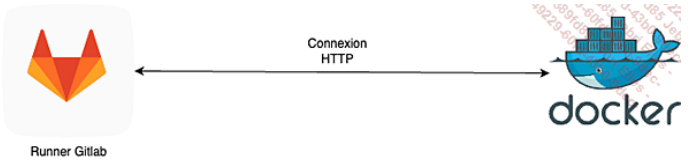
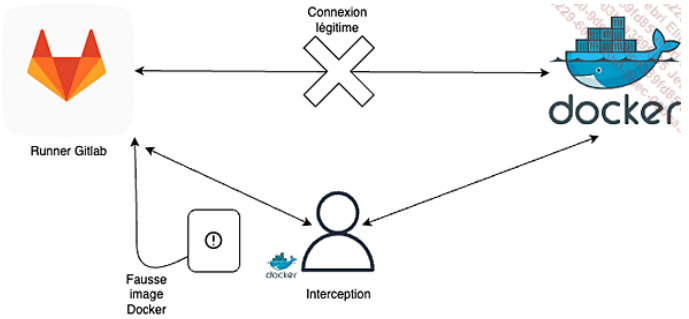
Les attaques sur les pipelines CI/CD

- Exposition accrue des environnements DevSecOps
- Attaques typiques : MITM, DDoS, Injections

1) Man-in-the-Middle (MITM) dans les pipelines



- Interception/modification des communications Runner ↔ registre/API
- Exemple : interception d'un pull d'image Docker non sécurisé



2) Attaques DDoS sur les pipelines CI/CD

- Cibler les endpoints d'API ou les runners (surcharge des builds)
- Impact : pipelines lents/indisponibles → blocage du delivery

3) Injections dans les pipelines CI/CD

- Variables non-sanitized (ex: BUILD_VERSION="1.2 ; rm -rf /")
- Fichiers YAML/commands mal configurés pouvant exécuter code arbitraire
- Voici un exemple de script Python permettant d’effectuer une validation des entrées avant exécution :

```
import shlex

user_input = "ls; rm -rf /"
if shlex.quote(user_input) != user_input:
    raise ValueError("Invalid input detected!")
```

Contre-mesures et bonnes pratiques (checklist)

- Utiliser TLS & mutual auth; signer images (cosign/Notary)
- Utiliser private registries + network controls (VPC/VNet, PrivateLink/PrivateEndpoint)
- Protéger CI variables, exiger protected branches, limiter permissions runner
- Implementer WAF/rate-limiting; autoscale & quotas pour runners
- Scan images (SCA/SAST), générer SBOMs, utiliser image allowlists

Exemples commandes & snippets utiles

- Activer Docker Content Trust (shell): export DOCKER_CONTENT_TRUST=1
- Bloquer IP dans AWS WAF (extrait): aws wafv2 create-ip-set --name BlockIPDDos --scope REGIONAL --addresses 200.1.2.3/32
- Valider input en Python: use shlex.quote or shlex.split to sanitize

Résumé & messages clés

- Pipelines CI/CD = surface d’attaque critique
- Combiner protections réseau, signing, scanning et hygiene CI
- Empower devs: templates sécurisés + automation (Vibecoding)