

Projet : Analyse des Avis et Alertes ANSSI avec Enrichissement des CVE

1. Contexte et Objectifs

La **cybersécurité** est devenue un enjeu majeur pour les entreprises et les organisations du monde entier face à la multiplication des attaques informatiques. Les vulnérabilités logicielles et matérielles constituent une **porte d'entrée privilégiée** pour les attaquants, rendant impératif leur **identification rapide** et leur **correction efficace**.

En France, l'**Agence Nationale de la Sécurité des Systèmes d'Information (ANSSI)** joue un rôle central dans la veille et la diffusion d'informations sur les menaces. Elle publie régulièrement des **bulletins de sécurité** ([CERT-FR – Centre gouvernemental de veille, d'alerte et de réponse aux attaques informatiques](#)) visant à informer les entreprises et les particuliers sur les vulnérabilités existantes et les risques associés. Les principaux types de bulletins (sujets de notre projet) sont :

- **Les avis de sécurité** : Ils signalent des vulnérabilités **connues** et fournissent des **recommandations concrètes** pour les corriger ou atténuer leurs effets. Ils permettent aux organisations d'agir préventivement pour sécuriser leurs systèmes.
- **Les alertes** : Elles concernent les vulnérabilités **critiques** qui sont activement **exploitées** par des acteurs malveillants. Ces vulnérabilités nécessitent une **intervention urgente** pour éviter d'éventuelles compromissions de sécurité.

Ces bulletins contiennent des identifiants **CVE (Common Vulnerabilities and Exposures)** qui permettent de référencer précisément chaque vulnérabilité.

Contrairement à son homologue américain, le **NIST (National Institute of Standards and Technology)** qui propose une **API dédiée et complète** ([NVD API](#)) permettant de collecter et d'analyser facilement ce type d'informations, il est plus difficile d'automatiser le traitement des flux publiés par l'ANSSI. En effet, l'ANSSI met uniquement à disposition un **flux RSS relativement sommaire** ([ANSSI RSS Feed](#)) destiné aux entreprises et aux particuliers. Les informations détaillées nécessitent de naviguer dans le **DOM des pages web** ou directement le **JSON** mentionnées dans ces flux pour être extraites et exploitées.

De plus, contrairement au NIST qui offre des fonctionnalités avancées, telles que des **interprétations statistiques automatisées** ou des **systèmes d'alertes personnalisés**, l'ANSSI ne permet pas, dans son format actuel, de générer des **statistiques avancées** ni des **alertes sur mesure** en fonction des besoins spécifiques des utilisateurs. Ce manque d'automatisation et de flexibilité justifie pleinement la réalisation d'un outil capable de traiter, d'enrichir et d'analyser ces données pour en tirer des conclusions exploitables.

Objectifs du Projet

- Extraire les données des flux RSS des avis et alertes ANSSI.
- Identifier les CVE mentionnées dans les bulletins.
- Enrichir les CVE avec des informations complémentaires via des API externes.
- Consolider les données dans un format exploitable (DataFrame pandas).
- Analyser et visualiser le DataFrame obtenu (vulnérabilités critiques, scores...)
- Générer des alertes personnalisées pour les produits affectés et envoyer des notifications par email.

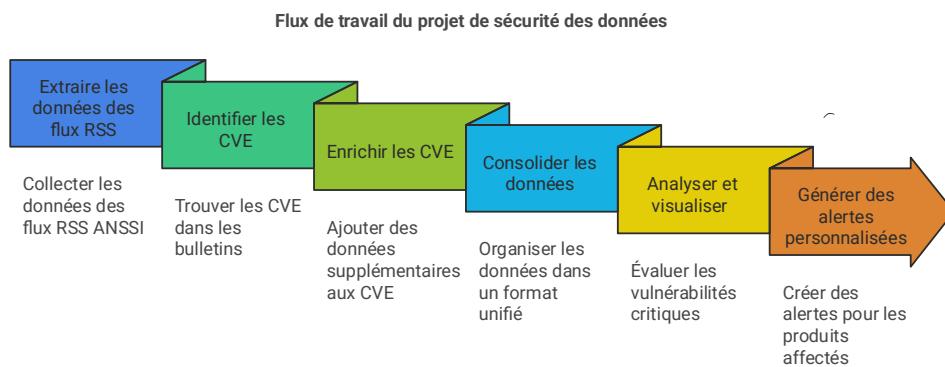
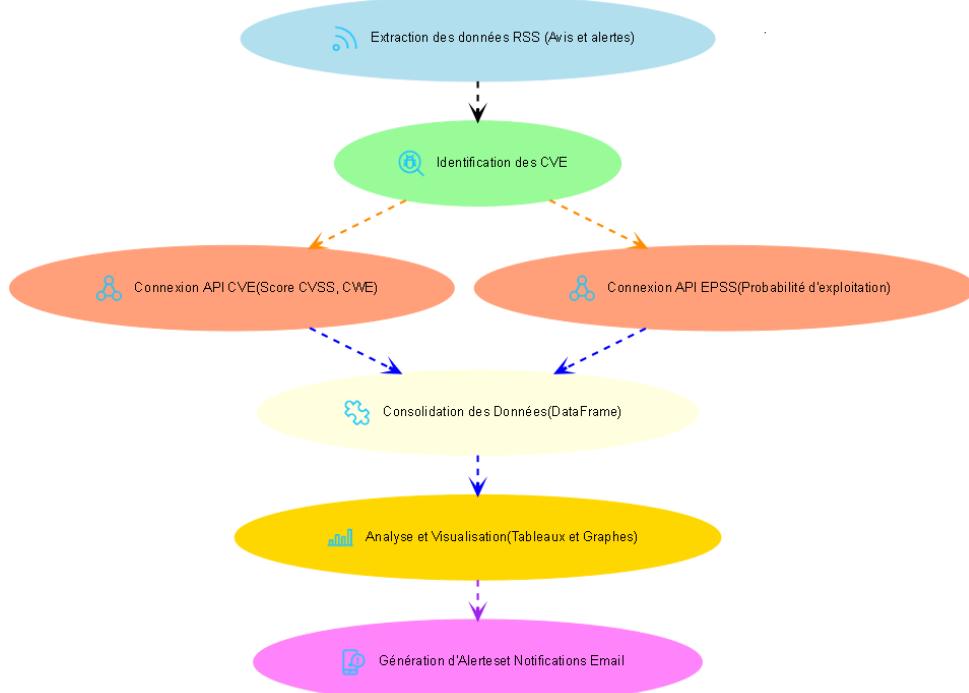


Figure 1

Flux de travail du projet générer par <https://www.napkin.ai/>

2. Déroulement du Projet



Étape 1 : Extraction des Flux RSS

Les flux RSS des avis et alertes de l'ANSSI sont extraits grâce à la bibliothèque `feedparser` ou une autre de votre choix. L'extraction permet d'obtenir les informations suivantes : Titre, Description, Date de publication et Lien vers le bulletin détaillé.

Exemple de code pour l'extraction des flux RSS :

```
import feedparser  
url = "https://www.cert.ssi.gouv.fr/avis/feed"  
rss_feed = feedparser.parse(url)  
  
for entry in rss_feed.entries:  
    print("Titre :", entry.title)  
    print("Description:", entry.description)  
    print("Lien :", entry.link)  
    print("Date :", entry.published)
```

Étape 2 : Extraction des CVE

Une fois les liens des bulletins obtenus, nous accédons aux fichiers JSON pour identifier les CVE mentionnées.

Vous remarquez que le lien du JSON s'obtient en ajoutant tout simplement "JSON" au line de l'alerte ou de l'avis de l'ANSSI.

Exemple de code :

```
import requests  
import re  
  
url = "https://www.cert.ssi.gouv.fr/alerte/CERTFR-2024-ALE-001/json/"  
response = requests.get(url)  
data = response.json()  
#Extraction des CVE reference dans la clé cves du dict data  
  
ref_cves=list(data["cves"])  
#attention il s'agit d'une liste des dictionnaires avec name et url comme clés  
print( "CVE référencés ", ref_cves)  
  
# Extraction des CVE avec une regex  
cve_pattern = r"CVE-\d{4}-\d{4,7}"  
cve_list = list(set(re.findall(cve_pattern, str(data))))  
print("CVE trouvés :", cve_list)
```

Étape 3 : Enrichissement des CVE

Les identifiants CVE extraits sont enrichis grâce à des API externes :

- API CVE de MITRE : Permet d'obtenir le score CVSS et le type CWE associé.
- API EPSS de FIRST : Permet d'obtenir la probabilité d'exploitation de la vulnérabilité.

Le score CVSS (Common Vulnerability Scoring System) :

Une note de **0 à 10** qui indique la **gravité** d'une vulnérabilité.

- **0-3** : Faible
- **4-6** : Moyenne
- **7-8** : Élevée
- **9-10** : Critique

Le type de vulnérabilité CWE (Common Weakness Enumeration) :

- Il décrit la **nature de la faille** (par exemple : erreur d'accès mémoire, injection SQL, etc.).

Le score EPSS (Exploit Prediction Scoring System) :

- Il estime la **probabilité** qu'une vulnérabilité soit exploitée par des attaquants.
- Une valeur entre **0 et 1** (plus le score est proche de 1, plus la faille est dangereuse).

Exemple de connexion à l'API CVE :

```
import requests
cve_id = "CVE-2023-24488"
url = f"https://cveawg.mitre.org/api/cve/{cve_id}"
response = requests.get(url)
data = response.json()

# Extraire la description
description = data["containers"][0]["cna"]["descriptions"][0]["value"]
# Extraire le score CVSS
#ATTENTION tous les CVE ne contiennent pas nécessairement ce champ, gérez l'exception,
#ou peut être au lieu de cvssV3_0 c'est cvssV3_1 ou autre clé
cvss_score = data["containers"][0]["cna"]["metrics"][0]["cvssV3_1"]["baseScore"]

cwe = "Non disponible"
cwe_desc="Non disponible"
problemtype = data["containers"][0]["cna"].get("problemTypes", {})
if problemtype and "descriptions" in problemtype[0]:
    cwe = problemtype[0]["descriptions"][0].get("cweld", "Non disponible")
    cwe_desc=problemtype[0]["descriptions"][0].get("description", "Non disponible")

# Extraire les produits affectés
affected = data["containers"][0]["cna"]["affected"]
for product in affected:
    vendor = product["vendor"]
    product_name = product["product"]
    versions = [v["version"] for v in product["versions"] if v["status"] == "affected"]
    print(f"Éditeur : {vendor}, Produit : {product_name}, Versions : {', '.join(versions)}")

# Afficher les résultats
print(f"CVE : {cve_id}")
print(f"Description : {description}")
print(f"Score CVSS : {cvss_score}")
print(f"Type CWE : {cwe}")
print(f"CWE Description : {cwe_desc}")
```

Exemple de connexion à l'API EPSS:

```
import requests
# URL de l'API EPSS pour récupérer la probabilité d'exploitation
cve_id = "CVE-2023-46805"
url = f"https://api.first.org/data/v1/epss?cve={cve_id}"
# Requête GET pour récupérer les données JSON

response = requests.get(url)
data = response.json()
# Extraire le score EPSS
epss_data = data.get("data", [])
if epss_data:
    epss_score = epss_data[0]["epss"]
    print(f"CVE : {cve_id}")
    print(f"Score EPSS : {epss_score}")
else:
    print(f"Aucun score EPSS trouvé pour {cve_id}")
```

Étape 4 : Consolidation des Données

Les informations extraites et enrichies sont consolidées dans un tableau (DataFrame Pandas) avec les colonnes suivantes (à titre non exhaustif) :

- **Titre du bulletin (ANSSI)** : Titre de l'avis ou de l'alerte.
- **Type de bulletin** : Avis ou Alerte.
- **Date de publication** : Date de publication du bulletin ANSSI.
- **Identifiant CVE** : Identifiant unique de la vulnérabilité.
- **Score CVSS** : Score de criticité selon le standard CVSS.
- **Base Severity** : Gravité associée au score CVSS (Exemple : **Critique, Élevée**).
- **Type CWE** : Catégorie de la vulnérabilité (Exemple : CWE-287).
- **Score EPSS** : Probabilité d'exploitation de la vulnérabilité.
- **Lien du bulletin (ANSSI)** : URL vers le bulletin original.
- **Description** : Détails sur la vulnérabilité (issue des API).
- **Éditeur/Vendor** : Nom de l'éditeur du produit affecté.
- **Produit** : Nom du produit concerné (ex : Apache HTTP Server).
- **Versions affectées** : Liste des versions impactées.

Exemple du DataFrame (Attention ceci est un exemple simpliste, à vous de l'approfondir)

ID ANSSI	Titre ANSSI	Type	Date	CVE	CVSS	Base Severity	CWE	EPSS	Lien	Description	Éditeur	Produit	Versions affectées
CERTFR-2024-ALE-001	Multiples vulnérabilités dans Ivanti	Alerte	2024-01-11	CV-E-2024-22024	8.3	High	CWE-287 (Authentication Bypass)	0.85	https://www.cert.ssi.gouv.fr/...	An authentication bypass vulnerability...	Ivanti	ICS	9.1R18, 22.6R2
CERTFR-2024-ALE-001	Multiples vulnérabilités dans Ivanti	Alerte	2024-01-11	CV-E-2023-46805			CVE-2024-21888	0.64	
CERTFR-2024-AVI-0128	Multiples vulnérabilités dans Microsoft Windows	Avis	2024-02-14	CV-E-2024-49126	9.0	Critical	CWE-77 (Command Injection)	0.92	https://www.cert.ssi.gouv.fr/...	A command injection vulnerability allows execution.	Ivanti	IPS	9.1R18, 22.6R1

- Attention une alerte ou un avis peuvent évoquer plusieurs CVE (parfois des centaines)

Une Alerta ou un avis ANSSI peut se trouver répété sur plusieurs lignes selon le nombre des CVE.

Étape 5 : Interprétation et Visualisation

L'**interprétation et la visualisation** des données constituent une étape cruciale pour exploiter les informations extraites et enrichies dans le cadre de ce projet. Avant de générer des alertes ou d'envoyer des notifications, il est essentiel de **comprendre** et **d'analyser** les vulnérabilités recensées afin de les prioriser de manière pertinente. La visualisation permet d'identifier rapidement les vulnérabilités critiques, les produits les plus impactés, ou les menaces avec une forte probabilité d'exploitation.

Grâce à des outils tels que **Pandas** pour l'analyse des données et **Matplotlib / Plotly /...** pour la visualisation, plusieurs représentations graphiques peuvent être construites, comme par exemple : (liste non exhaustive, à vous de faire preuve d'imagination)

- **Histogramme des scores CVSS** : pour observer la distribution des vulnérabilités selon leur niveau de gravité (critique, élevée, moyenne).
- **Diagramme circulaire des types de vulnérabilités (CWE)** : pour identifier les catégories de faiblesses les plus fréquentes (par exemple : Injection SQL, Débordement de mémoire, Authentification contournée).
- **Courbe des scores EPSS** : pour comprendre la probabilité d'exploitation des vulnérabilités recensées et les prioriser en fonction de leur risque réel.

- **Classement des produits ou éditeurs les plus affectés** : pour mettre en lumière les systèmes les plus vulnérables et cibler les actions à entreprendre.
- Heatmap des corrélations entre CVSS et EPSS : Analyser la relation entre le **score CVSS** (niveau de gravité) et le **score EPSS** (probabilité d'exploitation).
- **NUAGE DE POINTS ENTRE SCORE CVSS ET SCORE EPSS** : Visualiser comment la probabilité d'exploitation (EPSS) évolue par rapport au niveau de gravité (CVSS).
- **Courbe cumulative des vulnérabilités en fonction du temps** : Montrer l'évolution temporelle du nombre de vulnérabilités détectées.
- **Boxplot des scores CVSS par éditeur** : Montrer la **dispersion des scores CVSS** pour les éditeurs les plus affectés.
- Des visualisations particulières pour un type défini CWE
- Montrer l'évolution temporelle du nombre de vulnérabilités détectées.
- Visualiser les versions les plus fréquemment touchées des produits concernés.
- Analyser le nombre de vulnérabilités par **éditeur** et distinguer les **types de bulletins** (avis ou alertes).
- Montrer la **dispersion des scores CVSS** pour les éditeurs les plus affectés.
- ...

Cette phase d'analyse visuelle permet de prendre des **décisions éclairées** sur les vulnérabilités nécessitant une attention immédiate. Les résultats de cette interprétation serviront ensuite de **base solide** pour définir les critères d'alertes personnalisées et optimiser l'envoi des notifications. En reliant les indicateurs comme le **CVSS**, le **CWE**, et le **EPSS**, les étudiants pourront identifier les **menaces prioritaires** et comprendre le **contexte** des vulnérabilités dans un environnement réel.

En somme, l'étape **d'interprétation et visualisation** ne se limite pas à produire des graphiques. Elle s'inscrit comme un processus analytique qui transforme des données brutes en **informations exploitables** pour des actions concrètes telles que la **génération d'alertes ciblées** et la **réduction proactive des risques**.

Etape 6 : Modèles Machine learning

Vous devriez intégrer une phase de Machine Learning avec au moins un modèle supervisé et un modèle non supervisé, appliqués aux données enrichies (CVE, EPSS, alertes ANSSI...). avec la validation de ces modèles.

Exemples possibles (non exhaustifs) :

- Non supervisé : regroupement thématique de vulnérabilités, clustering, réduction de dimension, topic modeling...
- Supervisé : prédiction de la criticité, de l'EPSS, ou classification liée aux produits ou familles de vulnérabilités.

Les choix techniques (modèles, cibles, variables explicatives) sont libres et doivent être justifiés dans le notebook final, ainsi que la validations des modèles.

Il faut donc compléter le notebook de l'étape 5 avec la partie ML et des nouvelles visualisations liées à cet aspect.

Étape 7 : Génération d'Alertes et Notifications Email

Des alertes personnalisées sont générées lorsque des vulnérabilités critiques affectent des logiciels spécifiques. Des notifications email sont envoyées aux abonnés.

Exemple de code pour l'envoi d'email :

```
import smtplib  
from email.mime.text import MIMEText  
  
def send_email(to_email, subject, body):  
    from_email = votre\_email@gmail.com #n'utiliser pas votre compte, créez un nouveau  
    password = "mot_de_passe_application"  
  
    msg = MIMEText(body)  
    msg['From'] = from_email  
    msg['To'] = to_email  
    msg['Subject'] = subject  
  
    server = smtplib.SMTP('smtp.gmail.com', 587)  
    server.starttls()  
    server.login(from_email, password)  
    server.sendmail(from_email, to_email, msg.as_string())  
    server.quit()  
  
send_email("destinataire@email.com", "Alerte CVE critique", "Mettez à jour votre serveur Apache  
immédiatement.")
```

3. Les équipes

Ce travail est à réaliser en binômes.

4. Les livrables

Chaque livrable correspond à une étape clé du projet. voici une liste **organisée** et **détaillée** des **livrables** à produire.

1. Code Python fonctionnel, avec un readme clair et précis
 - a. Extraction des données ANSSI
 - b. Enrichissement des données via API
 - c. Consolidation des données (DataFrame Pandas)
 - d. Génération d'alertes et notifications
2. Fichier de données consolidées : un fichier **CSV** contenant les données consolidées et enrichies issues des bulletins ANSSI et des API externes.
3. Un Jupyter Notebook (.ipynb que vous compressez en zip, zip exclusivement et non pas 7z ou rar) et **son exportation au format html**
 - a. Le chargement du fichier csv produit
 - b. une exploration de ce DataFrame
 - c. une grande variété de visualisations et d'analyses pertinentes
 - d. Modèles Machine learning (un supervisé et un non supervisé)
 - e. validation des modèles proposés

Vous transmettez à votre prof, le fichier contributions.txt contenant le nom de chaque membre du binôme avec son ratio de participation. Exemple :

Nom1 Prenom1 75%

Nom2 Prenom2 25%

5. Notation

La **note de votre prof** une fois le travail fourni prendra en compte :

- Qualité et organisation du code Python, les fonctionnalités implémentées
- Rigueur dans l'extraction et l'enrichissement des données, gestions des exceptions...
- Visualisation (Qualité et quantité) et Analyse pertinente des résultats obtenus.
- Pertinence des choix et des modèles ML proposés, l'implémentation et la validation de ces modèles.
- Capacité à générer et à personnaliser des alertes.
- Respect des livrables et des deadlines

Ce peerreview se fera en deux parties: La présentation et les livrables

Pendant et jusqu'à la fin de la présentation tous les autres étudiants vont évaluer la présentation selon plusieurs critères (Introduction du sujet, Préparation des données, Analyse exploratoire des données, ML Apprentissage non supervisé, ML Apprentissage supervisé, Clarté de la présentation, Répartition du travail entre les membres de l'équipe...)

Après la séance, chacun parmi vous devra évaluer le travail de trois équipes.

En gros, chacun parmi vous doit évaluer les travaux de trois équipes selon des critères prédéfinis. Je vous offre ainsi la possibilité de voir le travail d'autres équipes et en même temps avoir un retour autre que la note de votre prof (**20% de la note finale et 80% pour le prof**). Il y aura des pénalités pour non-participation ou participation fictive.

Je compte sur vous pour avoir un travail authentique, j'espère ne pas avoir le même code. Partager certaines idées pourquoi pas, mais pas le code. Ne tombez pas dans le piège du plagiat ou plagiat déguisé (détection plagiat).

Jouez le jeu !

6. Dates importantes

Publication du sujet	Vendredi 16 janvier 2026
Constitution des binomes	Vendredi 16 janvier 2026
Dépôt des livrables	Avant la dernière séance soit avant jeudi 22 janvier 2026
Début des Evaluation par les Pairs	Jeudi 22 janvier 2026 à 12h00
Fin des Evaluation par les Pairs	jeudi 29 janvier 2026 à 23h

7. Django, Pour aller loin ! (optionnel)

Pour aller loin, vous pouvez intégrer votre travail en utilisant le framework django

<https://www.djangoproject.com/>

8. **IMPORTANT** Gestion Responsable des Accès aux Ressources Externes

Dans le cadre de ce projet, une partie des données nécessaires provient de ressources externes, notamment les flux RSS de l'**ANSSI** et les **API publiques** comme celles de **CVE** et **EPSS**. Étant donné le nombre important d'étudiants impliqués, il est crucial de garantir un usage responsable de ces ressources afin d'éviter une surcharge des serveurs ou une interprétation erronée du trafic généré.

Pour répondre à cette problématique, les mesures suivantes seront mises en place :

1. Utilisation de délais entre les requêtes (Rate Limiting) :

Vous devriez inclure un délai dans votre code (par exemple, 2 secondes entre chaque requête) pour limiter la fréquence des accès aux sites externes. Cela

permettra de répartir la charge de manière équitable et de minimiser l'impact sur les serveurs.

2. Fourniture de données pré-téléchargées (j'y travaille)

Afin de réduire au maximum les interactions directes avec les sites externes, des copies statiques des flux RSS et des réponses JSON des API seront mises à disposition. Vous travaillerez localement sur ces fichiers, ce qui garantit une expérience cohérente tout en protégeant les ressources externes. Il faut que votre code fonctionne avec l'accès aux url, mais l'utilisation des fichiers locaux permet d'aller vite sur la création du premier grand dataframe et sans risque de se faire bannir.

Exemples d'utilisation:

```
#%% récupération de la liste des toutes les alertes
import os
ids_alertes=[]
ids_alertes=os.listdir(r"alertes/")

#%% récuparation de la liste des tous les avis
import os
ids_avis=[]
ids_avis=os.listdir(r"avis/")

import json
#acceder à un avis
avis_id="CERTFR-2024-AVI-0012"
with open(r"avis/"+avis_id, 'r') as f:
    data=json.load(f)

#acceder à une alerte
alerte_id="CERTFR-2024-ALE-011"
with open(r"alertes/"+alerte_id, 'r') as f:
    data=json.load(f)

#acceder aux info mitre d'un cve
cve_id = "CVE-2023-46805"
with open(r"mitre/"+cve_id, 'r') as f:
    data=json.load(f)

#acceder aux info first d'un cve (epss)
cve_id = "CVE-2023-46805"
with open(r"first/"+cve_id, 'r') as f:
    data=json.load(f)
```