Université IBN TOFAIL – Kénitra
ECOLE NATIONALE DES SCIENCES APPLIQUEES

Cours Sécurité des Systèmes Embarqués et IoT

CH6: Sécurisation du Cloud et des APIs IoT

Par:

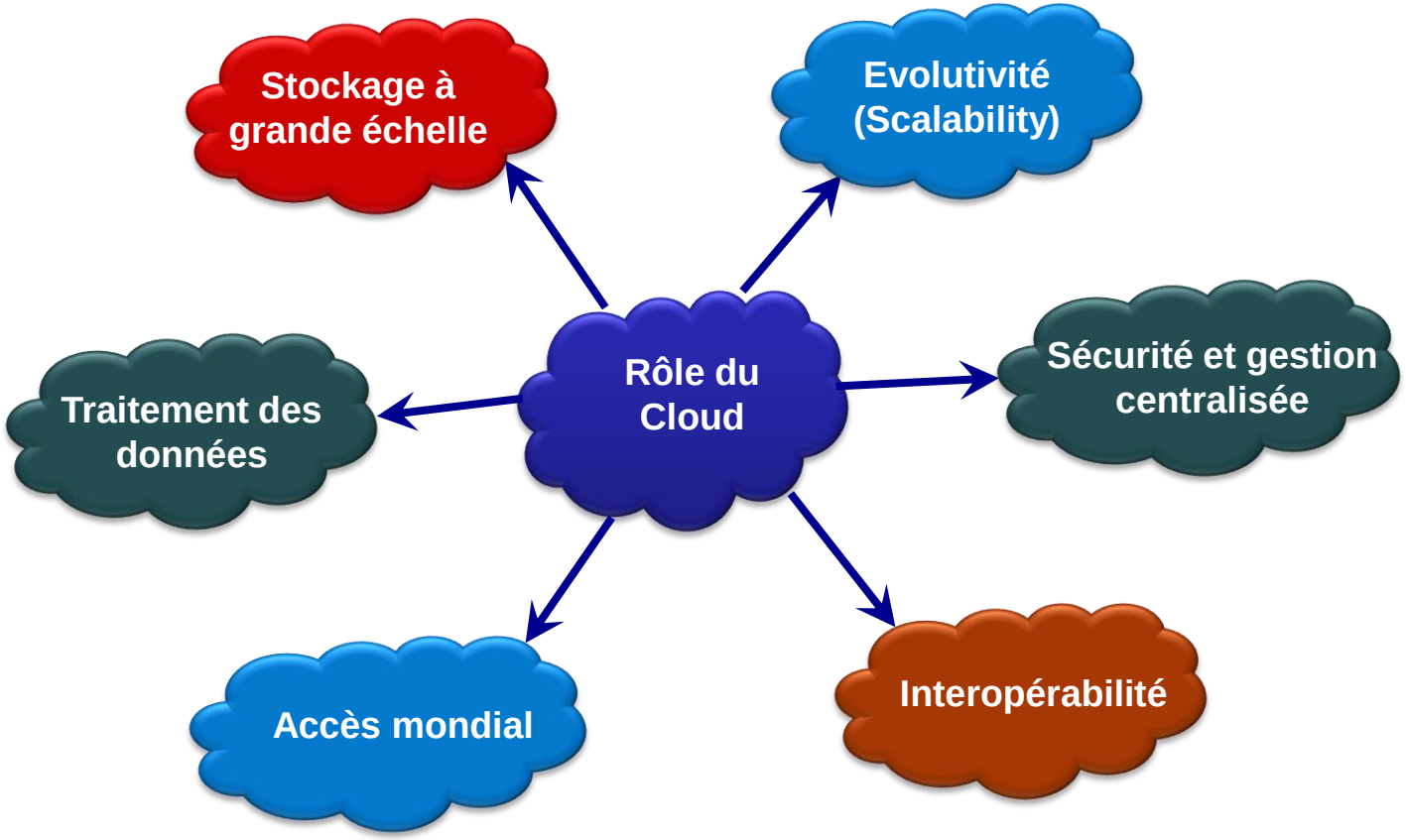
Hassan EL FADIL

elfadil.h@gmail.com

1. Introduction

1.1. Rôle du Cloud dans l'IoT:

- L'IoT implique une **énorme quantité de données** produites par des capteurs, des caméras, des actionneurs et d'autres dispositifs connectés.
- Le Cloud permet de gérer ces données efficacement:



1.2. Principaux enjeux de sécurité : CIAAN (Confidentiality, Integrity, Availability, Authentication, Non-repudiation)

Confidentialité (Confidentiality) :
- Assurer que seules les entités autorisées peuvent accéder aux données sensibles.
- Méthodes : Cryptographie (AES, RSA), protocoles sécurisés (TLS).

Intégrité (Integrity) :
- Garantir que les données ne sont ni altérées ni modifiées pendant leur stockage ou transmission.
- Méthodes : Hashage (SHA-256), signatures numériques.

Disponibilité (Availability) :
- Assurer que les systèmes sont accessibles et fonctionnels à tout moment, même en cas d'attaque.
- Méthodes : Tolérance aux pannes, protection contre les attaques DDoS.

Authentification (Authentication) :
- Vérifier l'identité des utilisateurs ou des dispositifs accédant au système.
- Méthodes : Mots de passe, biométrie, certificats numériques.

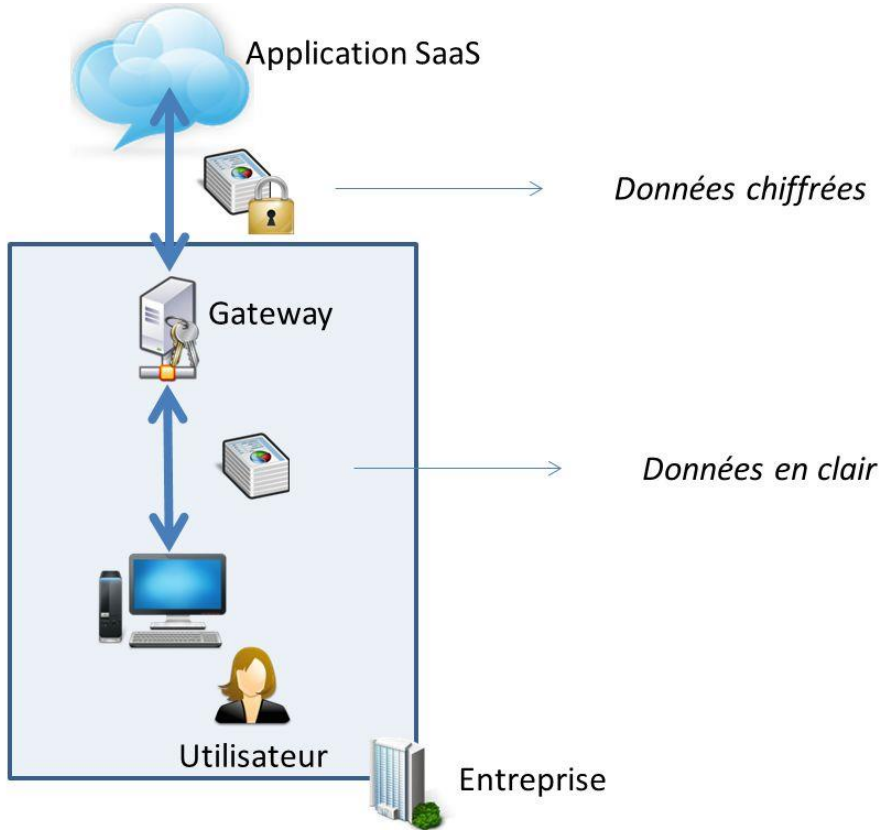
Non-répudiation (Non-repudiation) :
- Empêcher une entité d'annuler ou de nier une action effectuée (ex. un envoi de données).
- Méthodes : Journaux d'audit sécurisés, signatures numériques.

2. Stratégies de Sécurisation du Cloud IoT

2.1. Chiffrement des données :

Les données circulant entre les objets connectés et le Cloud doivent être protégées par un chiffrement robuste pour empêcher leur interception.

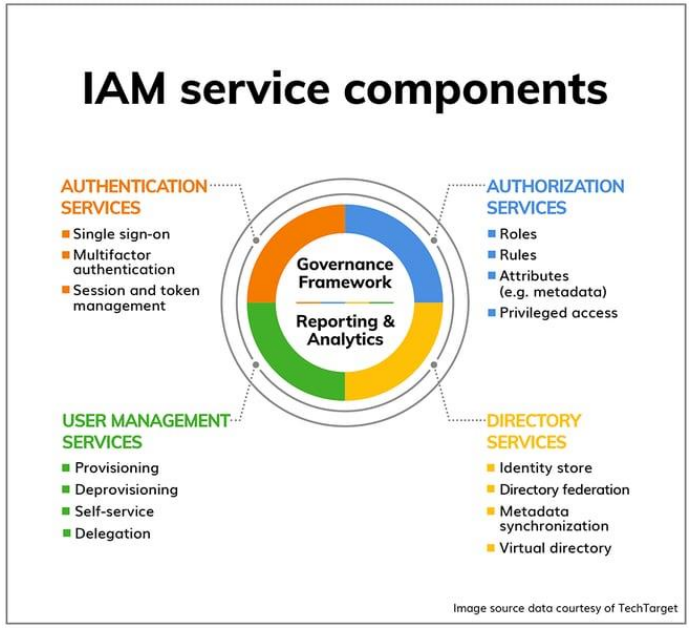
- **Chiffrement en transit** :
  - Utilisation du **TLS 1.3** pour sécuriser les échanges HTTP, MQTT, et autres protocoles IoT.
  - VPN (Virtual Private Network) et tunnels sécurisés (IPSec) pour les communications sensibles.
- **Chiffrement au repos** :
  - Stockage chiffré avec **AES-256** dans les bases de données Cloud (AWS KMS, Azure Key Vault).
  - Gestion sécurisée des clés avec **HSM (Hardware Security Module)**.



2.2. Authentification et gestion des accès (IAM):

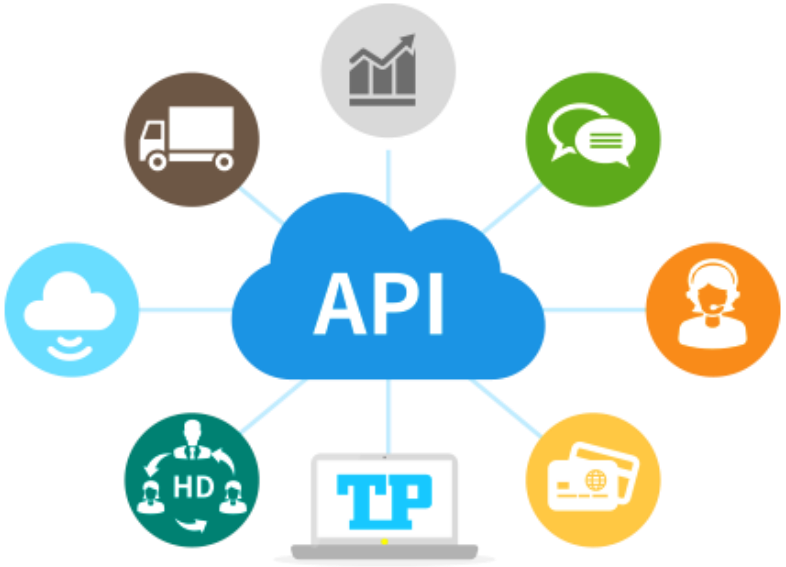
Il est essentiel de mettre en place une **gestion stricte des identités et des accès**.

- **Authentification forte** :
  - **MFA (Multi-Factor Authentication)** pour protéger les comptes administrateurs.
  - Utilisation de **certificats X.509** pour identifier les objets connectés.
- **Contrôle des permissions (Zero Trust)** :
  - Principe du **moindre privilège** : chaque appareil ou utilisateur n'accède qu'aux ressources nécessaires.
  - **IAM (Identity and Access Management)** pour restreindre l'accès aux services Cloud (AWS IAM, Azure AD).



2.3. Sécurisation des APIs IoT:

- Les **APIs** sont la principale interface entre les objets connectés et le Cloud.
- Elles doivent être protégées contre les attaques telles que l'injection SQL, le vol de tokens d'authentification et les attaques DDoS.
- **Bonnes pratiques de sécurité des APIs** :
  - **Utilisation d'OAuth 2.0 et JWT** pour l'authentification des API.
  - **Rate limiting** pour limiter le nombre de requêtes et prévenir les attaques DDoS.
  - **Web Application Firewall (WAF)** pour bloquer les requêtes malveillantes.
  - **Chiffrement des communications** via HTTPS/TLS



- ***API: intermédiaire** entre une application (client) et un service (serveur).*
- *Reçoit des **requêtes** (demandes), traite les données, et renvoie des **réponses**.*

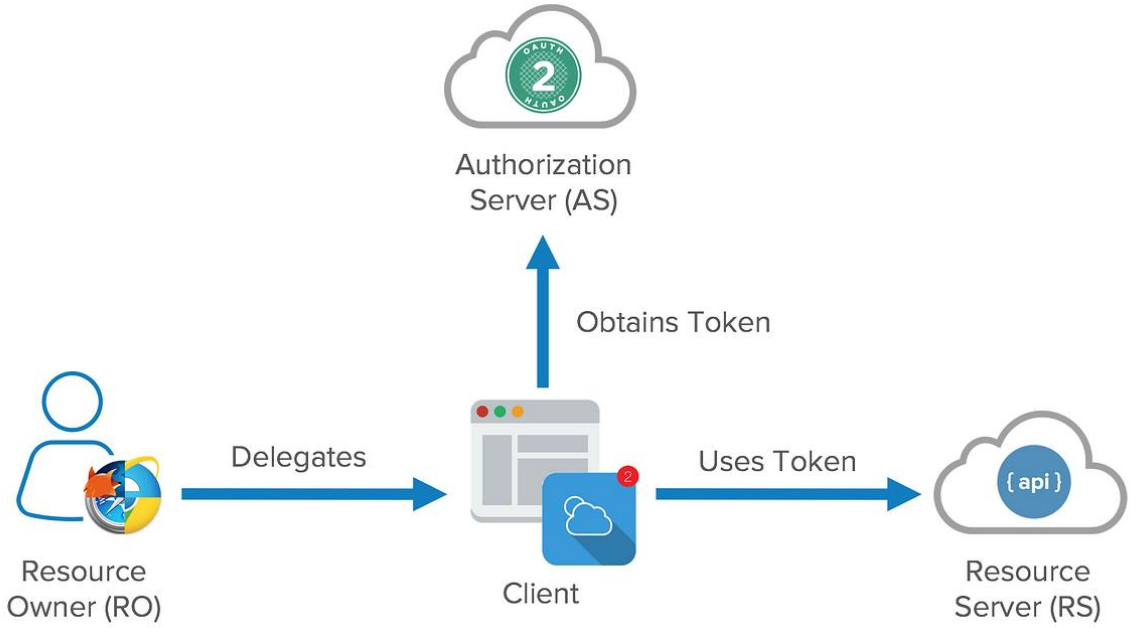
2.3.1. OAuth 2.0 : Protocole d'Autorisation Sécurisée:

a) Rôle d'OAuth 2.0 :

- **Protocole standard d'autorisation** qui permet à une application (client) d'accéder aux ressources d'un utilisateur sur un service tiers (ex: API, Cloud, IoT) **sans exposer ses identifiants**.
- Il est largement utilisé pour sécuriser l'accès aux **APIs IoT, applications web et mobiles**.
- **Avantage :** L'application n'a jamais besoin de stocker ou gérer le mot de passe de l'utilisateur !
- **Exemple** : Une caméra IoT qui envoie ses données sur un serveur cloud peut utiliser OAuth 2.0 pour s'authentifier sans partager directement son mot de passe.

b) Acteurs Principaux d'OAuth 2.0 :

- **Client** : Application qui demande l'accès (ex: un appareil IoT, une application mobile).
- **Resource Owner (propriétaire de la ressource)**: L'utilisateur ou l'appareil qui possède les données.
- **Authorization Server (serveur d'autorisation)** : Vérifie l'identité du client et délivre un **jeton d'accès (Access Token)**.
- **Resource Server (serveur de ressources)** : Stocke les données protégées et accepte uniquement les requêtes authentifiées via un **token OAuth 2.0**.

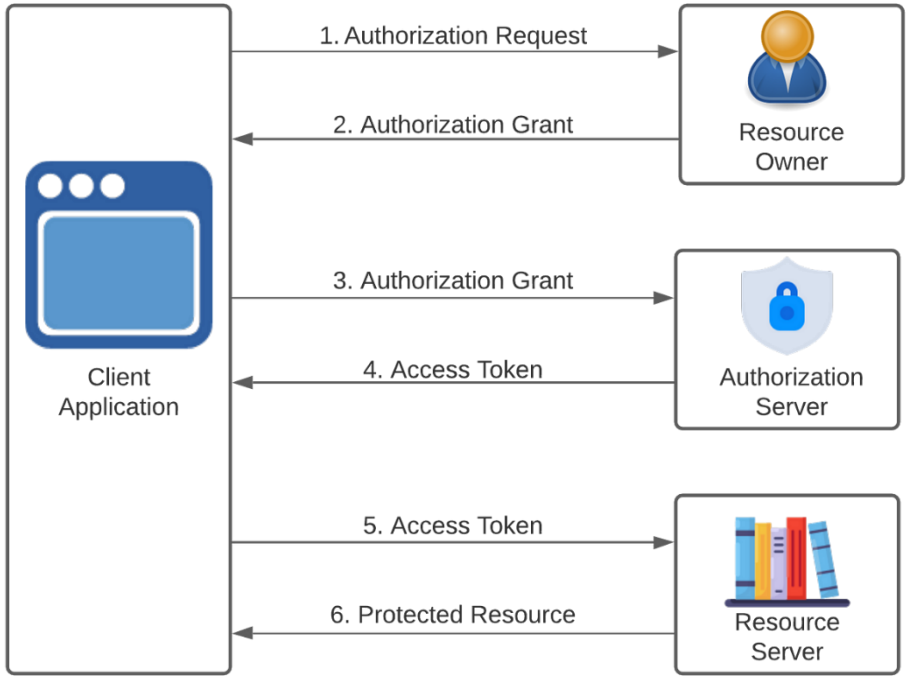


Resource Owner:
- Un **utilisateur humain** (ex: un utilisateur qui donne accès à son compte Google pour une application tierce).
- Un **appareil IoT** (ex: un thermostat intelligent, une caméra connectée, …).

Resource Server:
- Un **serveur cloud** (ex: un serveur AWS, Google Cloud, Azure qui stocke des données IoT).
- Une **API** qui expose des ressources protégées (ex: API REST d'un service IoT).

c) Fonctionnement d'OAuth 2.0 :

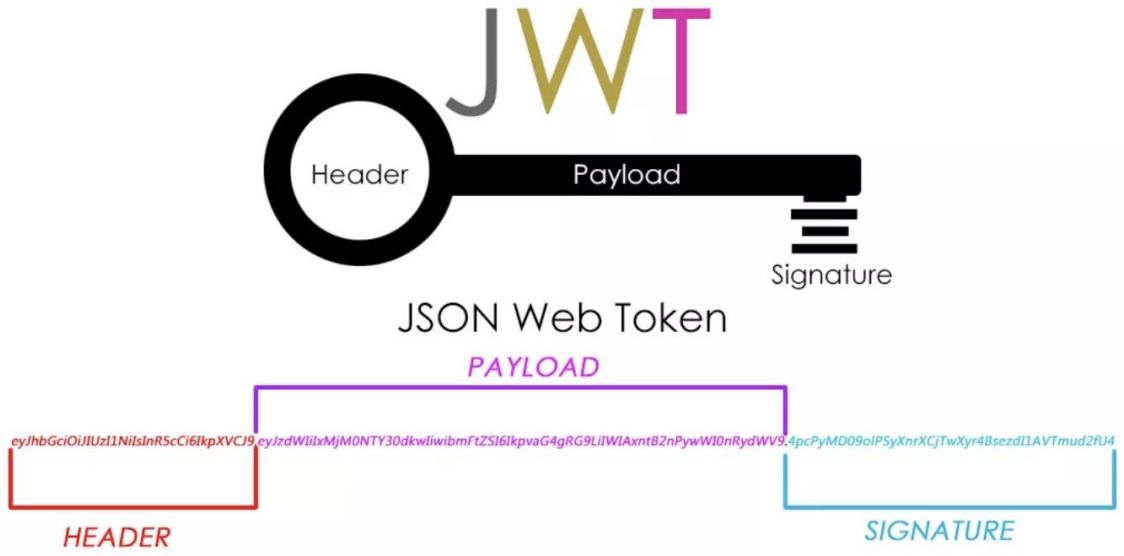
1. **Authorization Request**: **Le Client Application** envoie une requête d'autorisation au **Resource Owner**. (Ex. Une application mobile demande l'autorisation d'accéder aux photos de l'utilisateur.)
2. **Authorization Grant**: **Le Resource Owner** accepte ou refuse la demande. Si accepté, il retourne un **Authorization Grant** (jeton d'autorisation) au **Client Application**.
3. **Token Request**: **Le Client Application** envoie l'**Authorization Grant** au **Authorization Server** et demande un **Access Token**.
4. **Access Token Response**: Le **Authorization Server** valide l'autorisation et délivre un **Access Token** (et parfois un **Refresh Token**).
5. **Resource Request**: **Le Client Application** envoie une requête au **Resource Server**, en incluant le **Access Token** dans l'en-tête de la requête.
6. **Resource Response**: **Le Resource Server** vérifie le token et, s'il est valide, renvoie les données demandées au **Client Application**.



2.3.2. JSON Web Token (JWT):

a) Définition de JWT:

- Un **JSON Web Token (JWT)** est un standard ouvert (**RFC 7519**) permettant d'échanger des informations de manière sécurisée entre deux parties sous forme d'un **token JSON signé**.
- Il est principalement utilisé pour l'**authentification** et l'**autorisation** dans les applications web et mobiles.
- Il est souvent utilisé avec **OAuth 2.0**, les **APIs IoT**, et les **systèmes distribués**



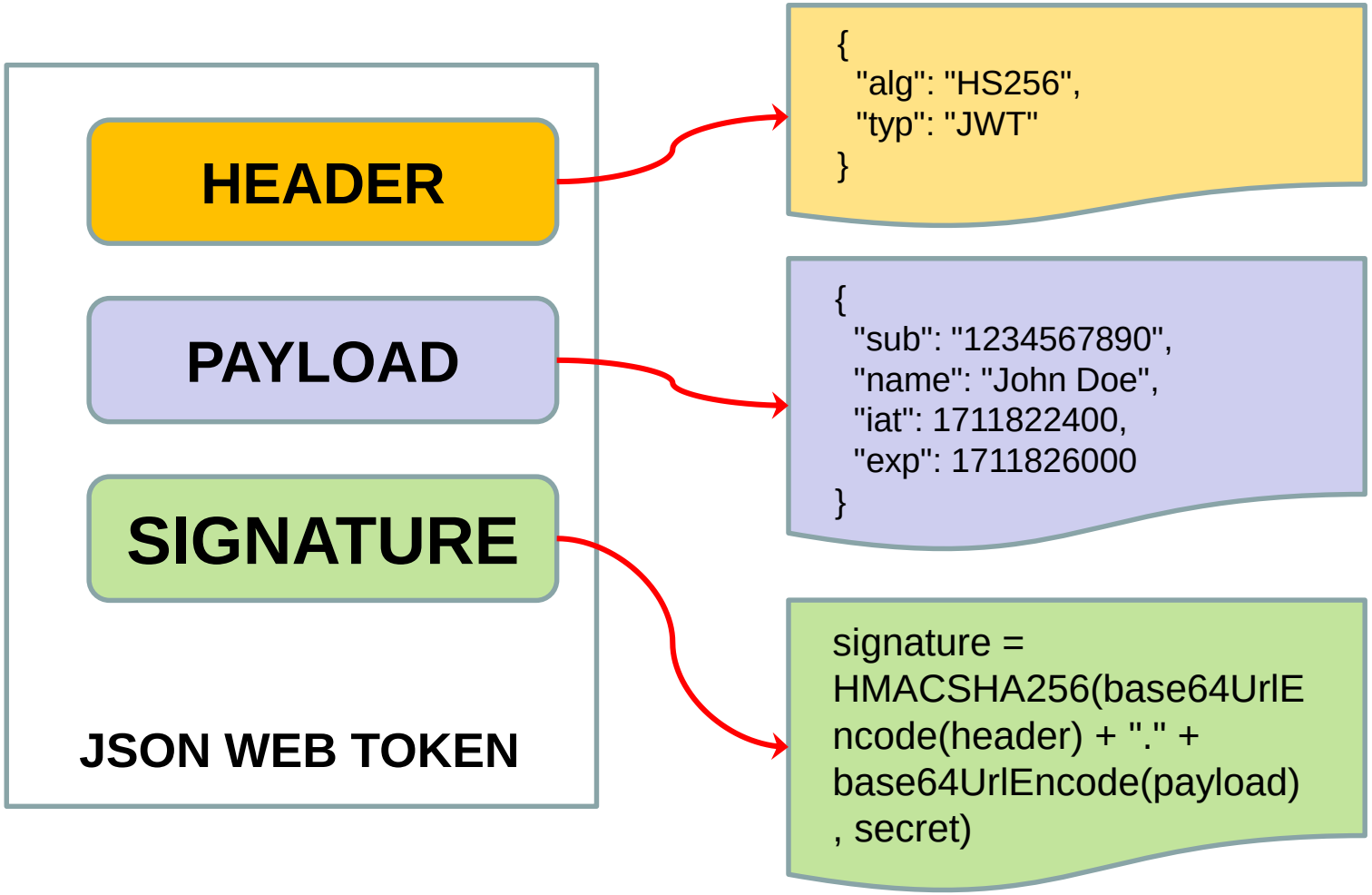
b) Structure de JWT:

Un JWT est composé de **trois parties** séparées par des points (.) :

xxxxx.yyyyy.zzzzz

Header Payload Signature

Chaque partie est encodée en **Base64URL**



c) Base64URL:

- Base64 est un système d'encodage binaire-texte qui permet de représenter des données binaires sous forme de texte en utilisant un ensemble de 64 caractères.
- Il est souvent utilisé pour transmettre des données binaires (comme des images, des fichiers ou des clés cryptographiques) dans des formats textuels, tels que les emails (MIME), les URL ou les JSON.

Alphabet Base64 : 64 caractères

ABCDEFGHIJKLMNOPQRSTUVWXYZ (26 lettres majuscules)
abcdefghijklmnopqrstuvwxyz (26 lettres minuscules)
0123456789 (10 chiffres)
+ / (2 caractères spéciaux)

Si la longueur des données d'origine n'est pas un multiple de 3 octets, un remplissage (=) est ajouté à la fin.

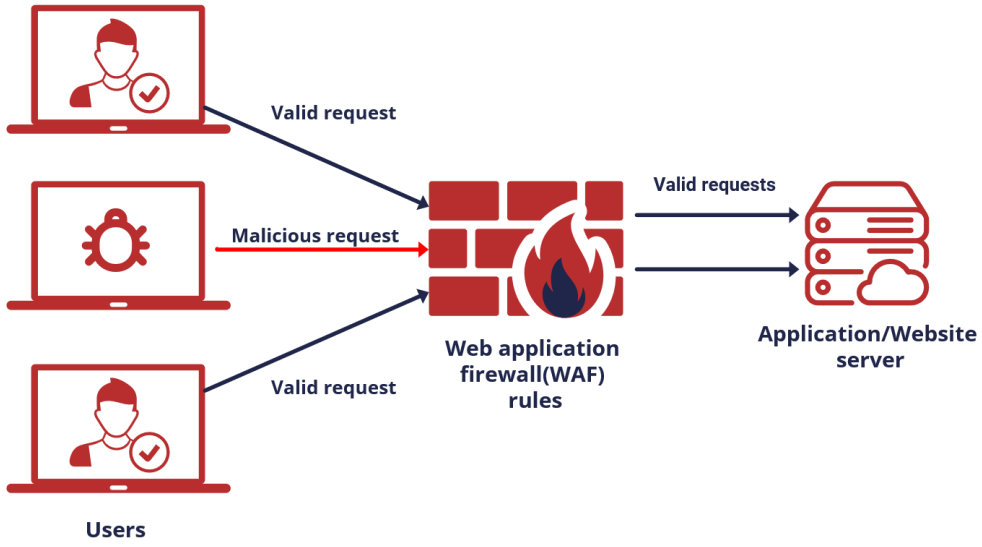
Base64URL: **+ et /** sont remplacés par **– et** _ car ils peuvent être mal interprétés dans une URL.

Exemple en Python avec JSON et Base64URL:

```
import json
import base64

# JSON à encoder
data = {"Année": 2025, "Nom du Master": "ISE"}
json_string = json.dumps(data)  # Convertir JSON en string

# Encodage en Base64URL
encoded = base64.urlsafe_b64encode(json_string.encode()).decode()
print("Base64URL Encodé:", encoded)

# Décodage en JSON
decoded_json = json.loads(base64.urlsafe_b64decode(encoded).decode())
print("Décodé JSON:", decoded_json)
```

2.3.3. Web Application Firewall (WAF) :

- Le WAF: barrière entre l'utilisateur et l'application web.
- Il analyse chaque requête HTTP entrant et sortant.
- Les principales méthodes de filtrage utilisées par un WAF incluent :
  - **Basé sur les signatures** : Détection des attaques connues en comparant les requêtes aux signatures de menaces répertoriées.
  - **Basé sur le comportement (anomalie)** : Surveillance des modèles de trafic pour détecter les comportements anormaux.
  - **Basé sur le modèle positif (whitelisting)** : Autorisation uniquement des requêtes conformes aux règles de l'application.



Exemples de solutions WAF:

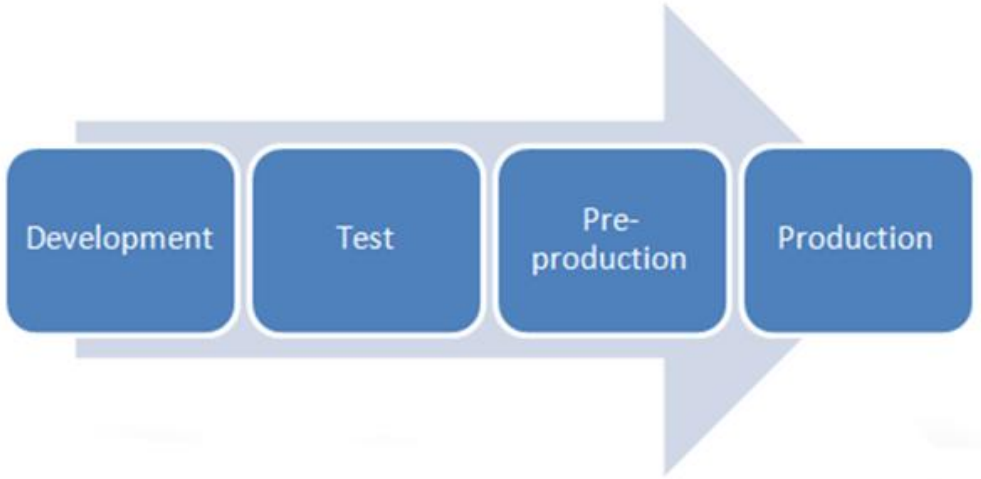
- **ModSecurity** : un WAF open-source utilisé avec Apache, Nginx et IIS.
- **Cloudflare WAF** : une solution cloud efficace contre diverses menaces.
- **AWS WAF** : une solution intégrée aux services cloud d'Amazon.
- **Imperva WAF** : une solution avancée offrant une protection en temps réel.

2.4. Séparation des environnements Cloud

2.4.1. Types d'environnements cloud:

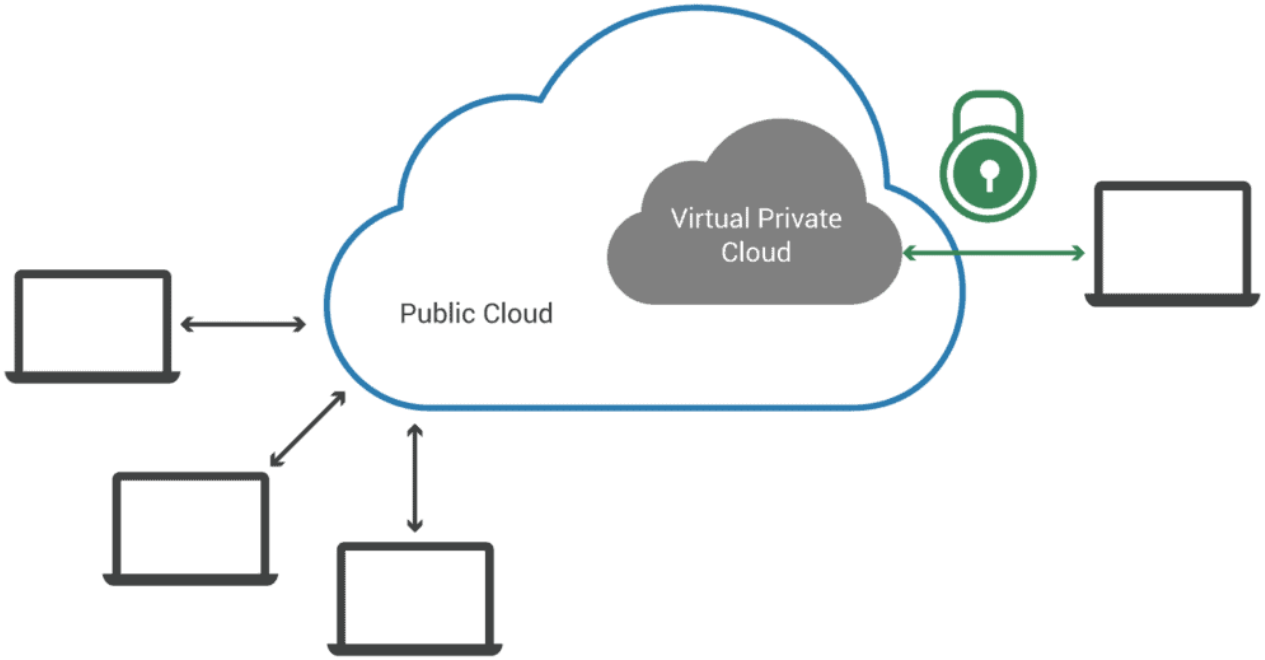
- La séparation des environnements Cloud est une **pratique essentielle** pour assurer la **sécurité, la stabilité et la gestion efficace** des ressources.

- **Types d'environnements Cloud**
  - **Développement (Dev)** : utilisé par les développeurs pour coder et tester.
  - **Test (Staging)** : réplique de production pour valider les fonctionnalités.
  - **Préproduction (Pre-Prod)** *(optionnel)* : dernière validation avant mise en ligne.
  - **Production (Prod)** : environnement final utilisé par les utilisateurs.



2.4.2. Méthodes de séparation:

- **Physique** : serveurs distincts.
- **Logique** : machines virtuelles ou conteneurs (Docker, Kubernetes).
- **Par comptes/projets** : différents comptes AWS, GCP, Azure.
- **Par réseau** : VPC (Virtual Private Cloud) pour isoler les ressources.



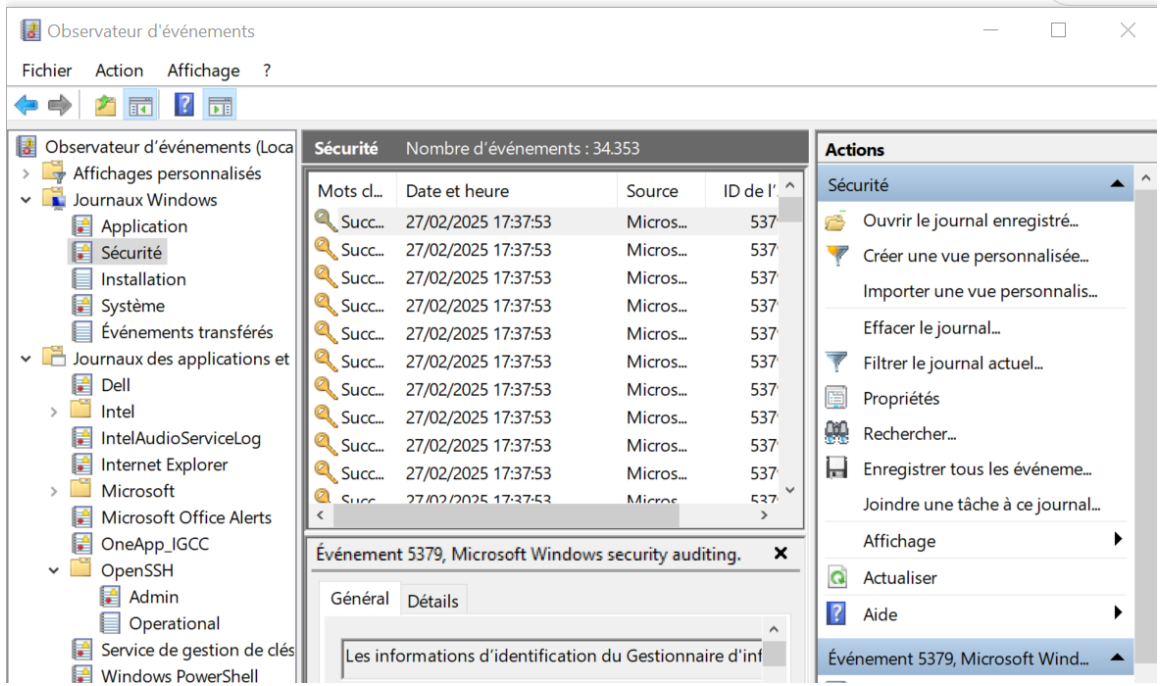
2.5.1. Surveillance continue et gestion des logs:

- Un **log** (ou journal d'événements) est un **fichier ou un enregistrement** contenant des informations sur les activités et événements d'un système, d'une application ou d'un réseau.
- Un **Cloud sécurisé** doit être capable **d'anticiper, détecter et réagir rapidement** aux menaces pour protéger les données et les infrastructures.
- La **surveillance continue** est essentielle pour détecter les activités suspectes en temps réel et prévenir les attaques.

- **Activation des logs d'accès et des événements Cloud** :
  - Les fournisseurs Cloud offrent des outils de journalisation comme **AWS CloudTrail**, **Azure Security Center** ou **Google Cloud Audit Logs** pour enregistrer toutes les activités et détecter les anomalies.
  - Ces logs permettent d'identifier les accès non autorisés, les tentatives d'intrusion et les modifications suspectes.

Remarque:

L'observateurs d'événements Windows (Event Viewer) permet de surveiller et d'analyser les journaux d'événements générés par le système d'exploitation, les applications et les services.

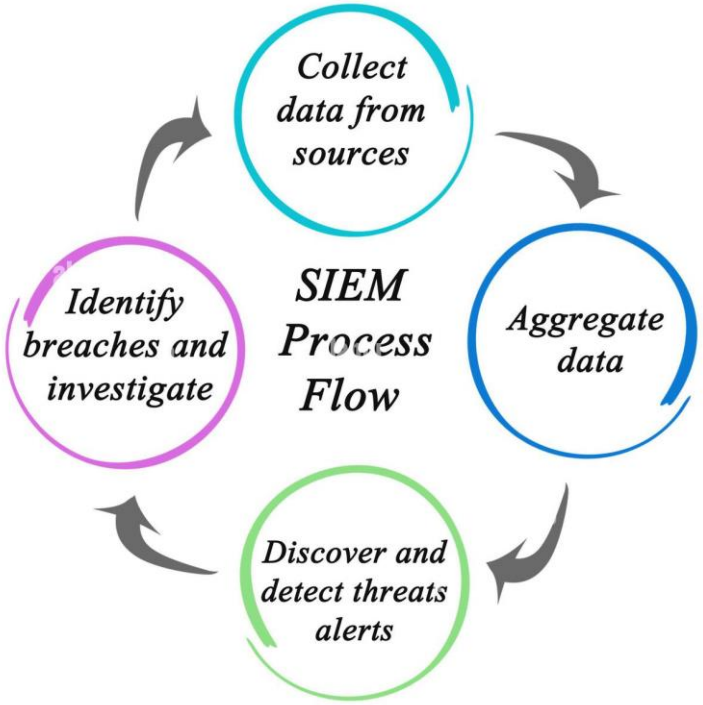


- **Win + R → taper eventvwr.msc → Entrée**
- Ou via **Panneau de configuration → Outils d'administration → Observateur d'événements**.

2.5.2. Analyse des journaux avec des SIEM:

Les solutions **SIEM (Security Information and Event Management)** comme **Splunk**, **IBM QRadar** ou **Microsoft Sentinel** permettent :

- De collecter et analyser les logs en temps réel.
- De détecter des modèles suspects (ex. tentatives de connexion répétées).
- D'alerter les équipes de sécurité en cas d'attaque potentielle.



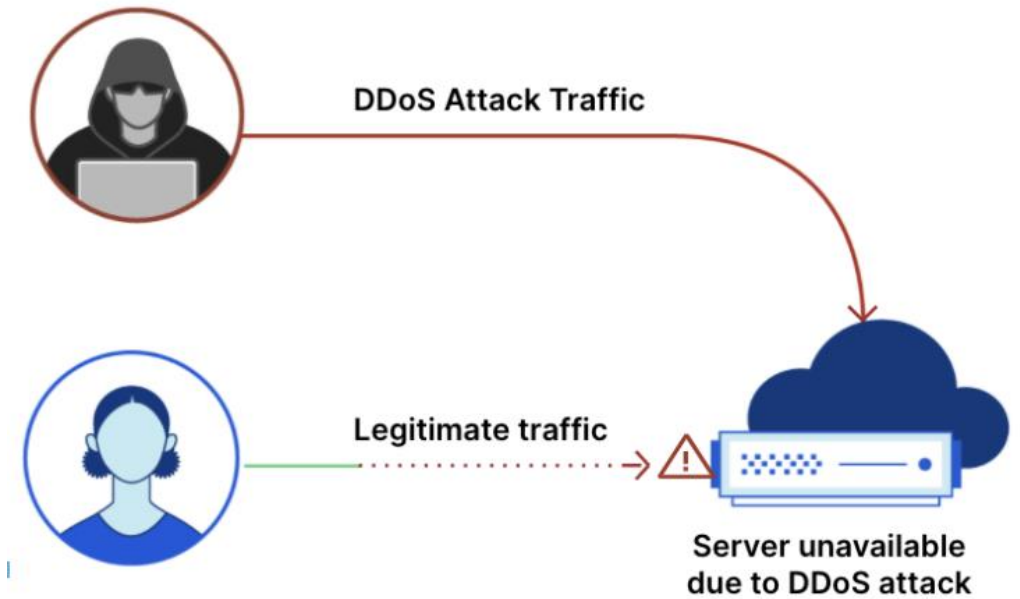
splunk>

IBM QRadar

Microsoft Sentinel

2.6. Protection contre les attaques DDoS

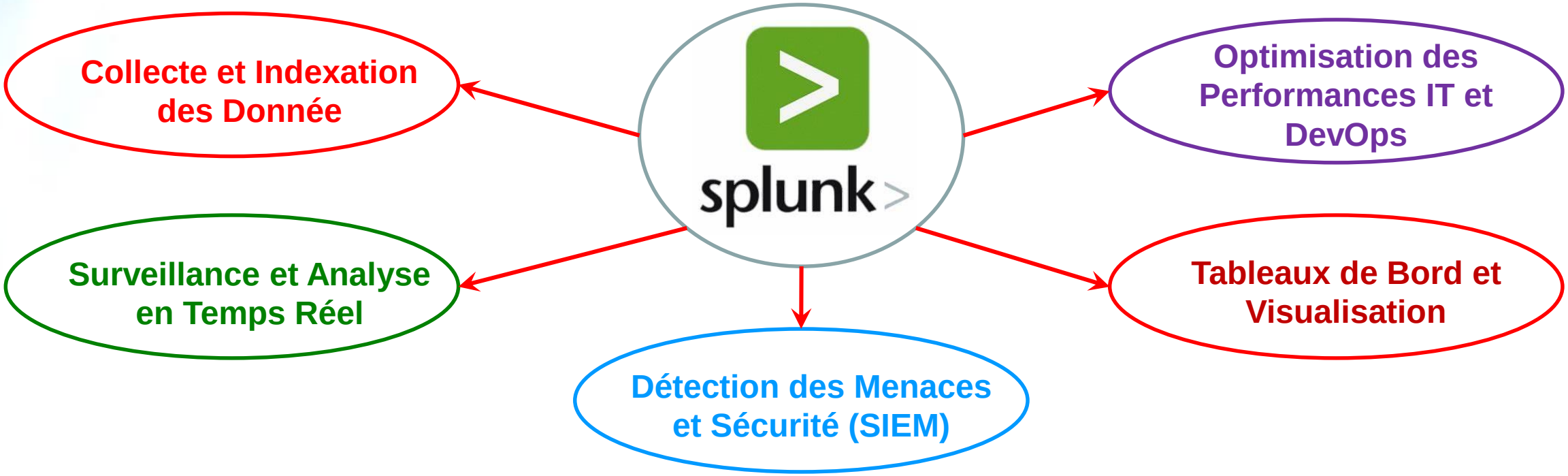
- Une **attaque par déni de service (DoS - Denial of Service)** vise à **surcharger un serveur, un réseau ou un service** pour le rendre **indisponible** aux utilisateurs légitimes.
- Distributed Denial of Service (**DDoS**): utilise **plusieurs sources** (souvent un réseau de machines infectées appelées **botnet**) pour submerger une cible.
- **Solutions anti-DDoS** :
  - Activation de **Cloudflare, AWS Shield, Azure DDoS Protection**.
  - Configuration de **limites de requêtes (Rate Limiting)** sur les APIs IoT.
  - Filtrage des adresses IP suspectes via **firewall Cloud**.



Application 1: Collecte et analyse de logs avec un SIEM open-source (Splunk)

1. Qu'est-ce que Splunk ?

- Splunk est une plateforme logicielle utilisée pour **collecter, analyser et visualiser** des données issues de différentes sources, en particulier les **logs et événements** générés par les systèmes informatiques



2. Installation de Splunk:

1. **Télécharge Splunk (version d'éssai):**

https://www.splunk.com/en_us/download.html

2. **Regarder la vidéo ci-dessous pour les étapes d'installation et d'utilisation:**

https://www.youtube.com/watch?v=4D-vq481gug

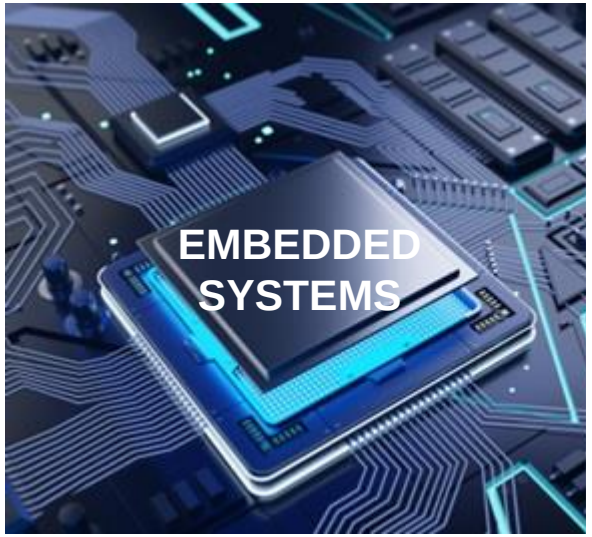
Application 2 : Surveillance des logs Cloud avec AWS CloudTrail

1. Qu'est-ce que AWS CloudTrail ?

- **AWS CloudTrail** est un service qui enregistre l'activité des utilisateurs et des API dans un compte AWS pour **la surveillance, l'audit et la sécurité**.

2. Comment l'utiliser? Voir vidéo ci-dessous:

https://www.youtube.com/watch?v=Vgi3ukxc1n0



FIN !
Questions ?