1 Développement d'un outil Python pour la gestion des flux réseau et sonintégration dans "Panda"

Lors de mon stage chez Cyklad, j'ai développé un outil Python pour analyser des logs réseau et générer des règles de firewall. Ce projet comprenait également l'intégration de cet outil à Panda, une interface graphique interne conçue par Cyklad, utilisée par l'équipe technique pour centraliser leurs outils de développement et de gestion réseau

Objectifs :

- Analyser les logs réseau pour en extraire des informations pertinentes (interfaces, IP, ports, protocoles).

- Automatiser la génération des règles de pare-feu adaptées aux flux observés.

- Intégrer l'outil à Panda, une plateforme regroupant tous les outils techniques pour simplifier et accélérer les workflows de l'équipe.

Étapes de développement et d'intégration :

Développement de l'outil d'analyse

L'outil commence par traiter des fichiers de logs bruts pour en extraire des données essentielles grâce à des expressions régulières.

Exemple d'extraction des données depuis une ligne de log :

```python
class Flow:
    def __init__(self, log_line):
        # Initialiser un objet Flow à partir d'une ligne de log et extraire les champs pertinents
        self.log_line = log_line  # Ligne de log brute
        self.int_src = None  # Interface source
        self.int_dst = None  # Interface destination
        self.ip_src = None  # Adresse IP source
        self.ip_dst = None  # Adresse IP destination
        self.dst_port = None  # Port de destination
        self.service = None  # Service associé au flux
        self.proto = None  # Protocole utilisé
        self.action = None  # Action appliquée (par ex., accepter ou bloquer)
        self.analyser_log()  # Analyser la ligne de log pour extraire les données
```

2 Génération de règles FortiGate

Les flux extraits sont regroupés et utilisés pour générer des règles adaptées. Ces règles peuvent inclure des adresses IP spécifiques, des ports, ou des protocoles identifiés dans les logs.

Exemple de génération de règles :

```
regle = (
    f"config firewall policy\n"
    f"edit 0\n"
    f"set srcintf {int_src}\n"
    f"set dstintf {int_dst}\n"
    f"set srcaddr {src_address_name}\n"
    f"set dstaddr {sous_reseau}\n"
    f"set action accept\n"
    f"set schedule always\n"
    f"set service {services_str}\n"
    f"set logtraffic all\n"
    f"next\n"
    f"end\n\n"
)
```

```
config firewall policy
edit 0
set srcintf internal
set dstintf wan1
set srcaddr LAN_[redacted]
set dstaddr IP_[redacted]
set action accept
set schedule always
set service IMAPS
set logtraffic all
next
end
```

Intégration dans Panda

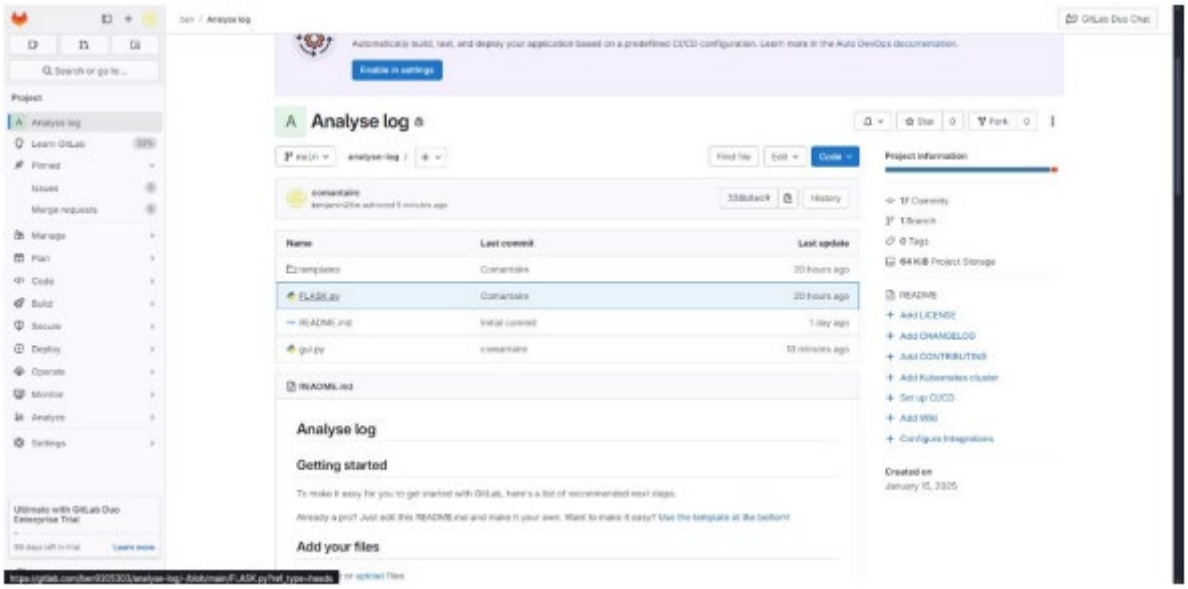
Panda est une interface graphique interne qui centralise les outils de développement et de gestion réseau. J'ai intégré l'outil Python pour permettre aux administrateurs de

Gestion du code avec GitLab

À la fin du projet, nous avons utilisé GitLab pour centraliser et partager le code source, tout en permettant des modifications si nécessaire. Bien que GitLab n'ait pas été utilisé tout au long du développement, il a joué un rôle important dans la phase finale pour garantir que l'outil soit accessible et documenté pour d'autres membres de l'équipe.

Voici comment GitLab a été utilisé en fin de projet :

Dépôt GitLab : Un dépôt GitLab a été créé pour stocker le code source. Cela a permis à l'équipe d'accéder facilement au projet via un lien partagé.



https://gitlab.com/ben9205303/analyse-log