Mini dossier M1 Cyber

Infrastructure Docker Sécurisée

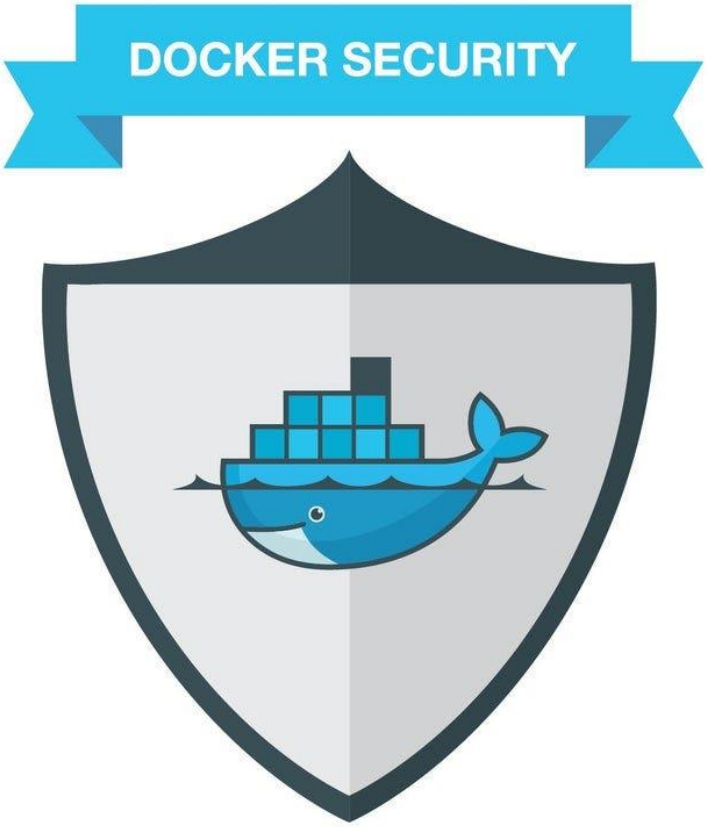


TABLE DES MATIERES

1.Technologies

1.0 Docker :

1.0.1 C'est quoi ?
Docker est une plateforme logicielle qui simplifie le processus de création, d'exécution, de gestion et de distribution des applications. Pour ce faire, elle virtualise le système d'exploitation de l'ordinateur sur lequel elle est installé.

1.0.1 Exemple :

Le problème :
Supposons que nous ayons trois applications différentes basées sur Python que nous prévoyons d'héberger sur un seul serveur (qui peut être soit une machine physique, soit une machine virtuelle).

Chacune de ces applications utilise une version différente de Python, ainsi que les bibliothèques et dépendances associées, qui diffèrent d'une application à l'autre.

Comme nous ne pouvons pas installer différentes versions de Python sur la même machine, cela nous empêche d'héberger les trois applications sur le même ordinateur.

Solution :
Dans un tel scénario, nous pourrions résoudre ce problème soit en ayant trois machines physiques, soit en ayant une seule machine physique, qui est suffisamment puissante pour héberger et faire fonctionner trois machines virtuelles sur elle.

Les deux options nous permettraient d'installer différentes versions de Python sur chacune de ces machines, ainsi que leurs dépendances associées.

Quelle que soit la solution choisie, les coûts liés à l'acquisition et à la maintenance du matériel sont assez élevés

Or, un conteneur délivre uniquement les ressources nécessaires à une application.

Cette méthode permet de réduire l’empreinte des applications sur l’infrastructure. Le conteneur regroupe tous les composants systèmes nécessaires à l’exécution du code, sans pour autant peser aussi lourd qu'un OS complet.

1.1 Nextcloud :

1.1.1 Fonctionnalités

Nextcloud est une plateforme de collaboration, permettant la gestion des contacts, le stockage de fichier, la synchronisation entre différents terminaux (ordinateurs, smartphone, tablettes) ainsi que l'intégration d'une messagerie sur le web.

1.2 Grype:

1.2.1 Fonctionnement

Grype est un scanner de vulnérabilités pour les images de conteneurs et les systèmes de fichiers. Il peut également trouver des vulnérabilités dans les paquets de langages spécifiques.

- Ruby (Gems)
- Java (JAR, WAR, EAR, JPI, HPI)
- JavaScript (NPM, Yarn)
- Python (Egg, Wheel, Poetry, requirements.txt/setup.py files)
- Dotnet (deps.json)
- Golang (go.mod)
- PHP (Composer)
- Rust (Cargo)

1.3 MariaDB :

1.3.1 Présentation & Fonctionnement

MariaDB est l'une des bases de données relationnelles open source les plus populaires. Conçu par les premiers développeurs de MYSQL. Elle fait partie de la plupart des offres de cloud computing et est proposée par défaut dans la plupart des distributions Linux.

Elle repose sur les valeurs de performance, de stabilité et d'ouverture, et la fondation MariaDB garantit que les contributions seront acceptées sur la base de leur valeur technique. Parmi les nouvelles fonctionnalités récentes, citons le clustering avancé avec Galera Cluster 4, les caractéristiques de compatibilité avec Oracle Database et les tables de données temporelles, qui permettent d'interroger les données telles qu'elles se présentent à n'importe quel moment dans le passé.

1.4 Nginx:

1.4.1 Serveur WEB

En tant que serveur web, NGINX excelle par sa performance et son efficacité. Contrairement à d'autres serveurs web qui utilisent un thread ou un processus pour chaque connexion, NGINX utilise une architecture orientée événements, ce qui lui permet de gérer un grand nombre de connexions simultanées avec une utilisation minimale des ressources système. Cela se traduit par une capacité à servir plus de clients avec moins de ressources, ce qui peut être un avantage significatif pour les sites à fort trafic.

1.4.2 Reverse Proxy

Un Reverse Proxy est un serveur qui se situe entre les clients et un ou plusieurs serveurs web, transmettant les requêtes des clients aux serveurs appropriés. NGINX peut être configuré comme un Reverse Proxy, permettant une répartition efficace du trafic, une mise en cache du contenu et une protection contre les attaques DDoS.

1.5 Fail2ban :

1.5.1 Analyse de logs & Filtre

Fail2ban est une application qui examine les logs de divers services (SSH, Apache, FTP…) en cherchant des correspondances entre des motifs définis dans ses filtres et dans les entrées de logs. Les filtres sont également appelés prisons. On y définit différentes conditions qui seront utilisées pour les analyses.

Lorsqu'une correspondance est trouvée une ou plusieurs actions sont exécutées.

Typiquement, fail2ban cherche des tentatives répétées de connexions infructueuses dans les fichiers journaux et procède à un bannissement en ajoutant une règle au pare-feu iptables ou nftables pour bannir l'adresse IP de la source.

1.6 IPTables :

1.6.1 Définition

Iptables est un logiciel libre de l'espace utilisateur Linux grâce auquel l'administrateur système configure les chaînes et règles dans le pare-feu en espace noyau. Il faut donc être administrateur (Root)

Différents programmes sont utilisés selon le protocole employé : iptables est utilisé pour le protocole IPv4, Ip6tables pour IPv6, Arptables pour ARP (Address Resolution Protocol) ou encore Ebtables, spécifique aux trames Ethernet.

1.6.2 Scripting

Nous avons mis en place un script bash qui est exécuté en fond et qui banni une liste d'IP flaggé comme abusive. Il existe plusieurs projet Github qui partage de nombreuses adresses IP blacklistées.

1.7 Lynis:

1.7.1 Fonctionnement

Lynis est un outil de sécurité conçu pour les systèmes fonctionnant sous Linux, macOS ou Unix. Il effectue une analyse approfondie de l'état de santé des systèmes afin de soutenir le durcissement système et les tests de conformité.

Comme Lynis est flexible, il est utilisé à différentes fins. Les cas d'utilisation typiques de Lynis sont les suivants

- Audit de sécurité
- Tests de conformité (par exemple PCI, HIPAA, SOx)
- Tests de pénétration
- Détection des vulnérabilités
- Durcissement du système

1.8 Python :

1.8.1 Langage & Bibliothèque

Python est un langage polyvalent et puissant qui permet de travailler rapidement et d'intégrer des systèmes plus efficacement.

Nous avons notamment utilisé les librairies cryptographie, fernet, os, requests.

1.9 Gitlab :

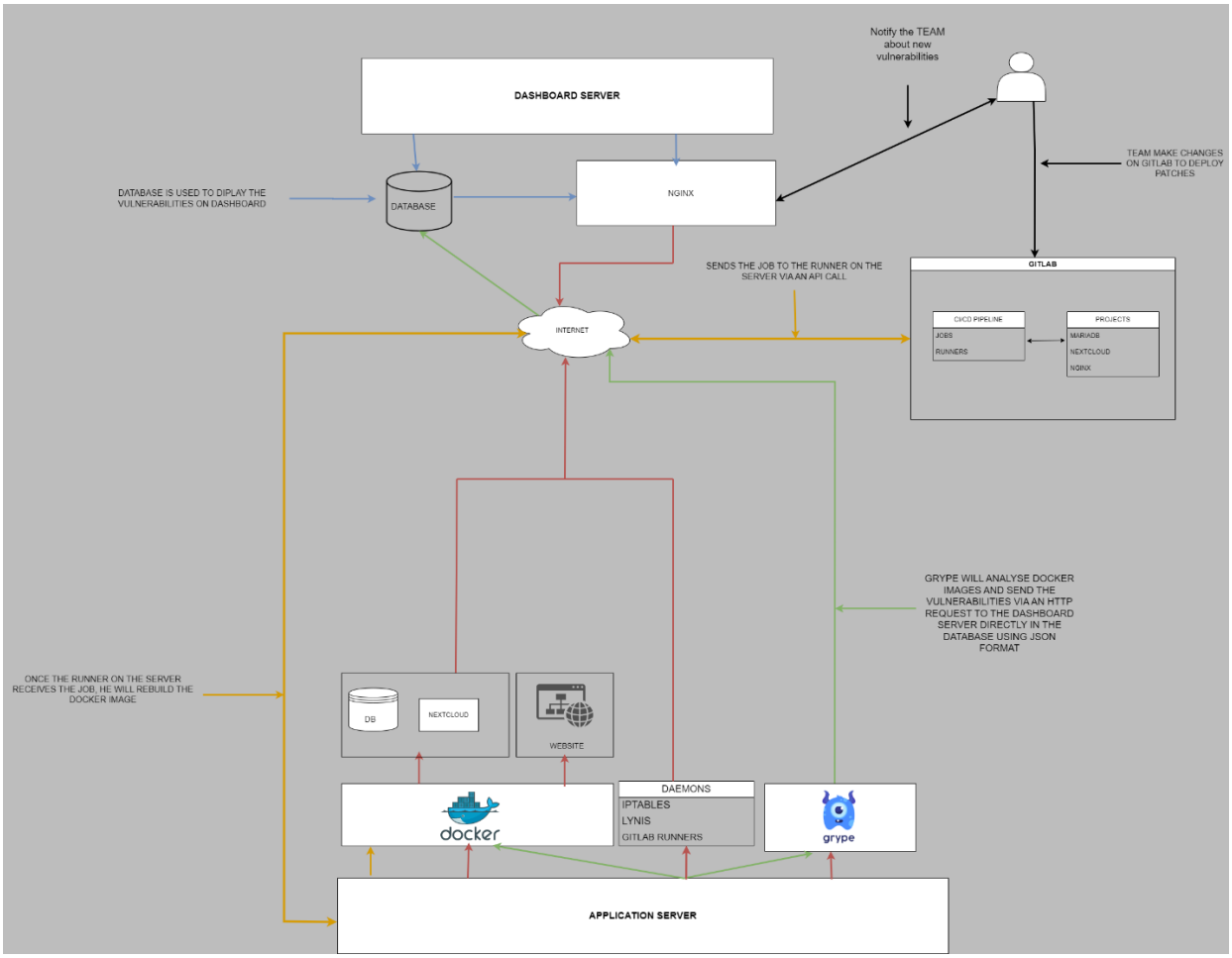
1.9.1 Versioning

GitLab est une plateforme DevOps basée sur le web qui fournit un ensemble complet d'outils pour la gestion et la réalisation de projets logiciels. Il s'agit d'un gestionnaire de référentiel de contrôle de version qui permet aux équipes de collaborer sur le code, de suivre les modifications et de gérer les processus de développement de logiciels.

GitLab prend en charge Git, un système de contrôle de version distribué, et offre des fonctionnalités telles que l'hébergement de code, le suivi des problèmes, l'intégration et le déploiement continus, l'examen du code et des outils de collaboration. Il peut être hébergé sur site ou utilisé en tant que service basé sur le cloud. GitLab est largement utilisé par les équipes de développement de logiciels pour une gestion de projet efficace et rationalisée.

2) ARCHITECTURE

2.1 Schéma



2.1.1 Flux & Activités

	Description	Source	Destination	Protocol
1	Analyse des images docker	GRYPE	IMAGES DOCKER	N/A
2	Envoie des vulnérabilités	GRYPE	Base de données sur le serveur Dashboard	HTTP
3	Mise en place des données	Base de données	Serveur WEB Nginx	TCP/IP, UNIX Socket,
4	Dashboard	NGINX	Internet	HTTP, TCP/IP
5	Equipe vers dashboard	Utilisateurs	Serveur WEB NGINX	HTTP, TCP/IP
6	Modification du code source d'un projet	Poste utilisateur	GITLAB	HTTP, TCP/IP
7	Runner Gitlabs	GITLAB	Serveur APPLICATION	HTTP, TCP/IP

2.1.2 Serveurs

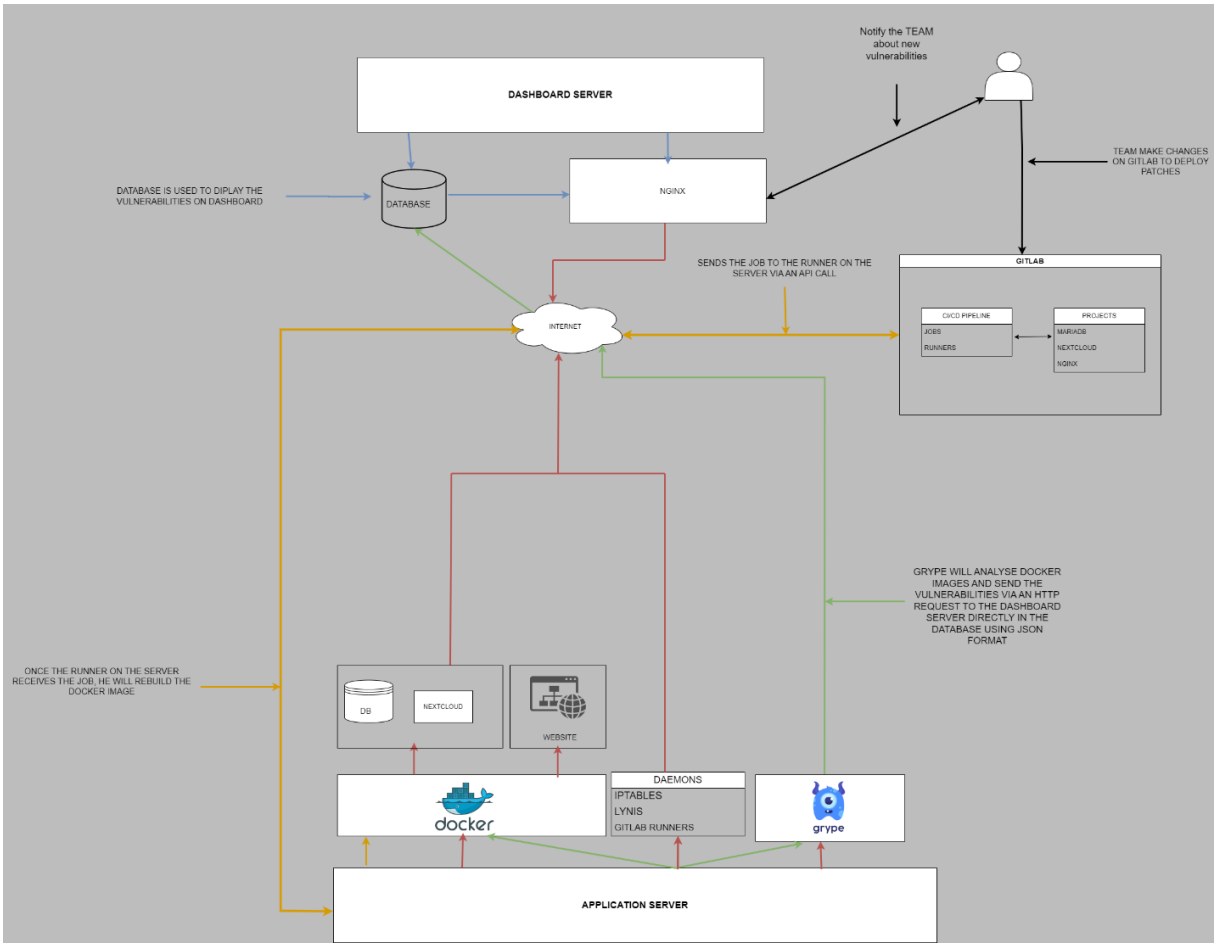
Serveur d'applications :

Il s'agit d'un serveur linux, sur lequel nous avons installé Docker, Lynis, Grype, des runners gitlabs, fail2ban etc...

Ce serveur à pour but d'avoir en production différentes applications, notamment nextcloud, une plateforme de collaboration**.** Egalement, l'hébergement de site web de tiers etc. Disponible sur Internet.

Chaque service ou conteneur à son réseau dédié, de ce fait, le réseau Nextcloud n'est pas intégré à celui du site tiers.

Via un crontab, grype viens scanner automatiquement les vulnérabilités sur les images Docker en production, et renvoie via une requête HTTP les données au format JSON directement dans la base de données du serveur Dashboard (celle-ci est en backend).

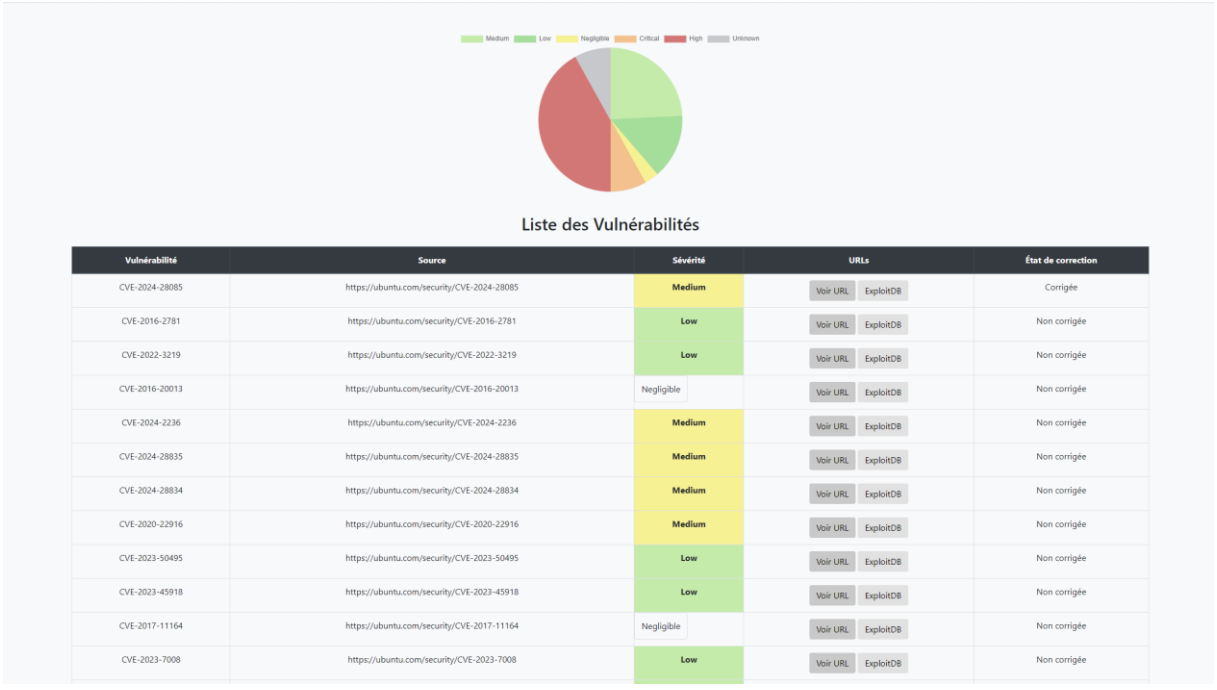


Le format JSON a été choisis pour les raisons suivantes :

1. Sa syntaxe simple basée sur des paires clé-valeur.
2. Sa compatibilité avec de nombreux langages de programmation.
3. Sa facilité d'intégration dans les applications.
4. Sa structure de données permettant d'imbriquer des objets et des tableaux.
5. Sa légèreté, qui permet d'économiser de la bande passante lors de l'échange de données via HTTP.
6. Son support pour différents types de données tels que les chaînes de caractères, les nombres, les booléens, les tableaux et les objets.

Serveur DASHBOARD :

Ce serveur nous sert à examiner les logs remontés via les étapes précédentes et à prendre des décisions rapides sur la mise en place des correctifs ou mises à jour des images. Les données remontées sont interprétées via un Dashboard créé à l'aide de Javascript.

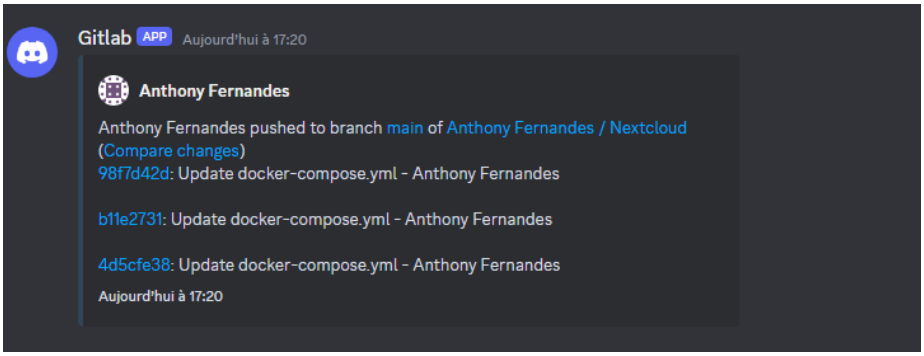


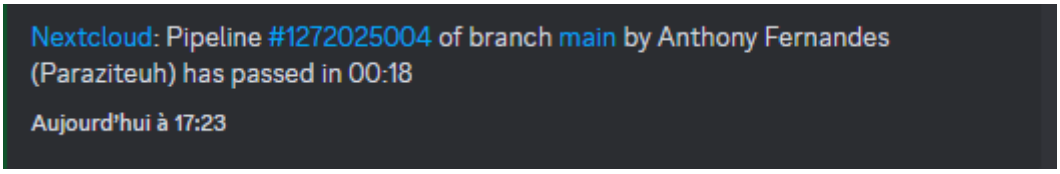
Serveur Discord & Gitlab

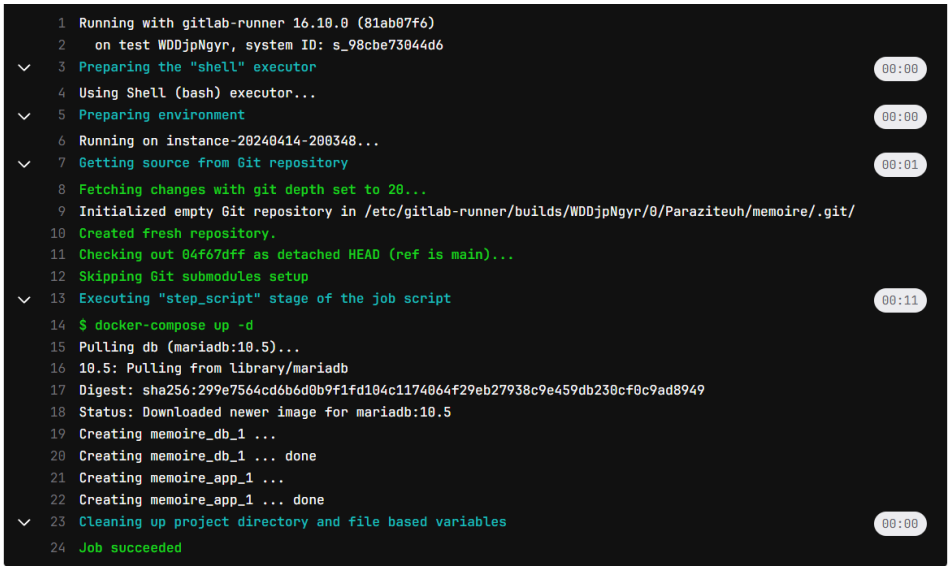
Des notifications sont remontées sur un serveur Discord via un bot, permettant de notifier rapidement les équipes. Par exemple si le travail exécuté par un pipeline d'intégration échoue, une notification est remontée sur discord permettant aux équipes d'avoir l'information quasi-instantanément.

En cas d'échec, le membre de l'équipe, va se connecter à Gitlab, modifier, corriger, le code et les fichiers. La modification enregistrer avec un "Git commit" va déclencher un ordre de travail. Le Git Runner installé sur le serveur d'applications va reconstruire l'image avec les mises à jour.

Exemple : Si MariaDB est en version 10.6 ; que l'équipe modifie sur gitlab la version en 10.5 un ordre de travail (job) va reconstruire l'image sur le serveur d'application.







On constate que le "Runner" récupère l'image à la version 10.5 et la déploie. On reçoit ensuite la notification de notre Bot.

3) Durcissement

3.1 Lynis

Sur les deux serveurs Applications et Dashboard, nous avons installé Lynis, un outil permettant de tester le durcissement du système d'exploitation.

L'outil va tester les droits sur les fichiers, vérifier les ports ouverts, comparer les binaires d'origine etc. Il donne des recommandations afin d'avoir un système avec le moins de vulnérabilités ainsi qu'un score de durcissement 0 étant le moins bon et 100 le meilleur.

```
  Lynis security scan details:

  Hardening index : 80 [################    ]
  Tests performed : 266
  Plugins enabled : 2

  Components:
  - Firewall               [V]
  - Malware scanner        [V]

  Scan mode:
  Normal [ ]  Forensics [ ]  Integration [ ]  Pentest [V] (running privileged)

  Lynis modules:
  - Compliance status      [?]
  - Security audit         [V]
  - Vulnerability scan     [V]

  Files:
  - Test and debug information      : /var/log/lynis.log
  - Report data                     : /var/log/lynis-report.dat
```

3.2 FAIL2BAN

Nos deux serveurs étant distants, nous nous connectons via SSH. Pour renforcer la sécurité de notre infrastructure, nous avons mis en place des prisons. En cas d'échec de connexion au service, les adresses IP sont automatiquement bannies.

```
Status for the jail: sshd
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     1
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 26
   |- Total banned:     26
   `- Banned IP list:   103.105.78.231 103.40.253.135 106.53.172.155 119.27.
183.81.169.238 186.4.174.165 190.144.57.186 192.241.141.43 192.99.247.77 194
```

4) Exploitation

Nous avons développé un outil de chiffrement type Ransomware nommés XSCAN avec Python.

4.1 Objectif :

Le module de chiffrement et déchiffrement de XSCAN a été conçu dans le cadre d'une expérience de sensibilisation à la sécurité informatique. Son objectif est de démontrer les risques associés au téléchargement de logiciels à partir de sources non vérifiées, ainsi que l'importance d'une infrastructure sécurisée pour protéger les données sensibles contre de telles menaces.

4.2 Utilisation Prévue :

Ce module est présenté comme un outil de sécurité pour Docker, prétendument destiné à scanner et sécuriser les conteneurs Docker. Cependant, en réalité, il s'agit d'un faux ransomware conçu à des fins d'expérimentation et de sensibilisation. Les utilisateurs seront trompés en pensant qu'ils téléchargent un scanner Docker légitime, mais lors de l'exécution du module, leurs fichiers seront chiffrés.

4.3 Sensibilisation à la Sécurité :

Ce module vise à sensibiliser les utilisateurs aux risques associés au téléchargement de logiciels à partir de sources non vérifiées. En simulant la présence d'un scanner Docker légitime qui s'avère être un faux ransomware, nous illustrons les dangers auxquels les utilisateurs peuvent être confrontés. De plus, en démontrant la facilité avec laquelle le chiffrement peut être contourné en reconstruisant le conteneur Docker sécurisé, nous mettons en évidence l'importance d'une infrastructure sécurisée pour protéger les données sensibles.

4.4 Comment ça fonctionne ?

Le script de chiffrement commence par générer automatiquement une clé de chiffrement symétrique à l'aide de la bibliothèque cryptographie. Cette clé est ensuite envoyée à une URL spécifiée via RequestBin pour simuler la communication avec un serveur distant.

Ensuite, le script parcourt le répertoire de travail et chiffre tous les fichiers présents à l'aide de la clé générée, en utilisant l'algorithme Fernet pour garantir la sécurité des données. Une fois le chiffrement terminé, les utilisateurs sont informés que leurs fichiers ont été sécurisés.

D'un autre côté, le script de déchiffrement commence par récupérer la clé de chiffrement à partir d'un fichier local où elle a été stockée. Ensuite, les utilisateurs sont invités à saisir une phrase secrète qui leur est fournie via RequestBin. Une fois la phrase secrète saisie, le script déchiffre tous les fichiers chiffrés à l'aide de la clé récupérée, permettant ainsi aux utilisateurs de restaurer leurs fichiers à l'état d'origine.

4.5 Choix Techniques et Bibliothèques utilisées

Python :

Python a été choisi comme langage de programmation principal pour ce module en raison de sa simplicité, de sa polyvalence et de sa popularité dans le domaine de la sécurité informatique.
De plus, nous connaissons bien ce langage.

cryptographie.fernet :

La bibliothèque cryptographie, et en particulier le module Fernet, est utilisée pour la génération de clés de chiffrement et le chiffrement des données.

OS :

Le module os est utilisé pour interagir avec le système d'exploitation sous-jacent.

Requests :

Le module requests permet d'envoyer des requêtes HTTP. Dans ce module, il est utilisé pour envoyer la clé de chiffrement générée à une URL spécifiée via RequestBin.

4.6 Docker et la Reconstruction de l'Infrastructure :

Une caractéristique clé de notre approche avec XSCAN est la mise en évidence de la facilité avec laquelle une infrastructure sécurisée comme Docker peut être reconstruite pour restaurer les données chiffrées par le faux ransomware. En utilisant Docker, chaque conteneur est isolé et peut être reconstruit à partir de zéro en cas de besoin. Ainsi, même si nos données sont chiffrées par le faux ransomware, il suffit de reconstruire le conteneur Docker à partir des sources fournies pour effacer le chiffrement et restaurer l'infrastructure dans son état d'origine.

Cette approche met en lumière les avantages de l'infrastructure basée sur les conteneurs, notamment la facilité de déploiement, de gestion et de récupération en cas de sinistre. En intégrant cette démonstration dans notre projet XSCAN, nous montrons aux utilisateurs comment une infrastructure bien conçue peut atténuer les risques associés aux menaces de sécurité telles que les ransomwares, tout en soulignant l'importance de mettre en œuvre des mesures de sécurité robustes pour protéger les données sensibles.