecture.html#standalone)
    - [Desktop](https://docs.securityonion.net/en/2.4/architecture.html#desktop)
    - [Distributed](https://docs.securityonion.net/en/2.4/architecture.html#distributed)
    - [Node Types](https://docs.securityonion.net/en/2.4/architecture.html#node-types)
  + [Hardware Requirements](https://docs.securityonion.net/en/2.4/hardware.html)
  + [Download](https://docs.securityonion.net/en/2.4/download.html)
  + [VMware](https://docs.securityonion.net/en/2.4/vmware.html)
  + [VirtualBox](https://docs.securityonion.net/en/2.4/virtualbox.html)
  + [Proxmox](https://docs.securityonion.net/en/2.4/proxmox.html)
  + [Booting Issues](https://docs.securityonion.net/en/2.4/trouble-booting.html)
  + [Airgap](https://docs.securityonion.net/en/2.4/airgap.html)
  + [Installation](https://docs.securityonion.net/en/2.4/installation.html)
  + [Amazon Cloud Image](https://docs.securityonion.net/en/2.4/cloud-amazon.html)
  + [Azure Cloud Image](https://docs.securityonion.net/en/2.4/cloud-azure.html)
  + [Google Cloud Image](https://docs.securityonion.net/en/2.4/cloud-google.html)
  + [Configuration](https://docs.securityonion.net/en/2.4/configuration.html)
  + [After Installation](https://docs.securityonion.net/en/2.4/post-installation.html)
* [Security Onion Console (SOC)](https://docs.securityonion.net/en/2.4/soc.html)
* [Security Onion Desktop](https://docs.securityonion.net/en/2.4/desktop.html)
* [Network Visibility](https://docs.securityonion.net/en/2.4/network.html)
* [Additional Network Visibility](https://docs.securityonion.net/en/2.4/additional-network.html)
* [Host Visibility](https://docs.securityonion.net/en/2.4/host.html)
* [Third Party Integrations](https://docs.securityonion.net/en/2.4/third-party-integrations.html)
* [Rules](https://docs.securityonion.net/en/2.4/rules.html)
* [Logs](https://docs.securityonion.net/en/2.4/logs.html)
* [Updating](https://docs.securityonion.net/en/2.4/updating.html)
* [Accounts](https://docs.securityonion.net/en/2.4/accounts.html)
* [Services](https://docs.securityonion.net/en/2.4/services.html)
* [Customizing for Your Environment](https://docs.securityonion.net/en/2.4/customizing.html)
* [Tricks and Tips](https://docs.securityonion.net/en/2.4/tricks.html)
* [Utilities](https://docs.securityonion.net/en/2.4/utilities.html)
* [Help](https://docs.securityonion.net/en/2.4/help.html)
* [Security Onion Pro](https://docs.securityonion.net/en/2.4/pro.html)
* [Security](https://docs.securityonion.net/en/2.4/security.html)
* [Telemetry](https://docs.securityonion.net/en/2.4/telemetry.html)
* [Release Notes](https://docs.securityonion.net/en/2.4/release-notes.html)
* [Appendix](https://docs.securityonion.net/en/2.4/appendix.html)
* [Cheat Sheet](https://docs.securityonion.net/en/2.4/cheat-sheet.html)

[**Security Onion Documentation**](https://docs.securityonion.net/en/2.4/index.html)

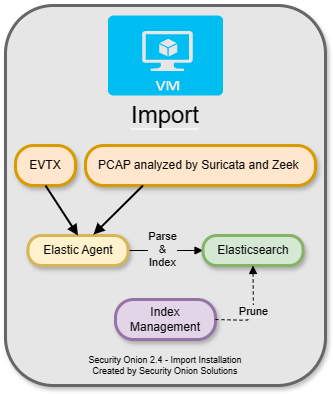
* [Getting Started](https://docs.securityonion.net/en/2.4/getting-started.html)
* Architecture
* [Edit on GitHub](https://github.com/Security-Onion-Solutions/securityonion-docs/blob/2.4/architecture.rst)

# Architecture

If you’re going to deploy Security Onion, you should first decide on what type of deployment you want. This could be anything from a temporary Import installation in a small virtual machine on your personal laptop all the way to a large scalable enterprise deployment consisting of a manager node, multiple search nodes, and lots of forward nodes. This section will discuss what those different deployment types look like from an architecture perspective.

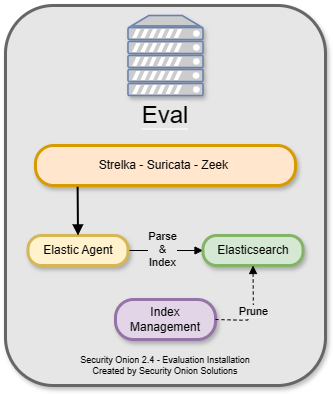
## Import

The simplest architecture is an Import node. An import node is a single standalone box that runs just enough components to be able to import pcap or evtx files using the [Grid](https://docs.securityonion.net/en/2.4/grid.html#grid) page. It does **not** support adding Elastic agents or additional Security Onion nodes.



## Evaluation

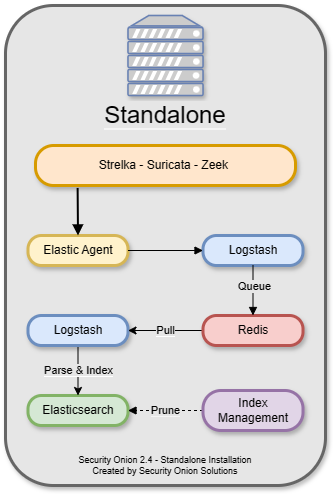
The next architecture is Evaluation. It’s a little more complicated than Import because it has a network interface dedicated to sniffing live traffic from a TAP or span port. Processes monitor the traffic on that sniffing interface and generate logs. [Elastic Agent](https://docs.securityonion.net/en/2.4/elastic-agent.html#elastic-agent) collects those logs and sends them directly to [Elasticsearch](https://docs.securityonion.net/en/2.4/elasticsearch.html#elasticsearch) where they are parsed and indexed. Evaluation mode is designed for a quick installation to temporarily test out Security Onion. It is **not** designed for production usage at all and it does not support adding Elastic agents or additional Security Onion nodes.



## Standalone

Standalone is similar to Evaluation in that all components run on one box. However, instead of [Elastic Agent](https://docs.securityonion.net/en/2.4/elastic-agent.html#elastic-agent) sending logs directly to [Elasticsearch](https://docs.securityonion.net/en/2.4/elasticsearch.html#elasticsearch), it sends them to [Logstash](https://docs.securityonion.net/en/2.4/logstash.html#logstash), which sends them to [Redis](https://docs.securityonion.net/en/2.4/redis.html#redis) for queuing. A second Logstash pipeline pulls the logs out of [Redis](https://docs.securityonion.net/en/2.4/redis.html#redis) and sends them to [Elasticsearch](https://docs.securityonion.net/en/2.4/elasticsearch.html#elasticsearch), where they are parsed and indexed.

This type of deployment is typically used for testing, labs, POCs, or **very** low-throughput environments. It’s not as scalable as a distributed deployment.



## Desktop

The installer includes a [Security Onion Desktop](https://docs.securityonion.net/en/2.4/desktop.html#desktop) option that builds a simple desktop environment. This environment includes a web browser which allows you to log into an existing Security Onion deployment. It also includes some analyst utilities like [Wireshark](https://docs.securityonion.net/en/2.4/wireshark.html#wireshark) and [NetworkMiner](https://docs.securityonion.net/en/2.4/networkminer.html#networkminer).

For more information, please see the [Security Onion Desktop](https://docs.securityonion.net/en/2.4/desktop.html#desktop) section.

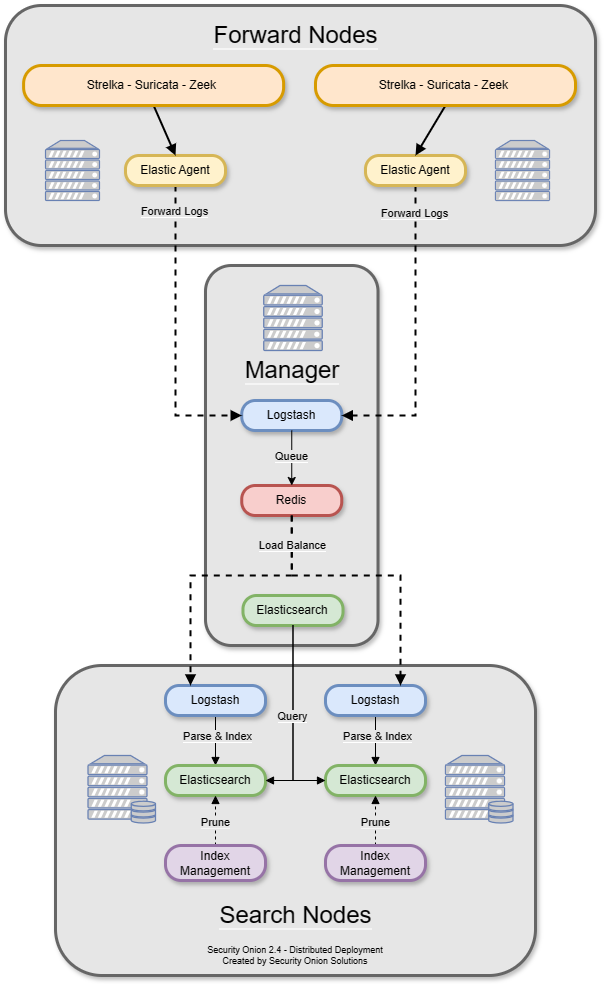
## Distributed

A standard distributed deployment includes a **manager node**, one or more **forward nodes** running network sensor components, and one or more **search nodes** running Elastic search components. This architecture may cost more upfront, but it provides for greater scalability and performance, as you can simply add more nodes to handle more traffic or log sources.

* Recommended deployment type
* Consists of a manager node, one or more forward nodes, and one or more search nodes

**Note**

If you install a dedicated manager node, you must also deploy one or more search nodes. Otherwise, all logs will queue on the manager and have no place to be stored. If you are limited on the number of nodes you can deploy, you can install a **manager search** node so that your manager node can act as a search node and store those logs. However, please keep in mind that overall performance and scalability of a **manager search** node will be lower compared to our recommended architecture of dedicated manager node and separate search nodes.



## Node Types

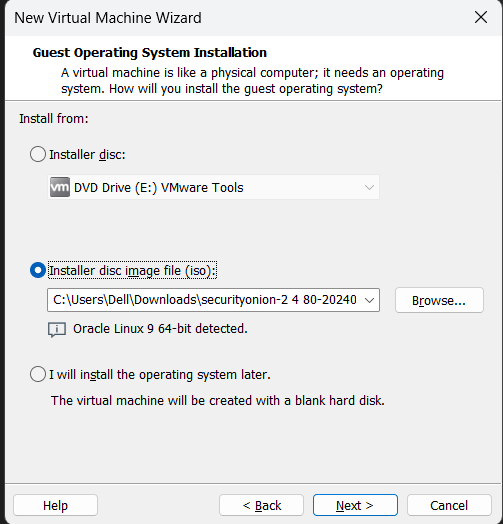
### **Management**

The manager node runs [Security Onion Console (SOC)](https://docs.securityonion.net/en/2.4/soc.html#soc) and [Kibana](https://docs.securityonion.net/en/2.4/kibana.html#kibana). It has its own local instance of [Elasticsearch](https://docs.securityonion.net/en/2.4/elasticsearch.html#elasticsearch), but that’s mainly used for managing the [Elasticsearch](https://docs.securityonion.net/en/2.4/elasticsearch.html#elasticsearch) cluster once search nodes join the cluster. An analyst connects to the manager node from a client workstation (perhaps [Security Onion Desktop](https://docs.securityonion.net/en/2.4/desktop.html#desktop)) to execute queries and retrieve data. Please keep in mind that a dedicated manager node requires separate search nodes.

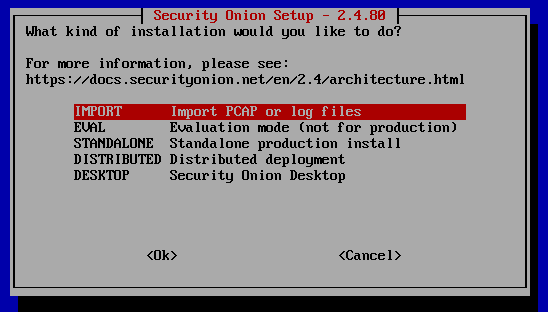
The manager node runs the following components:

* [Security Onion Console (SOC)](https://docs.securityonion.net/en/2.4/soc.html#soc)
* [Elasticsearch](https://docs.securityonion.net/en/2.4/elasticsearch.html#elasticsearch)
* [Logstash](https://docs.securityonion.net/en/2.4/logstash.html#logstash)
* [Kibana](https://docs.securityonion.net/en/2.4/kibana.html#kibana)
* [ElastAlert 2](https://docs.securityonion.net/en/2.4/elastalert.html#elastalert)
* [Redis](https://docs.securityonion.net/en/2.4/redis.html#redis)

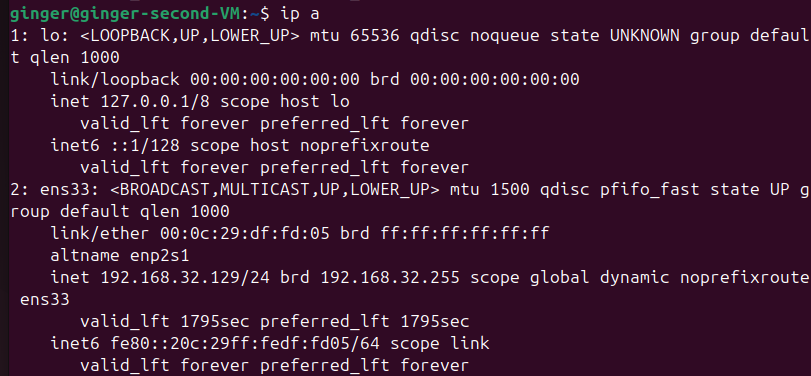
## Thực hành:

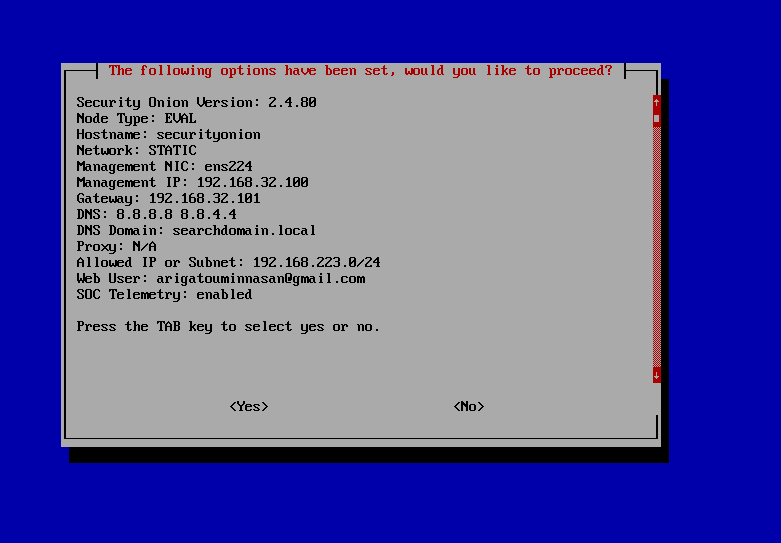


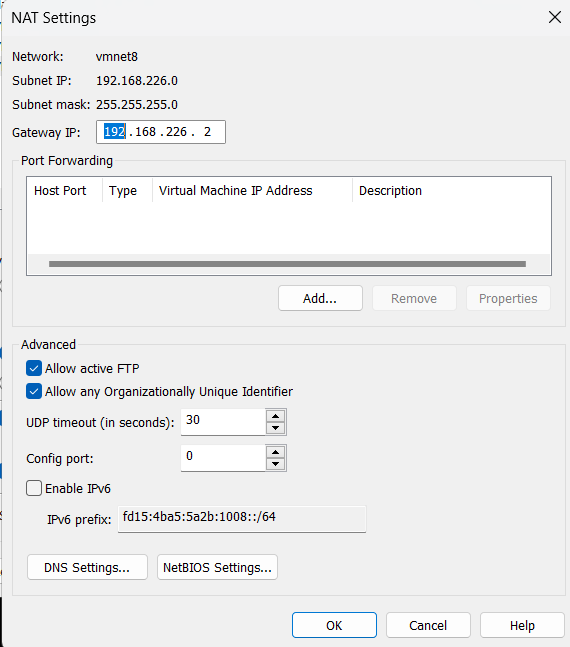




Hót only:





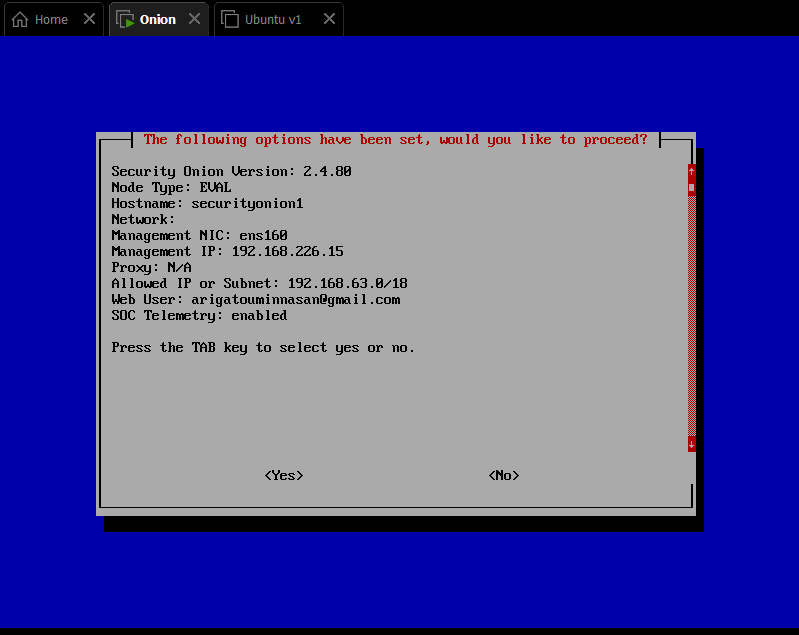


Dùng management: NAT

Monitor: Bridge (cho zui), Custom0







Cài desktop để dùng web browser, nh phải vào web browser duyệt ms đc dùng 🙂cái mèo gì thế



