Document Technique – Scanner de Vulnérabilités Réseau en Go

❶ Introduction

Ce projet consiste à développer un scanner de sécurité réseau capable de :

✓ Scanner les ports ouverts sur une plage d'IP.

✓ Identifier les services en cours d'exécution (bannières).

✓ Vérifier les failles connues via une base CVE (via l'API NVD).

✓ Générer des rapports détaillés en JSON et HTML.

❷ Plan d'action

Étape	Description	Technologies
Analyse des besoins	Définition des fonctionnalités du scanner	Documentation
Mise en place du projet	Initialisation du module Go et structure des fichiers	go mod init
Scan des ports ouverts	Développement du scan TCP des ports sur une plage d'IP	net (Go)
Identification des services	Récupération des bannières des services	net.Conn
Vérification des vulnérabilités	Intégration de l'API NVD pour récupérer les CVEs	API REST
Export des résultats	Génération des rapports JSON et HTML	encoding/json, html/template
Tests et optimisation	Vérification du bon fonctionnement et amélioration des performances	Tests manuels & logs
Rédaction du rapport	Documentation et retour d'expérience	Markdown & PDF

❸ Technologies Mises en Place

Langage : Go (Langage rapide et performant pour le réseau)

- **Packages Go Utilisés:**

✓ net - Scanner les ports ouverts

✓ time - Gérer les délais de connexion

✓ encoding/json - Génération du fichier JSON

✓ html/template - Génération du fichier HTML

✓ net/http - Communication avec l'API CVE

✓ strconv - Conversion de types

- **API & Bases de Données :**

✓ NVD (National Vulnerability Database) via API REST pour récupérer les CVEs associées aux services détectés.

- **Outils de Développement :**

✓ VS Code - Éditeur de code

✓ GoLand - Alternative pour le développement Go

✓ Postman - Test des requêtes API

✓ Wireshark - Analyse des paquets réseau

✓ Linux (Debian) - Environnement de test

❹ Arborescence du Projet

```
network-scanner/
├── go.mod
├── go.sum
├── main.go
├── scanner/
│   ├── scan_ip.go
│   ├── port_scanner.go
│   ├── service_identifier.go
├── reports/
│   ├── scan_results_json.go
│   ├── scan_results_html.go
├── vulnerability/
│   ├── cve_database.go
├── network-scanner/results/
│   ├── scan_results.json
│   ├── scan_results.html
```

❺ Exemples d'Exécution

• Exécution dans le terminal :

```
$ go run main.go
Starting Network Scanner...
Scanning 192.168.1.1...
Scanning 192.168.1.2...
Scanning 192.168.1.3...
Scan complete. Results saved to results/scan_results.json and results/scan_results.html
```

• Résultat en JSON (`scan_results.json`) :

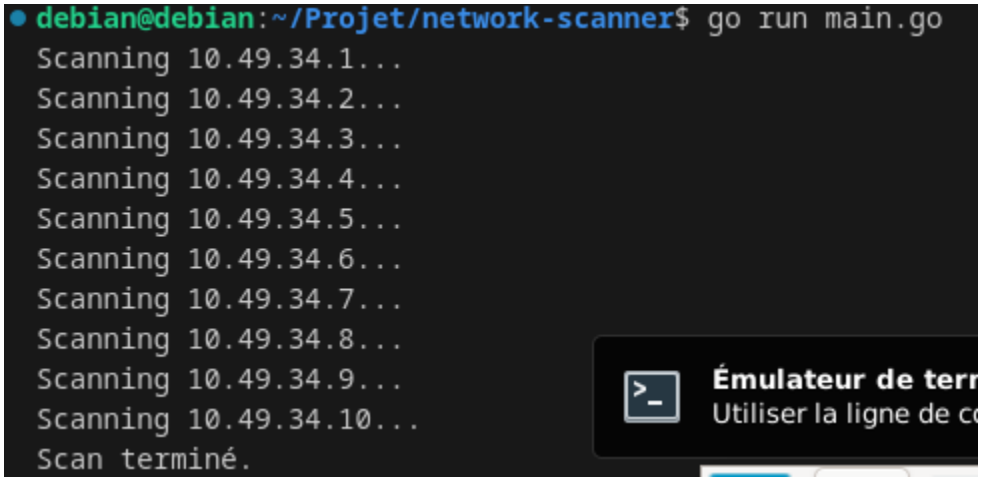
```
[
 {
  "ip": "192.168.1.1",
  "ports": [
   {
    "port": 22,
```

```
      "service": "SSH-2.0-OpenSSH_7.6p1 Debian-4",
      "cves": [
       {
         "id": "CVE-2018-15473",
         "description": "OpenSSH 7.6p1 est vulnérable à une attaque de validation d'identité à distance.",
         "cvss_score": 5.3
       }
      ]
    },
    {
      "port": 80,
      "service": "HTTP/1.1 200 OK\nServer: Apache/2.4.29 (Debian)",
      "cves": [
       {
         "id": "CVE-2017-7679",
         "description": "Apache HTTP Server 2.4.29 est vulnérable à une fuite de mémoire dans mod_mime.",
         "cvss_score": 7.5
       }
      ]
    }
   ]
 },
 {
   "ip": "192.168.1.2",
   "ports": [
```

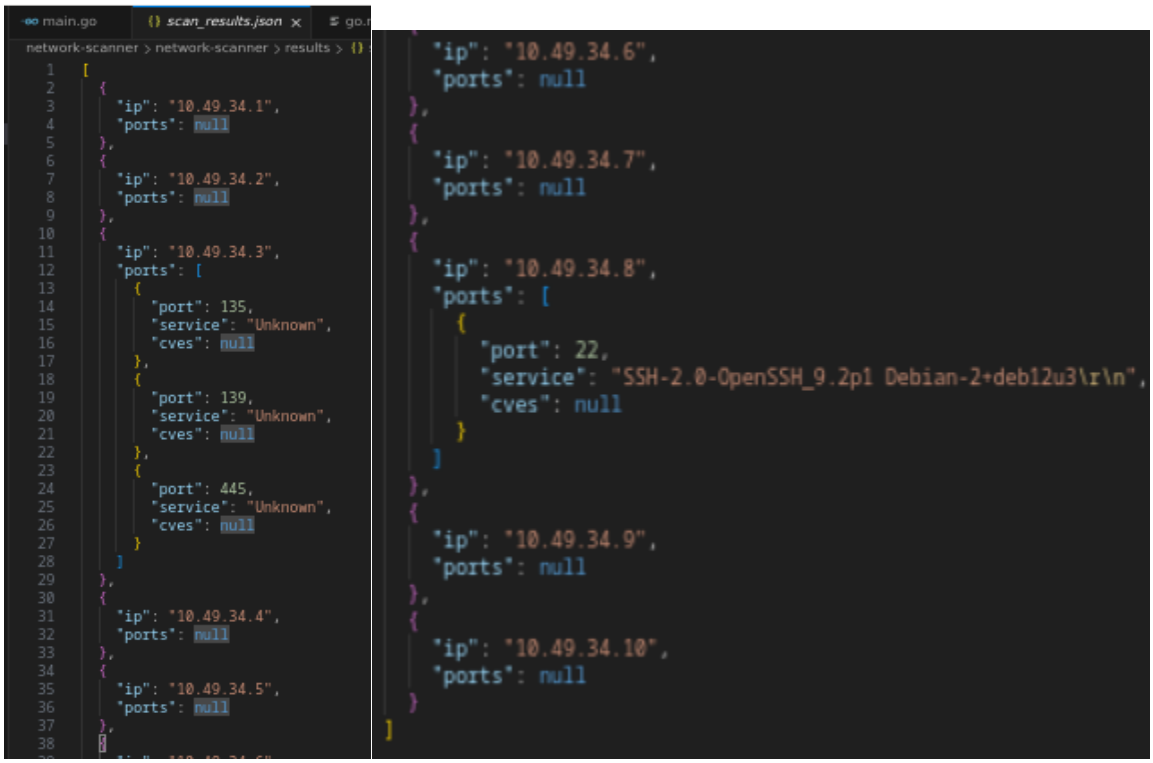
```
    {
      "port": 21,
      "service": "220 vsftpd 3.0.3",
      "cves": [
        {
          "id": "CVE-2011-2523",
          "description": "vsftpd 2.3.4 contient une backdoor permettant l'exécution de code à distance.",
          "cvss_score": 9.8
        }
      ]
    },
    {
      "port": 80,
      "service": "HTTP/1.1 301 Moved Permanently\nServer: nginx/1.14.2",
      "cves": []
    }
  ]
 },
 {
  "ip": "192.168.1.3",
  "ports": [
    {
      "port": 25,
      "service": "220 mail.example.com ESMTP Exim 4.94.2",
      "cves": [
```

```
      {
        "id": "CVE-2019-15846",
        "description": "Exim 4.92-4.94.2 est vulnérable à une exécution de code à distance en raison d'un défaut de gestion des TLS SNI.",
        "cvss_score": 9.1
      }
    ]
  }
 ]
}
```

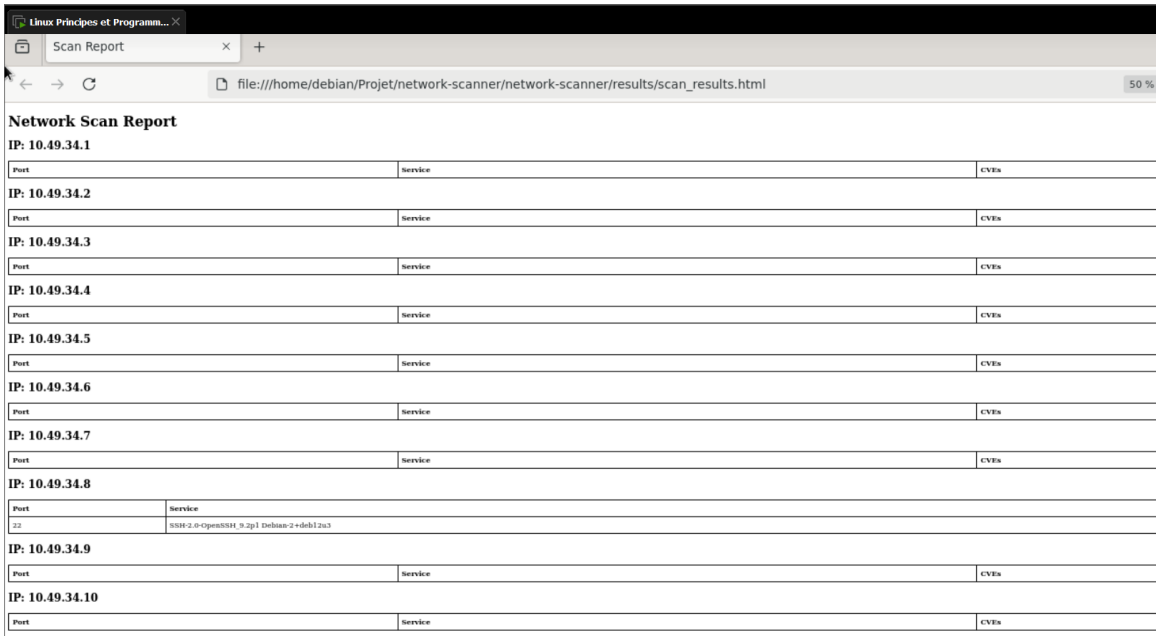

Exécution à l'école dans le terminal :



• **Résultat en JSON :**



• **Résultat en HTML :**



❻ Problèmes Rencontrés & Solutions

Problème	Solution
Go ne trouve pas les modules (`package not in std`)	Vérifier “go.mod”, exécuter “go mod tidy”, vérifier les chemins d'import
Ports non détectés sur certaines IP	Ajuster le “timeout” à “500ms”
Difficulté à récupérer les CVEs	Utilisation de l’API NVD avec “net/http” et parsing JSON
Export HTML mal formaté	Correction avec “html/template” pour éviter l'injection XSS
Performances lentes sur de grandes plages d'IP	Optimisation avec “goroutines” pour exécuter les scans en parallèle

❼ Apports du Projet pour mes Compétences

✓ Programmation réseau en Go

✓ Utilisation des goroutines pour le parallélisme

✓ Manipulation de JSON et HTML avec Go

✓ Intégration d’une API externe (NVD)

✓ Génération de rapports JSON & HTML

8 Conclusion & Perspectives d’Amélioration

✓ Ce projet a permis de développer un outil efficace de scan réseau en Go.

✓ Il offre une bonne base pour une future extension avec plus de fonctionnalités (ex : détection d’OS, support UDP, interface graphique).

✓ Pour l’avenir, une optimisation via `goroutines` et une meilleure gestion des logs seraient des pistes intéressantes.