CyberWatch

Pipeline de veille cyber automatisé

TP n8n — Master SI / Cybersécurité

Architecture du pipeline

```
NVD API → n8n → Supabase → NocoDB → Metabase → Agent IA
                                  → Discord
```

Objectifs de la journée

- Collecter des CVE réelles depuis l'API NVD (National Vulnerability Database)
- Stocker et structurer les données dans Supabase via n8n
- Visualiser les données en temps réel avec NocoDB
- Enrichir les CVE avec un LLM (Claude / GPT) avant stockage
- Construire un agent IA capable d'interroger la base et d'agir sur Discord

Stack technique

Outil	Rôle
n8n	Orchestrateur — automatise tous les workflows
NVD API	Source de données CVE officielles (NIST)
Tavily	Recherche web enrichie pour contexte cyber
Supabase	Base de données PostgreSQL — stockage des CVE
NocoDB	Interface no-code pour visualiser Supabase en live
Metabase	Dashboards BI et analyse des données CVE
Claude / GPT	LLM pour enrichir et analyser les CVE
Discord	Canal de notification et d'interaction avec l'agent

📡 Étape 1 — Collecte NVD API

Récupérer les CVE critiques et les injecter dans Supabase

1.1 Créer un compte NVD

L'API NVD est gratuite. Sans clé : 5 requêtes / 30 secondes. Avec clé : 50 requêtes / 30 secondes.

1. Aller sur https://nvd.nist.gov/developers/request-an-api-key
2. Créer un compte et récupérer la clé API
3. Dans n8n, créer un credential HTTP Header Auth : apiKey = TA_CLE

(optionel)

1.2 Configurer le nœud HTTP Request dans n8n

Paramètre	Valeur
Method	GET
URL	https://services.nvd.nist.gov/rest/json/cves/2.0
pubStartDate	2025-01-01T00:00:00.000
pubEndDate	2025-04-16T23:59:59.999
cvssV3Severity	CRITICAL
resultsPerPage	100

URL à complèter pour tester dans le navigateur

`https://services.nvd.nist.gov/rest/json/cves/2.0?resultsPerPage=10`

1.3 Structure de la réponse NVD

La réponse contient un tableau vulnerabilities, chaque entrée ayant cette structure :

```
{
  vulnerabilities: [
    {
      cve: {
        id: 'CVE-2025-XXXXX',
        published: '2025-01-15T...',
        descriptions: [{ lang: 'en', value: '...' }],
        metrics: {
          cvssMetricV31: [{ cvssData: { baseScore: 9.8, ... } }]
        },
        configurations: [...]
      }
    }
  ]
```

```
}
```

1.4 Nœud Code — Parser la réponse

Ajouter un nœud Code après le HTTP Request. Ce code extrait et structure chaque CVE :

```
const items = $input.all();
const parsed = [];

for (const item of items) {
  const vulnerabilities = item.json.vulnerabilities || [];

  for (const vuln of vulnerabilities) {
    const cve = vuln.cve;
    if (!cve) continue;

    // Description en anglais
    const desc = (cve.descriptions || []).find(d => d.lang === 'en');

    // Score CVSS V3
    const cvssV3 = (cve.metrics?.cvssMetricV31 ||
                    cve.metrics?.cvssMetricV30 || [])[0]?.cvssData || {};

    // Vendor / Product depuis CPE
    let vendor = null, product = null;
    outer: for (const config of (cve.configurations || [])) {
      for (const node of (config.nodes || [])) {
        for (const cpe of (node.cpeMatch || [])) {
          const parts = (cpe.criteria || '').split(':');
          if (parts.length >= 5) {
            vendor  = parts[3] !== '*' ? parts[3] : null;
            product = parts[4] !== '*' ? parts[4] : null;
            break outer;
          }
        }
      }
    }

    parsed.push({ json: {
      cve_id:            cve.id,
      published_at:      cve.published,
      last_modified:     cve.lastModified,
      status:            cve.vulnStatus,
      cvss_v3_score:     cvssV3.baseScore ?? null,
      cvss_v3_severity:  cvssV3.baseSeverity ?? null,
      attack_vector:     cvssV3.attackVector ?? null,
      attack_complexity: cvssV3.attackComplexity ?? null,
      privileges_required: cvssV3.privilegesRequired ?? null,
      user_interaction:  cvssV3.userInteraction ?? null,
      confidentiality:   cvssV3.confidentialityImpact ?? null,
      integrity:         cvssV3.integrityImpact ?? null,
      availability:      cvssV3.availabilityImpact ?? null,
      affected_vendor:   vendor,
      affected_product:  product,
      description:       desc?.value ?? null,
      collected_at:      new Date().toISOString()
```

```
    }});
  }
}

return parsed;
```

1.5 Table Supabase

Créer la table cve dans Supabase (SQL Editor) :

```
create table cve (
  id uuid default gen_random_uuid() primary key,
  cve_id text unique,
  published_at timestamp,
  last_modified timestamp,
  status text,
  cvss_v3_score float,
  cvss_v3_severity text,
  cvss_v2_score float,
  attack_vector text,
  attack_complexity text,
  privileges_required text,
  user_interaction text,
  confidentiality text,
  integrity text,
  availability text,
  affected_vendor text,
  affected_product text,
  description text,
  collected_at timestamp default now()
);
```

1.6 Upsert dans Supabase

Pour éviter les doublons, utiliser un nœud HTTP Request (Supabase REST API) avec le header Prefer :

Paramètre	Valeur
Method	POST
URL	https://TON_PROJECT.supabase.co/rest/v1/cve
Header apikey	TON_ANON_KEY
Header Authorization	Bearer TON_ANON_KEY
Header Content-Type	application/json
Header Prefer	resolution=merge-duplicates
Body	{{ $json }}

> **✓✓ Validation**
> Lancer le workflow manuellement et vérifier dans Supabase que les lignes apparaissent.

Relancer une seconde fois : les CVE existantes doivent être mises à jour, pas dupliquées.

Étape 2 — NocoDB

Visualiser les données Supabase en temps réel

2.1 Lancer NocoDB

Lancer NocoDB en Docker sur votre laptop :

```
# Windows PowerShell
docker run -d --name nocodb -p 8080:8080 nocodb/nocodb:latest

# Accès : http://localhost:8080
```

2.2 Connexion à Supabase

NocoDB ne supporte pas IPv6 directement — utiliser le Session Pooler de Supabase.

4. Dans Supabase : Settings → Database → Connection → Session pooler
5. Copier l'URI de connexion (format postgresql://...)
6. Dans NocoDB : New Base → Connect to external database → PostgreSQL
7. Coller l'URI dans le champ Use Connection URL
8. Activer SSL → Save

> ⚠ **Erreur IPv4 fréquente**
> Si vous voyez 'Not IPv4 compatible' dans Supabase, vous devez utiliser le Session Pooler et non la Direct Connection.
> L'URI Session Pooler contient pooler.supabase.com dans l'host.

2.3 Ce que vous pouvez faire dans NocoDB

- Voir les CVE apparaître en temps réel pendant que n8n tourne
- Filtrer par severity, vendor, attack_vector sans écrire de SQL
- Créer des vues sauvegardées (ex : 'CVE critiques sans patch')
- Trier par cvss_v3_score décroissant pour voir les plus dangereuses

Étape 3 — Metabase

Visualiser et analyser les CVE avec des dashboards interactifs

3.1 Différence avec NocoDB

	NocoDB
Usage	Parcourir et éditer les données
Public cible	Opérationnel (saisie, correction)
Analogie	Excel collaboratif
Modification	Oui

Les deux outils se connectent à la même base Supabase — NocoDB pour explorer, Metabase pour analyser.

3.2 Lancer Metabase en Docker

Lancer Metabase sur votre laptop (port 3001 pour éviter le conflit avec l'app Express) :

```
# Windows PowerShell
docker run -d --name metabase -p 3001:3000 metabase/metabase:latest

# Accès après 1-2 minutes d'initialisation :
# http://localhost:3001
```

⏳ Patience

Metabase prend 1 à 2 minutes à démarrer la première fois.
Si la page ne répond pas immédiatement, attendre et rafraîchir.

3.3 Connexion à Supabase

Même logique que NocoDB : utiliser le Session Pooler pour la compatibilité IPv4.

9. Créer un compte admin sur http://localhost:3001
10. Add your data → PostgreSQL
11. Dans Supabase : Settings → Database → Connection → Session pooler → copier l'URI
12. Dans Metabase : coller l'URI ou remplir les champs manuellement
13. Activer SSL → Save

3.4 Créer les visualisations

Dans Metabase : New → Question → sélectionner la table cve

Visualisation 1 — Répartition par sévérité

Paramètre	Valeur
Summarize	Count of rows
Group by	cvss_v3_severity
Type de graphique	Pie chart (camembert)
Nom	CVE par sévérité

Visualisation 2 — Score CVSS moyen par vecteur d'attaque

Paramètre	Valeur
Summarize	Average of cvss_v3_score
Group by	attack_vector
Type de graphique	Bar chart (barres)
Nom	Score moyen par vecteur

Visualisation 3 — Timeline des CVE publiées

Paramètre	Valeur
Summarize	Count of rows
Group by	published_at (by week)
Type de graphique	Line chart (courbe)
Nom	CVE publiées par semaine

3.5 Assembler le dashboard

14. New → Dashboard → nommer : CyberWatch
15. Add a question → ajouter les 3 visualisations créées
16. Réorganiser les cartes par glisser-déposer
17. Save

3.6 Aller plus loin — questions SQL

Metabase permet aussi d'écrire du SQL directement : New → SQL Query

```
-- Top 10 vendors les plus touchés
SELECT affected_vendor, COUNT(*) as nb_cve,
       ROUND(AVG(cvss_v3_score)::numeric, 2) as score_moyen
FROM cve
WHERE affected_vendor IS NOT NULL
GROUP BY affected_vendor
```

```
ORDER BY nb_cve DESC
LIMIT 10;
```

```
-- CVE critiques exploitables sans interaction utilisateur
SELECT cve_id, affected_vendor, affected_product,
       cvss_v3_score, published_at
FROM cve
WHERE cvss_v3_severity = 'CRITICAL'
  AND user_interaction = 'NONE'
  AND privileges_required = 'NONE'
ORDER BY cvss_v3_score DESC;
```

Étape 4 — Agent IA CyberWatch

Un agent qui interroge la base, cherche sur le web et notifie Discord

5.1 Architecture de l'agent

L'agent reçoit des questions via un webhook et dispose de 5 outils pour y répondre :

Outil	Description
query_supabase	Interroger la base de CVE locale
collect_nvd	Déclencher une nouvelle collecte NVD
search_tavily	Rechercher des infos récentes sur internet
send_discord	Envoyer un rapport sur Discord

5.2 Chat trigger

Créer un nouveau workflow n8n avec un nœud Chat trigger.

5.3 System prompt de l'agent

Dans le nœud AI Agent, utiliser ce system prompt :

```
Tu es CyberWatch, un agent expert en cybersécurité.
Tu aides les analystes à surveiller les CVE et vulnérabilités.

Outils disponibles :
- query_supabase : interroger la base CVE locale
- collect_nvd : déclencher une nouvelle collecte NVD
- search_tavily : rechercher des infos récentes sur internet
- send_discord : envoyer un rapport sur Discord

Règles :
1. Commence toujours par query_supabase avant d'aller sur internet
2. Si la donnée est absente ou trop ancienne, utilise collect_nvd
   puis search_tavily pour enrichir
3. Termine par un résumé clair et structuré
4. Si un discord_webhook est fourni, envoie automatiquement le rapport
5. Pour les CVE critiques, mentionne toujours si un patch est disponible
```

5.4 Outil 1 — Query Supabase

Nœud HTTP Request configuré ainsi :

Paramètre	Valeur

Method	GET
URL	https://TON_PROJECT.supabase.co/rest/v1/cve
Header apikey	TON_ANON_KEY
Param select	*
Param cvss_v3_severity	eq.CRITICAL
Param order	published_at.desc
Param limit	20

Description pour l'agent : Interroge la base locale de CVE. Utilise cet outil en priorité pour répondre aux questions sur les vulnérabilités connues.

5.5 Outil 2 — Collecte NVD

Appel vers le webhook du workflow de collecte NVD déjà créé :

Paramètre	Valeur
Method	POST
URL	https://IP_DU_VPS /webhook/nvd-collect
Body severity	{{ $json.severity || 'CRITICAL' }}

5.6 Outil 3 — Recherche Tavily

Créer un compte Tavily (gratuit, 1000 req/mois) sur tavily.com, puis :

```
POST https://api.tavily.com/search
{
  "api_key": "{{ $credentials.tavily }}",
  "query": "{{ $json.query }}",
  "search_depth": "advanced",
  "max_results": 5
}
```

5.7 Outil 4 — Notification Discord

Webhook Discord avec message formaté :

```
POST {{ $json.discord_webhook }}
{
  "embeds": [{
    "title": "🛡️ CyberWatch Report",
    "description": "{{ $json.message }}",
    "color": 3447003,
    "timestamp": "{{ $now }}"
  }]
}
```

5.8 Questions de test pour l'agent

💬 Exemples de questions à envoyer via curl ou Postman

Question simple (Supabase uniquement) :

```
"Combien de CVE critiques ont été publiées ce mois-ci ?"
```

Question multi-outils :

```
"Quelles sont les dernières failles sur Cisco ? Envoie un rapport Discord."
```

Question avec collecte :

```
"Collecte les nouvelles CVE Apache et fais un résumé des risques."
```

✓ Récapitulatif

Ce que vous avez construit aujourd'hui

Étape	Ce que vous avez appris
1 — NVD API + n8n	Appels HTTP, parsing JSON complexe, upsert base de données
2 — NocoDB	Interface no-code sur PostgreSQL, visualisation temps réel
3 — Metabase	Dashboards BI, SQL analytique, différence exploration vs analyse
4 — Enrichissement IA	LLM comme brique de transformation dans un pipeline de données
5 — Agent IA	Tool calling, raisonnement multi-étapes, workflows non-déterministes
2 — NocoDB	Interface no-code sur PostgreSQL, visualisation temps réel
3 — Enrichissement IA	LLM comme brique de transformation dans un pipeline de données
4 — Agent IA	Tool calling, raisonnement multi-étapes, workflows non-déterministes

🚀 Pour aller plus loin

- Ajouter un trigger Cron pour une collecte automatique toutes les heures
- Brancher un canal Discord dédié par niveau de sévérité (CRITICAL → @here)
- Ajouter Tavily pour enrichir chaque CVE avec des articles de blog et PoC publics
- Créer une interface web simple (Webhook → formulaire HTML) pour interroger l'agent
- Ajouter la mémoire de conversation dans l'agent (nœud Window Buffer Memory)

📚 Ressources utiles

NVD API : https://nvd.nist.gov/developers/vulnerabilities
n8n docs : https://docs.n8n.io
Tavily API : https://docs.tavily.com
Supabase docs : https://supabase.com/docs
NocoDB docs : https://docs.nocodb.com