Département RÉSEAUX ET TÉLÉCOMMUNICATIONS

Développement d'outils de supervision et implication dans l'évolution du réseau et de la sécurité informatique

RAPPORT D'ALTERNANCE (BUT2 ALT)

RÉDIGÉ PAR Lancelot MELLANO

ANNÉE UNIVERSITAIRE 2024 – 2025

[Photo à intégrer]

VILLE DE SAINTE-MAXIME

Lu et approuvé par

☐ **Autorise la diffusion en interne**
☐ **N'autorise pas la diffusion**

Mairie de Sainte-Maxime
24 Bd des Mimosas, 83120 Sainte-Maxime

Clément JACOB
Responsable du service Système & Réseaux
cjacob@ste-maxime.fr

Celine THEYS
Maitre de conférérence
celine.theys@univ-cotedazur.fr

Institut Universitaire de Technologie
Département Réseaux et Télécommunications
650, route des Colles
06560 Valbonne
Tél. : 04.93.95.51.70

Présentation de rapport d'alternance

Développement d'outils de supervision et implication dans l'évolution du réseau et de la sécurité informatique

Volume 1/1

Rapport rédigé par :

Lancelot MELLANO

Étudiant de BUT Réseaux et Télécommunication
IUT Nice Côte-d'Azur
Promotion 2023/2025

Maître d'apprentissage :
Clément JACOB
Responsable du Service Systèmes et Réseaux
Mairie de Sainte-Maxime

Tuteur pédagogique :
Mme Céline THEYS
Maitre de conférences
Dept. Réseaux et Télécommunications
IUT de Nice Côte d'Azur

Copies du présent document :
Département Réseaux et Télécommunications : 1
Mairie de Sainte-Maxime : 1
Lancelot MELLANO : 1

Remerciements :

Présentation de l’entreprise

a) Présentation générale :

Située dans le département du Var (83), en région Provence-Alpes-Côte d’Azur, la commune de Sainte-Maxime situé au cœur du golfe de Saint-Tropez, incarne un territoire à la fois vivant et tourné vers l’avenir. Son histoire remonte au Xe siècle, époque à laquelle des moines auraient fondé un monastère et donné au lieu le nom de sainte Maxime, en son honneur. Aujourd’hui, la ville compte environ 14 000 habitants et s’appuie sur une administration municipale solide, composée de près de 450 agents.

Figure 1 - Hôtel de ville de Sainte-Maxime

La mairie de Sainte-Maxime est une collectivité territoriale de niveau communal. Elle a pour mission de gérer les affaires publiques de la commune et d’offrir aux habitants des services de proximité variés, essentiels à la vie quotidienne et au développement du territoire. Ces services, à la fois administratifs, techniques, sociaux et culturels, participent activement au bon fonctionnement de la ville et à la qualité de vie des Maximois.

L'organisation de la mairie repose sur deux piliers complémentaires. D’un côté, l’exécutif municipal, composé du Maire (Vincent MORISSE), de ses adjoints et conseillers municipaux, qui définissent les grandes orientations politiques de la ville. De l’autre, une administration structurée en directions thématiques, chargées de mettre en œuvre ces décisions. Ces directions sont coordonnées par le Directeur Général des Services (DGS), le plus haut fonctionnaire de la collectivité.

Chacune est responsable d'un domaine précis, tel que les finances, les ressources humaines, les services techniques, les affaires juridiques, le développement durable, la police municipale ou encore les systèmes d'information.

Cette organisation permet de garantir la continuité du service public, d'assurer la mise en œuvre concrète des politiques municipales, et de maintenir un lien direct entre la commune et ses habitants. La mairie joue ainsi un rôle central dans la vie locale, en alliant proximité, efficacité et sens du service.

b) Présentation de mon service :

La Direction des Systèmes d'Information (DSI) est l'une des directions stratégiques de la mairie de Sainte-Maxime. Elle a pour mission de garantir la performance, la disponibilité et la sécurité de l'ensemble des systèmes informatiques municipaux. Cette direction transversale fournit aux autres services les outils numériques nécessaires à leur activité quotidienne : réseau, postes de travail, applications métiers, cybersécurité, téléphonie, etc. La DSI, dirigée par Stéphane GUGGINO, est structurée autour de trois entités principales : le service Projets Métiers, le service Archives et Documentation, et le service Systèmes et Réseaux, complétés par une assistante administrative :

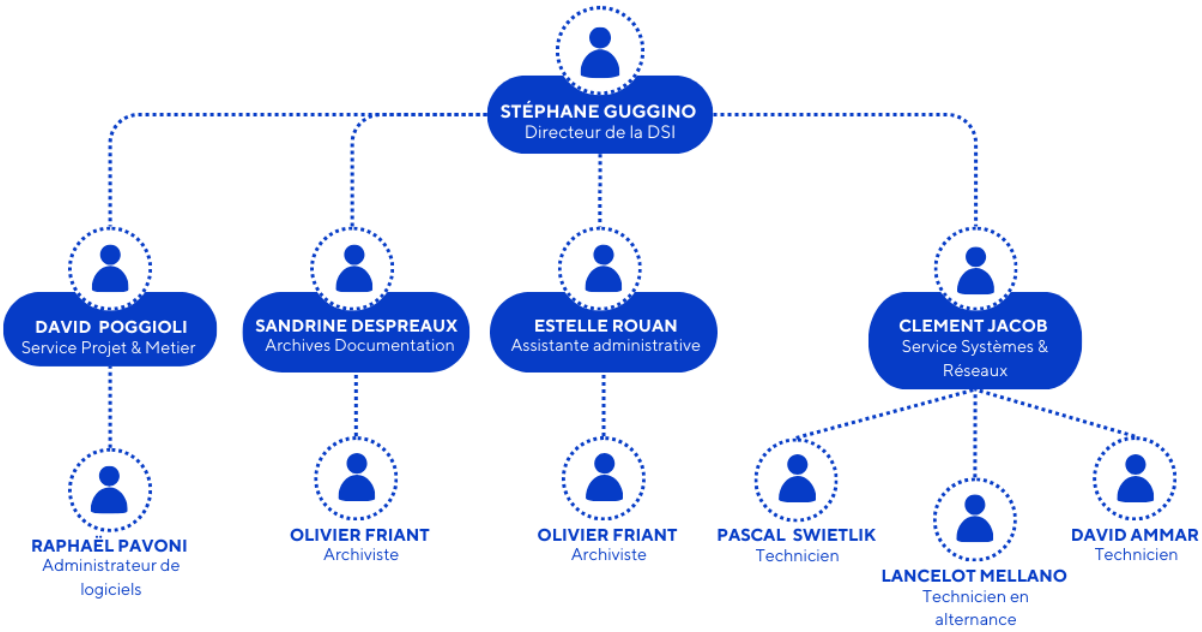


Figure 2 - Organigramme de la Direction des Système d'Information (DSI)

J'ai intégré le service Systèmes et Réseaux, dirigé par Clément JACOB. Cette unité constitue le cœur technique de la DSI. Elle est composée de trois agents à temps plein et d'un apprenti (moi). Sa mission est de garantir la disponibilité, la performance et la

sécurité de l'ensemble des infrastructures informatiques de la mairie. Elle assure la gestion des serveurs, du réseau interne, de la téléphonie IP, des sauvegardes, la mise en place de solutions de sécurité et veille à la continuité de service. Elle intervient également en appui technique pour le déploiement de matériel, la configuration des systèmes et la résolution d'incidents complexes.

En raison de notre effectif réduit, l'équipe est polyvalente : chacun intervient sur un large éventail de sujets, ce qui permet de maintenir une bonne réactivité et d'assurer une continuité de service efficace. La communication et la coopération sont renforcées par des points réguliers et un partage constant des compétences.

J'ai rejoint le service en septembre 2024, dans le cadre de mon alternance en BUT Réseaux & Télécommunications. Mon recrutement s'inscrit dans une volonté de renforcer la sécurité informatique, d'approfondir la connaissance du réseau, et de soutenir l'équipe sur les aspects support technique, création d'applications sur mesure et gestion des demandes utilisateurs.

Mon environnement de travail est composé d'un poste informatique complet, identique à celui des techniciens du service :

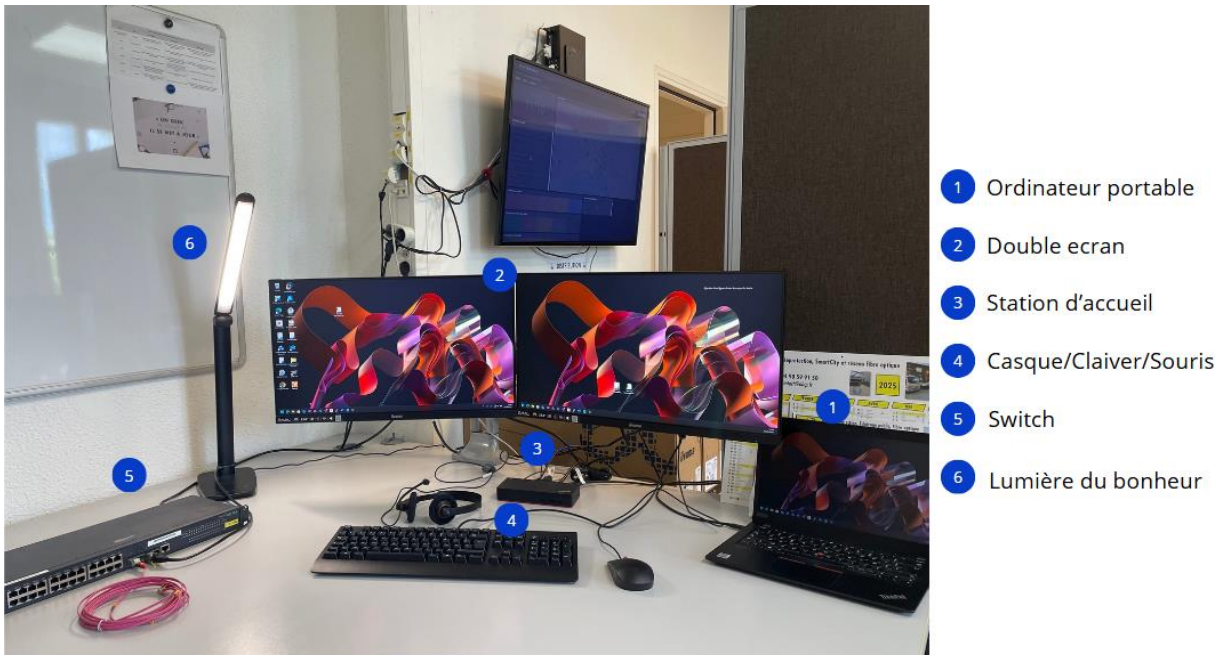


Figure 3 - Environnement de travail

Mon intégration dans le service Systèmes et Réseaux m'a permis de découvrir concrètement le fonctionnement d'une infrastructure informatique en collectivité, tout en apportant ma contribution aux projets de transformation numérique de la ville.

Introduction :

I) Conception et développement d'un outil automatisé de supervision des configurations des équipements actifs :

L'objectif de ce projet était de concevoir un outil automatisé pour surveiller de manière quotidienne les configurations des équipements actifs du réseau. Ces équipements, qui regroupent l'ensemble des switchs de niveau 2 et 3, jouent un rôle essentiel : ils permettent à l'ensemble des utilisateurs de la mairie de communiquer entre eux, d'accéder aux ressources internes, et d'échanger avec le monde extérieur. Ils constituent donc la pièce centrale qui fait fonctionner le réseau. Toute erreur de configuration ou toute modification non autorisée peut avoir un impact majeur sur la sécurité et la disponibilité des services. L'outil développé permet ainsi de vérifier en permanence que les configurations restent conformes aux configurations de référence définis, en détectant automatiquement toute anomalie ou tentative de piratage. La présente partie décrit la conception de cet outil, sa mise en œuvre technique, et les étapes de validation et de déploiement.

I.1) Conception fonctionnelle et algorithmique de l'outil de supervision :

Pour mener à bien ce projet et structurer efficacement le développement, nous avons commencé par formaliser le fonctionnement attendu de l'outil sous forme d'algorithmes. Cette démarche a permis de clarifier les étapes clés, de valider les choix techniques, et de faciliter la mise en œuvre des différentes fonctionnalités. Les quatre algorithmes suivants constituent le socle de l'outil de supervision :

A- Algorithme de récupération et de sauvegarde des configurations de référence :

Cet algorithme constitue la première étape clé du processus de supervision. Son objectif est de récupérer la configuration actuelle de chaque équipement actif (switch de niveau 2 ou 3) et d'en constituer une version de référence. Le principe retenu est que la première configuration enregistrée pour chaque équipement est considérée comme valide, et sert donc de configuration de référence. Par la suite, lors des audits quotidiens automatisés, cette configuration de référence sera systématiquement comparée à la configuration actuelle du switch au moment de chaque vérification, afin de détecter d'éventuelles modifications non autorisées ou erreurs de configuration.

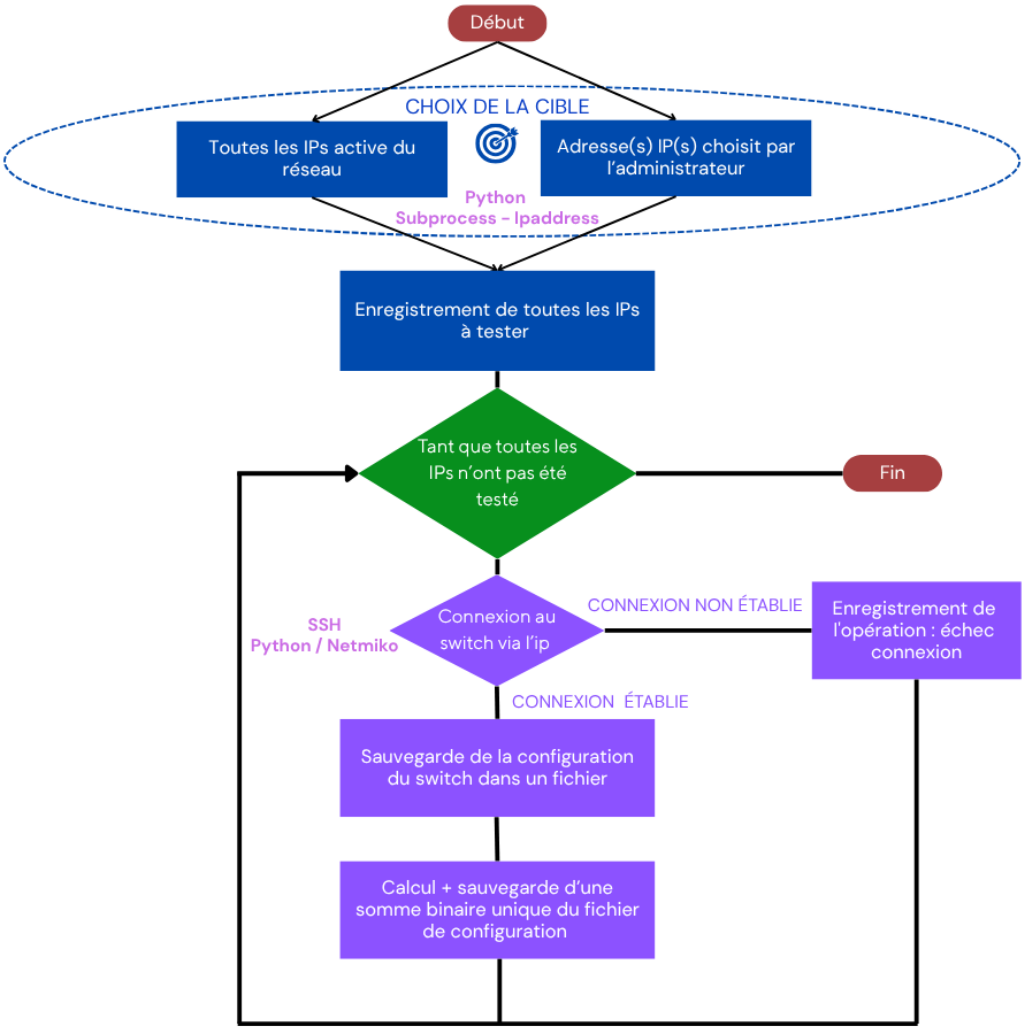


Figure 5 - Algorithme de récupération et de sauvegarde des configurations de référence

Dans un premier temps, l'algorithme permet de définir les équipements à traiter. L'administrateur peut choisir entre deux modes : analyser l'ensemble des switchs actifs identifiés sur le réseau, ou cibler un ou plusieurs équipements spécifiques. Les adresses IP à tester sont ensuite enregistrées. Le script parcourt ensuite l'ensemble de ces adresses de manière itérative.

Pour chaque équipement, une tentative de connexion SSH est effectuée à l'aide de la librairie Python Netmiko. En cas d'échec de connexion, l'opération est consignée comme telle dans les logs (historique). En cas de succès, la configuration du switch est automatiquement récupérée et sauvegardée localement sous forme de fichier texte.

Enfin, pour garantir l'intégrité de cette configuration de référence, une somme binaire (empreinte MD5) est calculée et enregistrée. Ce mécanisme permet de vérifier ultérieurement que le fichier de référence n'a pas été altéré, que ce soit de manière accidentelle ou malveillante.

Cet algorithme constitue ainsi la base du système de supervision : il permet de disposer, pour chaque équipement actif, d'une configuration de référence fiable et traçable.

B - Algorithme de comparaison des configurations et de génération du reporting :

Cet algorithme constitue le cœur du mécanisme d'audit quotidien automatisé. Son objectif est de vérifier que les configurations des équipements actifs n'ont pas subi de modifications non autorisées ou d'erreurs de paramétrage. Pour ce faire, il compare systématiquement la configuration de référence de chaque équipement à sa configuration actuelle lors de l'audit, et enregistre les éventuelles différences dans un rapport synthétique envoyé à l'équipe DSI :

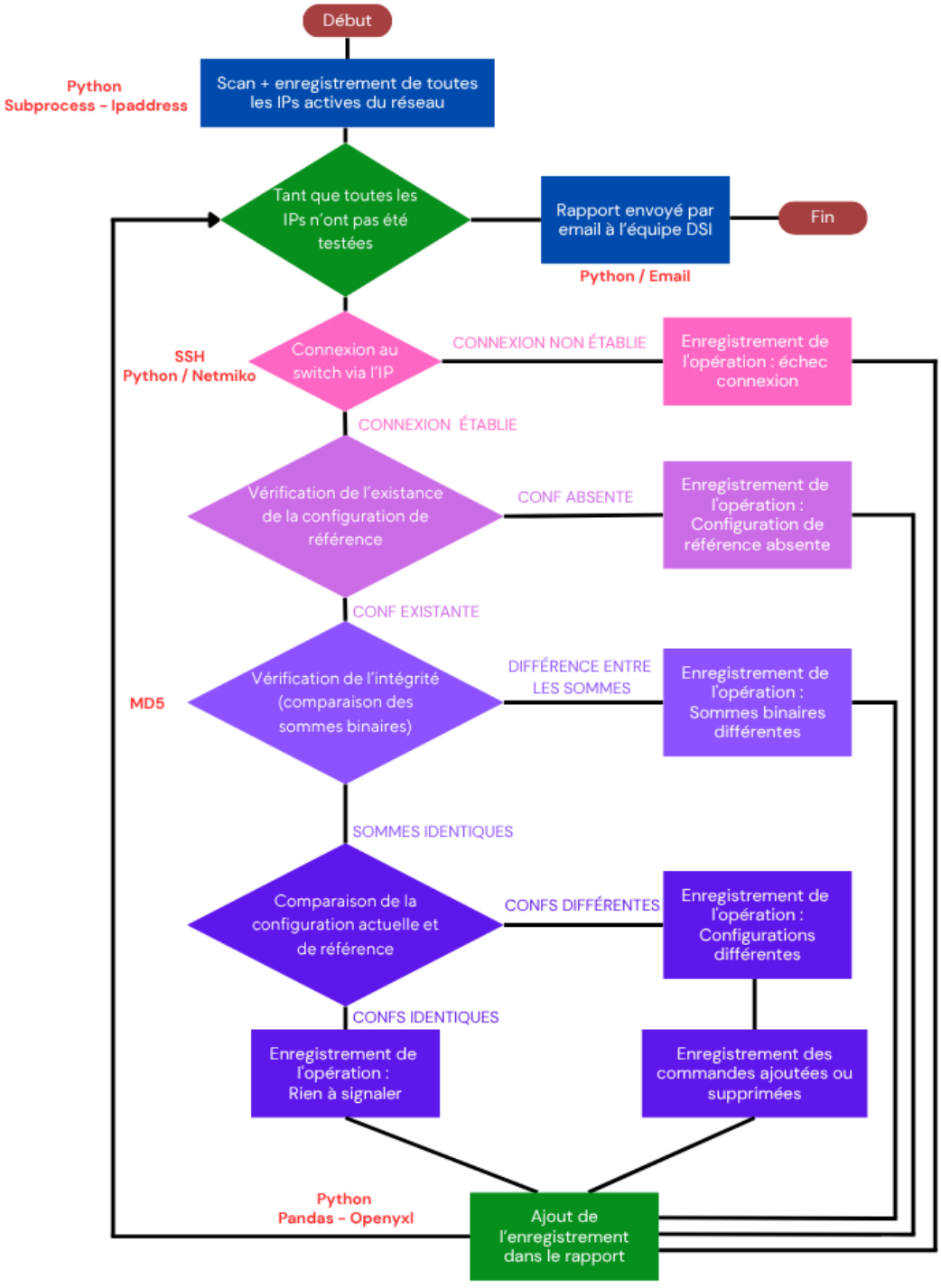


Figure 6 - Algorithme de comparaison des configurations et de génération du reporting

Le processus débute par un scan du réseau, identique à celui utilisé pour la constitution des configurations de référence, afin d'identifier les équipements actifs à contrôler.

Pour chaque équipement actif détecté, l'algorithme suit le déroulement suivant :

- Connexion SSH au switch : une tentative de connexion est effectuée. Si la connexion échoue, l'échec est consigné dans le rapport, et l'algorithme passe immédiatement au switch suivant.
- Vérification de la présence de la configuration de référence : si la connexion est établie, l'algorithme vérifie qu'une configuration de référence est disponible pour l'équipement concerné.
  Si cette configuration est absente, l'anomalie est enregistrée dans le rapport, et l'algorithme passe au switch suivant.
- Vérification de l'intégrité de la configuration de référence : si la configuration est disponible, l'algorithme vérifie d'abord son intégrité. Lors de la création initiale de la référence, une empreinte numérique (MD5) a été calculée et stockée à un instant où l'on savait que la configuration était valide. Lors de chaque audit, l'algorithme recalcule cette empreinte à partir du fichier de référence, et la compare à l'empreinte initialement enregistrée. Si une différence est détectée (empreinte MD5 non conforme), cela signifie que le fichier de référence a été altéré : l'anomalie est alors consignée, et le traitement passe directement au switch suivant.
- Comparaison de la configuration actuelle et de la configuration de référence : si l'intégrité de la configuration de référence est confirmée, l'algorithme procède alors à une comparaison détaillée entre la configuration actuelle du switch et sa référence. Les différences sont identifiées ligne par ligne, en distinguant les ajouts et les suppressions. Ces informations sont enregistrées dans un fichier de suivi et intégrées au rapport final.

Enfin, l'ensemble des résultats est consolidé dans un rapport automatique au format tableur, qui est transmis quotidiennement par e-mail à l'équipe DSI. Ce processus garantit une supervision fiable et continue des équipements actifs du réseau.

C - Algorithme de sécurisation et de gestion des identifiants :

Cet algorithme joue un rôle essentiel dans la sécurisation de l'outil de supervision. Il permet de protéger les informations sensibles, en particulier les identifiants d'accès aux équipements (login et mot de passe via SSH) et les identifiants pour l'envoi des rapports par e-mail. Afin de garantir que ces identifiants ne soient jamais stockés ni transmis en clair.

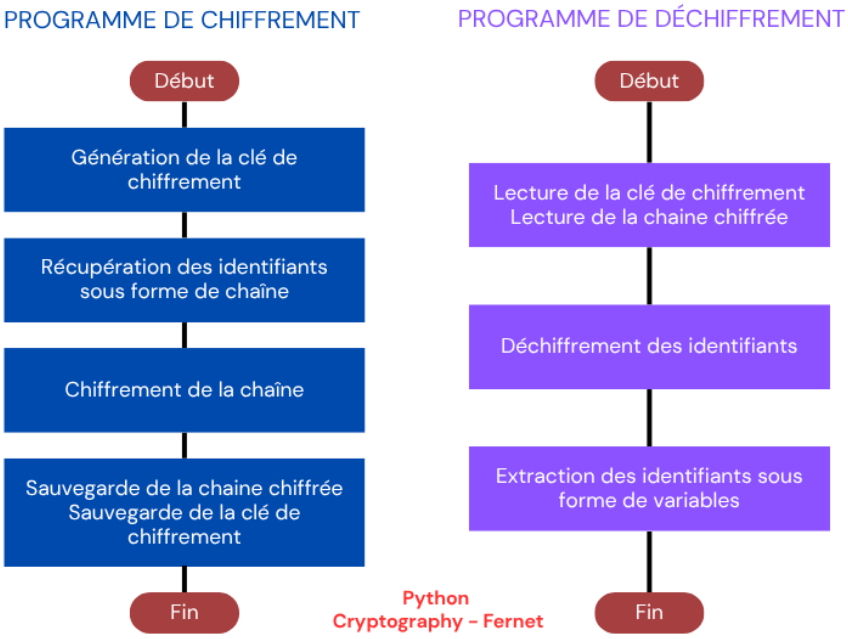


Figure 7 - Algorithme de sécurisation et de gestion des identifiants

Le mécanisme se décompose en deux volets :

- Programme de chiffrement : lors de l'initialisation, une clé de chiffrement unique est générée. Les identifiants sont alors rassemblés sous forme de chaîne, puis chiffrés à l'aide de cette clé. Les données chiffrées sont sauvegardées dans un fichier dédié, de même que la clé de chiffrement, stockée séparément.
- Programme de déchiffrement : lors de l'exécution de l'outil de supervision, la clé de chiffrement est lue, ainsi que les données chiffrées. L'algorithme procède alors au déchiffrement des identifiants, qui sont extraits et mis en mémoire sous forme de variables temporaires, utilisables par les différents modules de l'outil. Les identifiants restent ainsi protégés tout au long du cycle de vie du programme.

Grâce à ce mécanisme, les identifiants sensibles ne transitent jamais en clair sur le système, ni dans les scripts ni dans les rapports. Cela renforce significativement la sécurité globale de l'outil, en limitant les risques d'exploitation malveillante des accès aux équipements ou aux services.

D- Algorithme de mise à jour des configurations de référence via l'application graphique :

Dans le cadre du processus de supervision, il peut arriver que des modifications de configuration soient légitimes, par exemple suite à une intervention planifiée ou à une évolution du réseau. Lorsqu'un administrateur identifie ce type de modification dans le rapport quotidien, il doit alors mettre à jour la configuration de référence correspondante, afin que l'outil ne la considère plus comme une anomalie lors des audits suivants.

Pour simplifier cette opération, une application graphique dédiée a été développé.

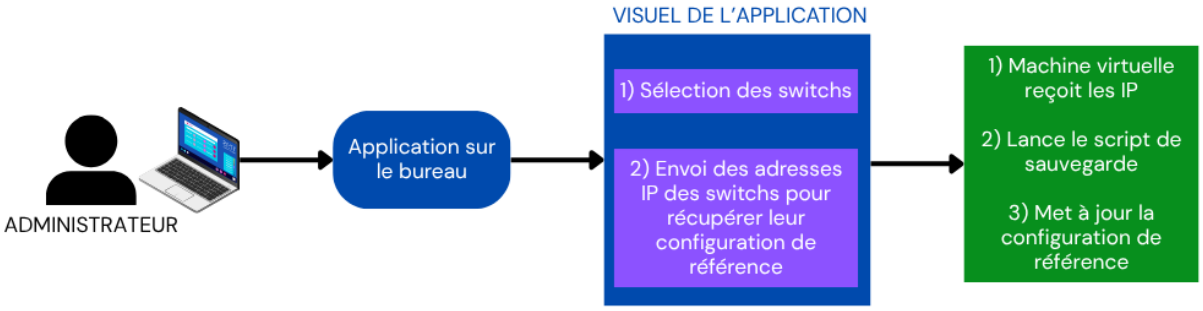


Figure 8 - Algorithme de mise à jour des configurations de référence via l'application graphique

Accessible directement depuis le poste de travail des administrateurs, cette interface permet de sélectionner facilement les équipements concernés, en affichant la liste des switchs actifs sur le réseau. Après sélection, l'application transmet les adresses IP des équipements ciblés à la machine virtuelle de supervision. Celle-ci exécute automatiquement le script de sauvegarde, qui récupère la configuration actuelle et la remplace en tant que nouvelle configuration de référence.

Ce mécanisme permet de sécuriser et de fluidifier le processus de mise à jour, en évitant les manipulations manuelles complexes. Il garantit que les configurations de référence restent toujours alignées avec l'état réel et valide du réseau. Cette phase de préparation, qui s'est étendue sur environ un mois et demi de septembre jusqu'à la mi-octobre, a joué un rôle déterminant dans la réussite de l'ensemble du projet.

I.2) Développement et implémentation des fonctionnalités principales :

Maintenant que le fonctionnement global de la solution et ses algorithmes clés ont été définis, cette partie présente la mise en œuvre concrète du projet. Il s'agit de donner vie aux algorithmes en développant les différents programmes nécessaires. Chaque section mettra en lumière les choix techniques retenus et les aspects clés du développement.

A – Programme de récupération et de sauvegarde des configurations de référence :

Comme décrit dans l'algorithme associé en première partie, cette fonctionnalité marque la première étape du processus de supervision. Son objectif est clair : établir une configuration de référence pour chaque équipement actif (switchs de niveau 2 ou 3). Pour mettre en œuvre ce fonctionnement, un script principal a été développé. Il permet de lancer l'opération selon deux modes :

- Un mode global, qui analyse tous les switchs détectés comme actifs sur le réseau. Ce mode est activé par l'administrateur lorsqu'il lance le programme avec une valeur de 1 passé en argument :

  « *python3 programme_recuperation_conf_reference 1* »

- Un mode ciblé, qui traite uniquement les équipements spécifiés manuellement par l'administrateur lors de l'appelle du programme :

  « *python3 programme_recuperation_conf_reference 192.168.10.1 192.168.10.2 ...* »

Voici la structure principale du programme, qui traite ces arguments et orchestre l'exécution :

```
if __name__ == "__main__":
    if len(sys.argv) < 2:
        print("Usage : python script.py <ip_address_1> <ip_address_2> ... | 1")
        sys.exit(1)

    # Récupérer les arguments passés
    args = sys.argv[1:]

    if args == ['1']:
        # Si "1" est passé en argument, exécuter pour tous les switchs
        print("Exécution pour tous les switchs actifs sur le réseau.")
        process_all_switches()
    else:
        # Sinon, traiter les adresses IP fournies
        for ip in args:
            print(f"Traitement de l'équipement {ip}...")
            process_switch(ip)
```

Figure 9 - Partie principale du programme de récupération et de sauvegarde des configurations de référence

Nous commençons avec le bloc *if __name__ == "__main__"*, qui correspond en Python au point de départ du programme. Cela signifie que tout ce qui se trouve à l'intérieur ne sera exécuté que si le fichier est lancé directement par l'utilisateur.

Juste après, une vérification est faite pour s'assurer qu'au moins un argument a été fourni (par exemple une adresse IP ou le chiffre 1). Si ce n'est pas le cas, le programme affiche un message d'aide pour expliquer comment l'utiliser correctement, puis s'arrête immédiatement grâce à *sys.exit(1)*. Cela permet d'éviter de lancer le programme par erreur ou sans cible définie.

Une fois les arguments correctement fournis, ils sont extraits depuis la ligne de commande à l'aide de la commande *sys.argv[1:]*. Cette ligne permet de récupérer tout ce qui a été passé après le nom du script.

Ensuite, le programme distingue les deux cas :

- Si l'argument reçu est exactement '1', cela signifie que l'utilisateur souhaite traiter tous les équipements actifs détectés automatiquement sur le réseau. Le script appelle alors la fonction process_all_switches(), qui effectue un scan réseau pour détecter les adresses IP en ligne, puis traite chaque switch identifié (connexion SSH, récupération de la configuration de référence, calcul + enregistrement de la somme MD5). Cette fonction est décrite plus en détail en [Annexe].

- Sinon, le script considère que ce sont des adresses IP précises qui ont été spécifiées. Il les parcourt une par une à l'aide d'une boucle for et appelle pour chacune la fonction process_switch(ip), qui gère le traitement d'un équipement donné (connexion SSH, récupération de la configuration de référence, calcul + enregistrement de la somme MD5). Cette fonction est également détaillée en [Annexe].

Grâce à cette structure, l'administrateur est libre de définir le périmètre d'analyse, qu'il s'agisse de l'ensemble du réseau ou d'équipements ciblés, pour récupérer les configurations de référence de façon souple et sécurisée.

B – Programme de comparaison des configurations et de génération du reporting :

Ce second programme est exécuté quotidiennement pour superviser l'état des configurations réseau, c'est donc lui qui fait l'audit de tous les équipements. Il repose sur les fichiers de référence générés au préalable, et a pour objectif de détecter automatiquement toute modification sur les équipements actifs de la collectivité.

```
prepare_excel() #vide le document rapport.xlsx

network_cidr = config["network"]  # Réseau à examiner avec le masque en CIDR ex: 192.168.1.0/24
switch_actif = scan_network(network_cidr)  # Récupération des adresses des switchs actifs

ip_list = []
for ip in ipaddress.ip_network(network_cidr, strict=False).hosts():
    ip_list.append(str(ip)) # Générer toutes les adresses IP théoriques dans le réseau


suivi_fichier = '/home/maintenance/Documents/SSH_NetWatcher/Programme_Python/suivi_config.json'

# Charger ou initialiser le fichier de suivi
if os.path.exists(suivi_fichier):
    with open(suivi_fichier, 'r') as f:
        suivi_config = json.load(f)
else:
    suivi_config = {}
```

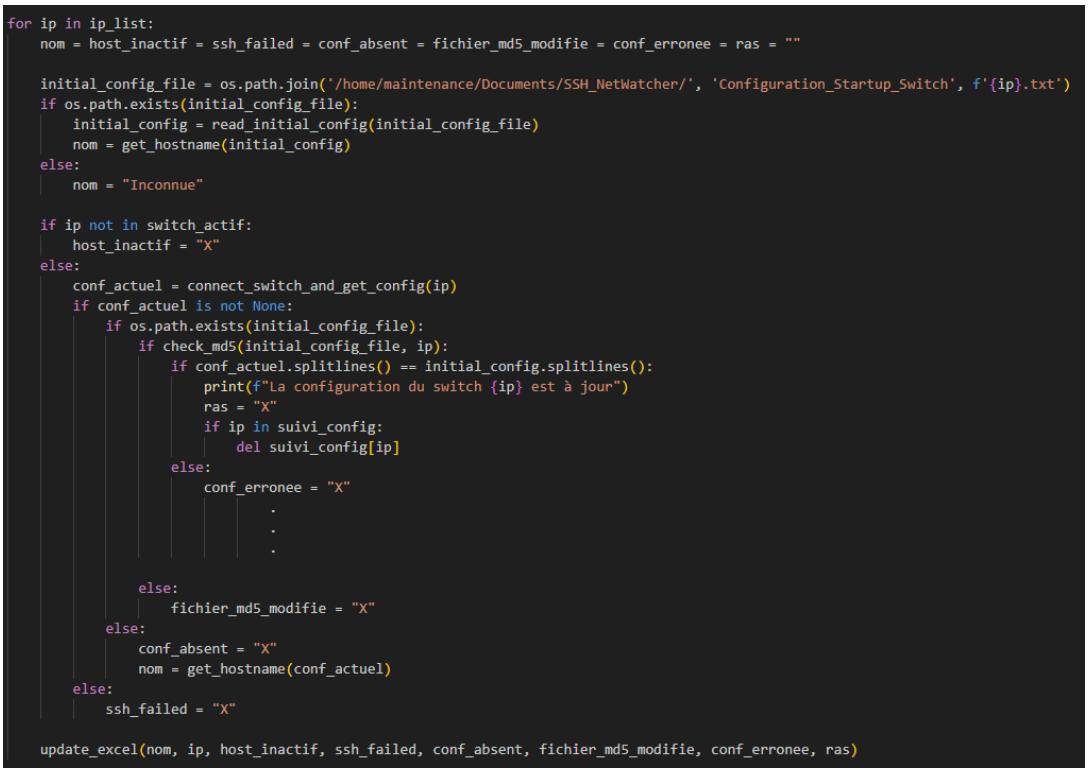
Figure 10 - Initialisation du programme de comparaison des configurations

Dès son lancement, il commence par réinitialiser le fichier Excel du rapport (rapport.xlsx) utilisé pour centraliser les résultats de l'audit, on utilisera ce rapport par-là suite. Il récupère ensuite :

- La liste des switchs réellement actifs (switch_actif), détectés grâce à un scan réseau réalisé sur le réseau qui a été donné network_cidr. Cette fonction est décrite en détaille en [Annexe].
- La liste de toutes les adresses IP du réseau(ip_list) qui a été donné par network_cidr.

Dans le cas concret de notre mairie, tous les équipements à superviser sont situés dans un réseau local configuré en /24, ce qui correspond à 254 adresses IP utilisables. Le programme analyse donc chaque IP du réseau, qu'elle corresponde ou non à un switch actif. Cela permet non seulement de vérifier la disponibilité des équipements attendus, mais aussi de détecter des adresses inoccupées, des hôtes inactifs ou des connexions SSH impossibles. Cette approche renforce la précision de l'audit.

Enfin, il charge (ou crée s'il n'existe pas encore) un fichier suivi_config.json. Ce fichier sert à conserver un historique détaillé des éventuelles modifications détectées dans les configurations au fil du temps : ajouts ou suppressions de commandes. Cela permet de suivre l'évolution des équipements dans le temps et de détecter tout changement non autorisé.



```
for ip in ip_list:
    nom = host_inactif = ssh_failed = conf_absent = fichier_md5_modifie = conf_erronee = ras = ""

    initial_config_file = os.path.join('/home/maintenance/Documents/SSH_NetWatcher/', 'Configuration_Startup_Switch', f'{ip}.txt')
    if os.path.exists(initial_config_file):
        initial_config = read_initial_config(initial_config_file)
        nom = get_hostname(initial_config)
    else:
        nom = "Inconnue"

    if ip not in switch_actif:
        host_inactif = "X"
    else:
        conf_actuel = connect_switch_and_get_config(ip)
        if conf_actuel is not None:
            if os.path.exists(initial_config_file):
                if check_md5(initial_config_file, ip):
                    if conf_actuel.splitlines() == initial_config.splitlines():
                        print(f"La configuration du switch {ip} est à jour")
                        ras = "X"
                        if ip in suivi_config:
                            del suivi_config[ip]
                    else:
                        conf_erronee = "X"
                                .
                                .
                                .

                else:
                    fichier_md5_modifie = "X"
            else:
                conf_absent = "X"
                nom = get_hostname(conf_actuel)
        else:
            ssh_failed = "X"

    update_excel(nom, ip, host_inactif, ssh_failed, conf_absent, fichier_md5_modifie, conf_erronee, ras)
```

Figure 11 - Partie principale du programme de comparaison des configurations

Le cœur du traitement commence avec une boucle qui parcourt l'ensemble des adresses IP du réseau (for ip in ip_list :). Pour chaque adresse, le programme initialise plusieurs indicateurs à vide. Ces indicateurs correspondent aux différentes étapes du test d'audit : hôte inactif, connexion SSH échouée, configuration de référence absente, empreinte MD5 non conforme, configuration erronée ou dans le meilleur des cas rien à signaler. Ces étapes correspondent à celles présentées dans la sous partie de l'algorithme B.

Le script cherche ensuite à retrouver la configuration de référence associée à l'adresse IP (fichier ip.txt stocké dans le dossier Configuration_Startup_Switch). Si le fichier existe, la configuration de référence est lue, et le nom du switch est extrait automatiquement grâce à une fonction qui récupère le nom d'hôte à partir du contenu (c'est la fonction get_hostname (), elle sera présentée en [Annexe]). Ce nom permet d'identifier clairement chaque équipement ce qui sera important pour le rapport ensuite généré. Si le fichier n'est pas présent, le nom est simplement noté comme « Inconnue ».

Ensuite, chaque étape est testée de manière successive, si une étape échoue, l'indicateur correspondant à cette étape sera marqué d'une croix (X) et le programme passera immédiatement au switch suivant. Ce système permet de tracer précisément le statut de chaque équipement audité.

Exemple : Si un switch n'est pas accessible via une connexion ssh alors :

host_actif= « », ssh_failed = « X », conf_absent = fichier_md5_modifie = conf_erronee = ras = ""

```python
if conf_actuel.splitlines() == initial_config.splitlines():
    print(f"La configuration du switch {ip} est à jour")
    ras = "X"
    if ip in suivi_config:
        del suivi_config[ip]
else:
    conf_erronee = "X"

    conf_actuel_dict = {}
    initial_config_dict = {}

    for line in conf_actuel.splitlines():
        line = line.strip()
        if line:
            conf_actuel_dict[line] = conf_actuel_dict.get(line, 0) + 1

    for line in initial_config.splitlines():
        line = line.strip()
        if line:
            initial_config_dict[line] = initial_config_dict.get(line, 0) + 1

    # Identifier les lignes ajoutées et supprimées
    added_lines = {
        line: count for line, count in conf_actuel_dict.items()
        if line not in initial_config_dict or conf_actuel_dict[line] > initial_config_dict.get(line, 0)
    }

    removed_lines = {
        line: count for line, count in initial_config_dict.items()
        if line not in conf_actuel_dict or initial_config_dict[line] > conf_actuel_dict.get(line, 0)
    }

    date_today = datetime.now().strftime("%d/%m/%Y")
    if ip not in suivi_config:
        suivi_config[ip] = {}

    suivi_config[ip][date_today] = {
        "ajout": added_lines,
        "suppression": removed_lines
    }

    with open(suivi_fichier, 'w') as f:
        json.dump(suivi_config, f, indent=4)
```

Figure 12 - Zoom sur la détection des différences entre les configurations

Le code représenté ci-dessus correspond aux trois petits points "..." mentionnés dans la figure 11. Il constitue une partie importante du programme : celle où la configuration actuelle du switch est différente de la configuration de référence. L'objectif est donc de détecter les modifications précises, ligne par ligne, dans un format compréhensible et exploitable.

Le fonctionnement est le suivant :

- D'abord, les deux configurations (référence et actuelle) sont parcourues ligne par ligne. Chaque ligne est stockée dans un dictionnaire (conf_actuel_dict pour l'actuelle, initial_config_dict pour la référence), avec :
    - La ligne de configuration en tant que clé.
    - Le nombre de fois où cette ligne apparaît comme valeur.

Ce format n'a pas été choisi par hasard : une première version de comparaison se contentait de vérifier si une ligne était présente ou non dans l'autre configuration. Cela fonctionnait en partie, mais ne permettait pas de détecter les cas où une même ligne apparaissait plusieurs fois dans un fichier, et une seule fois dans l'autre. Ces différences passaient inaperçues. Grâce à l'utilisation d'un dictionnaire avec un compteur, il est maintenant possible d'identifier non seulement les différences de contenu, mais aussi les différences de fréquence, ce qui garantit une comparaison bien plus précise et fidèle à la réalité réseau.

- Ensuite, deux structures sont générées :
    - added_lines : les lignes ajoutées ou présentes en plus grand nombre dans la configuration actuelle.
    - removed_lines : les lignes supprimées ou présentes en moins grand nombre que dans la configuration de référence.
- Enfin, ces écarts sont enregistrés dans une structure suivi_config, associant à chaque switch (ip) et à chaque jour, les lignes ajoutées et supprimées. Ces données sont stockées dans un fichier JSON (suivi_config.json) permettant une traçabilité complète des modifications dans le temps. Il sera ensuite utilisé pour rédiger le rapport final.

Exemple du fichier suivi_config :


```
{
    "192.168.1.6": {
        "12/06/2025": {
            "ajout": {
                "hostname Switch-3":1
            },
            "suppression": {
                "clock timezone GMT+1 add 01:00:00": 1,
                "clock summer-time CEST 02:00:00 March last Sunday 03:00:00 October last Sunday 01:00:00": 1,
                "clock protocol ntp": 1

            }
        },
        "13/06/2025": {
            "ajout": {},
            "suppression": {

                "password-control enable": 1,
                "undo password-control aging enable": 1,
                "undo password-control length enable": 1,
                "password-control login idle-time 0": 1
            }
        }
    }
}
```


Figure 13 - Exemple du fichier suivi_config qui permet de reporter les commandes différentes

Cette approche rend l'analyse fine, fiable et compréhensible pour un administrateur réseau. Elle offre un vrai gain pour la supervision de sécurité, en détectant rapidement toute anomalie ou modification non autorisée.

Lorsque nous avons atteint la fin des étapes, avant de passer à un autre switch, nous enregistrons le résultat de l'audit qui vient d'être réalisé dans un fichier excel grâce à la fonction update_excel () que nous pouvons voir à la fin de la figure 11 (Cette fonction est expliquée en [Annexe]). Ce fichier constitue le cœur de notre rapport, c'est grâce à lui que les administrateurs pourront suivre l'état des configurations des équipent actifs du réseau.

Exemple du fichier Excel :

Fichier avant audit :

	A	B	C	D	E	F	G	H
1	Nom	IP des Switch	Hôte inactif	Connexion SSH échoué	Fichier de conf initial absent	Incohérence md5	Conf actuel erroné	RAS
2								
3								

Figure 14 - Squelette du fichier Excel constituant le rapport final

Fichier après audit :

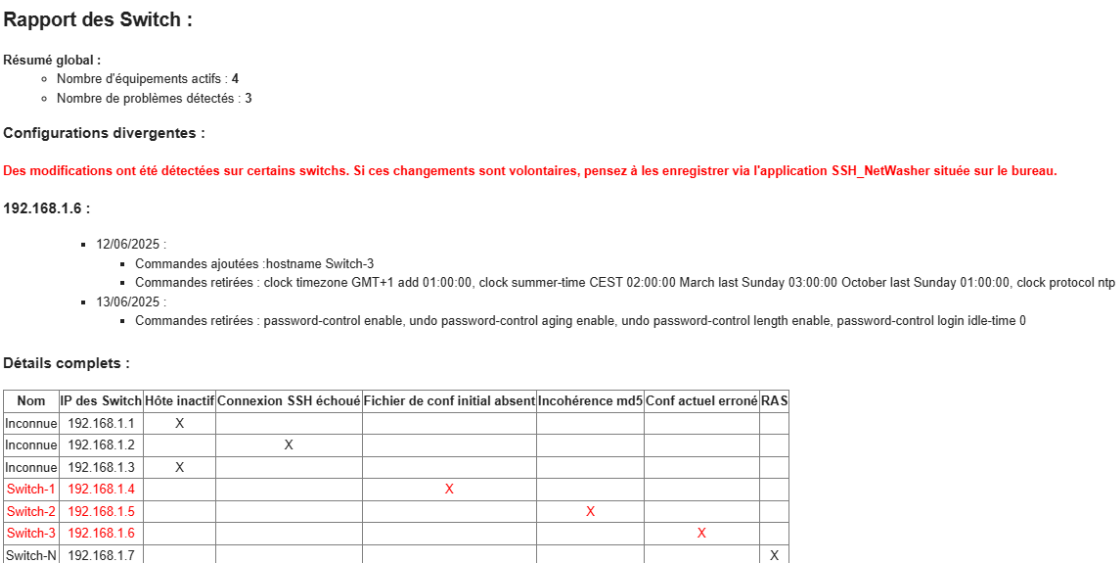
	A	B	C	D	E	F	G	H
1	Nom	IP des Switch	Hôte inactif	Connexion SSH échoué	Fichier de conf initial absent	Incohérence md5	Conf actuel erroné	RAS
2	Inconnue	192.168.1.1	X					
3	Inconnue	192.168.1.2			X			
4	Inconnue	192.168.1.3	X					
5	Switch-1	192.168.1.4			X			
6	Switch-2	192.168.1.5				X		
7	Switch-3	192.168.1.6					X	
8	Switch-N	192.168.1.7						X

Figure 15 - Fichier Excel remplit suite à un audit

C’est grâce à ces deux fichiers (suivi_config.json et rapport.xlsx) que le script construit ensuite un rapport automatique envoyé chaque jour par e-mail. Pour cela, plusieurs bibliothèques Python sont utilisées : Pandas permet de lire et manipuler facilement les données du tableau Excel, tandis que Openpyxl est utilisée en arrière-plan pour gérer le fichier au format .xlsx.

Les informations collectées sont ensuite mises en forme en HTML pour générer un e-mail clair et structuré. Le message contient un résumé global (nombre de problèmes détectés, nombre d'équipements actifs), ainsi qu'un tableau récapitulatif de l'état de chaque switch. Si des différences sont détectées dans les configurations, elles sont également listées ligne par ligne pour faciliter l'analyse par l'administrateur.

Enfin, grâce aux bibliothèques smtplib et ssl, le programme envoie ce rapport automatiquement via une adresse mail dédiée, en utilisant une connexion sécurisée au serveur SMTP. L'envoi est entièrement automatisé, garantissant ainsi une supervision continue et fiable sans intervention manuelle. Voici un exemple du mail reçu dans notre cas :



Rapport des Switch :

Résumé global :

- Nombre d'équipements actifs : **4**
- Nombre de problèmes détectés : **3**

Configurations divergentes :

Des modifications ont été détectées sur certains switchs. Si ces changements sont volontaires, pensez à les enregistrer via l'application SSH_NetWasher située sur le bureau.

192.168.1.6 :

- 12/06/2025 :
  - Commandes ajoutées :hostname Switch-3
  - Commandes retirées : clock timezone GMT+1 add 01:00:00, clock summer-time CEST 02:00:00 March last Sunday 03:00:00 October last Sunday 01:00:00, clock protocol ntp
- 13/06/2025 :
  - Commandes retirées : password-control enable, undo password-control aging enable, undo password-control length enable, password-control login idle-time 0

Détails complets :

Nom	IP des Switch	Hôte inactif	Connexion SSH échoué	Fichier de conf initial absent	Incohérence md5	Conf actuel erroné	RAS
Inconnue	192.168.1.1	X					
Inconnue	192.168.1.2		X				
Inconnue	192.168.1.3	X					
Switch-1	192.168.1.4			X			
Switch-2	192.168.1.5				X		
Switch-3	192.168.1.6					X	
Switch-N	192.168.1.7						X

Figure 16 - Exemple rapport final reçu par email

Pour faciliter la lecture du rapport par les administrateurs, les problèmes détectés sont automatiquement affichés en rouge, permettant ainsi un aperçu instantané des équipements concernés. Le nombre d'équipements actifs est calculé en comptant chaque ligne dont la colonne « Nom » contient une valeur différente de « Inconnue », ce qui indique qu'il s'agit bien d'un switch identifié. Ensuite, le nombre de problèmes est incrémenté uniquement lorsqu'une croix est présente dans l'une des colonnes de test (autre que « RAS ») et que le nom de l'équipement est connu.

Ainsi se déroule l'audit automatisé de l'ensemble des équipements actifs de notre réseau. L'ensemble des résultats est ensuite synthétisé dans un rapport structuré, enrichi de statistiques et d'alertes visuelles, puis automatiquement envoyé par mail à l'équipe informatique pour assurer un suivi quotidien efficace et réactif. L'automatisation de ce programme est assurée par une tâche planifiée via Crontab, qui permet de l'exécuter automatiquement chaque jour à la même heure, sans à avoir besoin d'intervention humaine.

C - Programme de sécurisation et de gestion des identifiants :

La sécurisation des identifiants utilisés par l'outil de supervision (identifiants SSH et mot de passe de l'expéditeur mail) constitue un enjeu essentiel. Pour éviter tout stockage ou affichage en clair, nous avons mis en place un système simple mais efficace basé sur la bibliothèque Python Cryptography – Fernet. Le processus repose sur deux programmes distincts, chacun ayant un rôle spécifique :

Le programme de chiffrement, lancé une seule fois à l'initialisation du projet, permet de créer une clé de chiffrement, de chiffrer les identifiants regroupés dans une chaîne, puis de sauvegarder séparément la clé et les données chiffrées dans deux fichiers : key.txt et informations.txt :

```python
# Générer une clé ou charger une clé existante
key = Fernet.generate_key()
cipher = Fernet(key)

# Identifiants à chiffrer
credentials = f"id_ssh={config['id_ssh']}\nmdp_ssh={config['mdp_ssh']}\nmdp_mail={config['mdp_auth_mail']}"

# Chiffrer le texte
encrypted_data = cipher.encrypt(credentials.encode())

# Sauvegarder le texte chiffré dans un fichier
with open('/home/maintenance/Documents/SSH_NetWatcher/Programme_Python/informations.txt', 'wb') as file:
    file.write(encrypted_data)

# Sauvegarder la clé séparément
with open('/home/maintenance/Documents/SSH_NetWatcher/Programme_Python/key.txt', 'wb') as key_file:
    key_file.write(key)
```

Figure 17 - Programme de chiffrement

Le programme de déchiffrement, utilisé automatiquement à chaque exécution de l'outil, relit la clé et les données chiffrées, puis reconstitue dynamiquement les identifiants dans des variables Python. Ces variables sont ensuite utilisées par les autres programmes, sans jamais exposer les informations sensibles :

```python
# Charger la clé
key = open('/home/maintenance/Documents/SSH_NetWatcher/Programme_Python/key.txt', 'rb').read()
cipher = Fernet(key)

# Lire et déchiffrer le fichier
with open('/home/maintenance/Documents/SSH_NetWatcher/Programme_Python/informations.txt', 'rb') as file:
    encrypted_info = file.read()

decrypted_data = cipher.decrypt(encrypted_info).decode()

# Traiter les identifiants

lines = decrypted_data.splitlines()
informations = {line.split('=')[0]: line.split('=')[1] for line in lines}
id_ssh = informations["id_ssh"]
mdp_ssh = informations["mdp_ssh"]
mdp_mail = informations["mdp_mail"]
```

Figure 18 - Programme de déchiffrement

Grâce à ce mécanisme, les identifiants confidentiels restent protégés tout au long de leur cycle de vie. Ils ne sont jamais visibles en clair, ni dans le code, ni dans les fichiers (ces derniers disposent de droits stricts empêchant quiconque d'y accéder, même pour consulter les données chiffrées), ni dans les journaux d'exécution. Ce choix renforce considérablement la robustesse de l'outil face aux risques d'exploitation ou de compromission des accès.

D – Fichier de configuration global :

L'un des objectifs majeurs de ce projet, au-delà de l'automatisation des audits des équipements actifs du réseau de la mairie, était de garantir la pérennité et la facilité d'utilisation de l'outil, même pour un administrateur ne maîtrisant pas le développement. Pour atteindre cet objectif, nous avons mis en place un fichier de configuration global (donne.json), qui permet d'adapter ou de faire évoluer l'outil sans modifier le code source.

Ce fichier regroupe tous les paramètres nécessaires au bon fonctionnement de l'outil :

- Les identifiants utilisés pour les connexions SSH et l'envoi d'emails.
- Les informations relatives à la messagerie (expéditeur, destinataires, objet du mail, contenu...).
- Le réseau à superviser.
- Les spécificités des différents types de switchs pris en charge (commandes de consultation, de sauvegarde, mode privilégié, etc.).

Ainsi, pour intégrer un nouveau type de switch, ou modifier des informations sensibles (comme un mot de passe ou une adresse email), il suffit de mettre à jour ce fichier sans avoir à toucher au code Python. Cette approche garantit une grande souplesse dans la maintenance de l'outil.

L'accès aux données de ce fichier est centralisé grâce à une simple fonction d'importation :

```python
def load_json(json_file):
    with open(json_file, 'r') as f:
        return json.load(f)
config = load_json('/home/maintenance/Documents/SSH_NetWatcher/Programme_Python/donne.json')
```

Figure 19 - Chargement du fichier de configuration globale

```json
{
    "network": "192.168.1.0/24",
    "objet_mail":"Rapport de test des switchs du Vlan 192.168.1.0/24 : ",
    "corps_mail": "Rapport de test des switchs du Vlan 192.168.1.0/24 : ",
    "src_mail": "maintenance@ste-maxime.fr",
    "auth_mail":"smtprelais@ste-maxime.fr",
    "mdp_auth_mail": "...",
    "dst_mail": ["lmellano@ste-maxime.fr","damar@ste-maxime.fr","cjacob@ste-maxime.fr","pswietlik@ste-maxime.fr"],
    "mdp_ssh": "...",
    "devices": {
        "hp_comware": {
            "type": "hp_comware",
            "affiche_conf_courrante": "display current-configuration",
            "mode_privilegie": "system-view",
            "cmd_save": ["save", "Y", "", "Y"]
        },
        "hp_procurve": {
            "type": "hp_procurve",
            "affiche_conf_courrante": "show running-config",
            "mode_privilegie": "system-view",
            "cmd_save": ["save"]
        },
        "exemple_device": {
            "type": "device_type",
            "affiche_conf_courrante": "commande correspondante au switch pour afficher la conf courrante",
            "mode_privilegie": "commande correspondante au switch pour accéder au mode privilegié",
            "cmd_save": ["commande ou suite de commande pour faire une sauvegarde sur ce type de switch "]
        }
    }
}
```

Figure 20 - Aperçu du fichier de configuration globale

E- Application de mise à jour des configurations de référence :

Tous les programmes présentés précédemment sont hébergés sur un serveur virtuel installé dans nos locaux. Ce choix permet de centraliser les outils de supervision sur une machine dédiée, simplifiant ainsi la maintenance des équipements actifs et assurant une structure pérenne.

Il est également important de comprendre que les configurations de référence sont amenées à évoluer, notamment lors de modifications du réseau ou de l'ajout de nouvelles fonctionnalités. Pour éviter que ces changements légitimes soient continuellement signalés comme des anomalies, il est essentiel de mettre à jour la configuration de référence correspondante. Comme il est indiqué dans le mail.

Afin de faciliter cette opération et d'éviter aux administrateurs d'avoir à se connecter manuellement au serveur virtuel, une application graphique a été développée. Elle leur permet d'effectuer ces mises à jour en toute simplicité, depuis leur poste de travail.

Voici à quoi ressemble l'application :

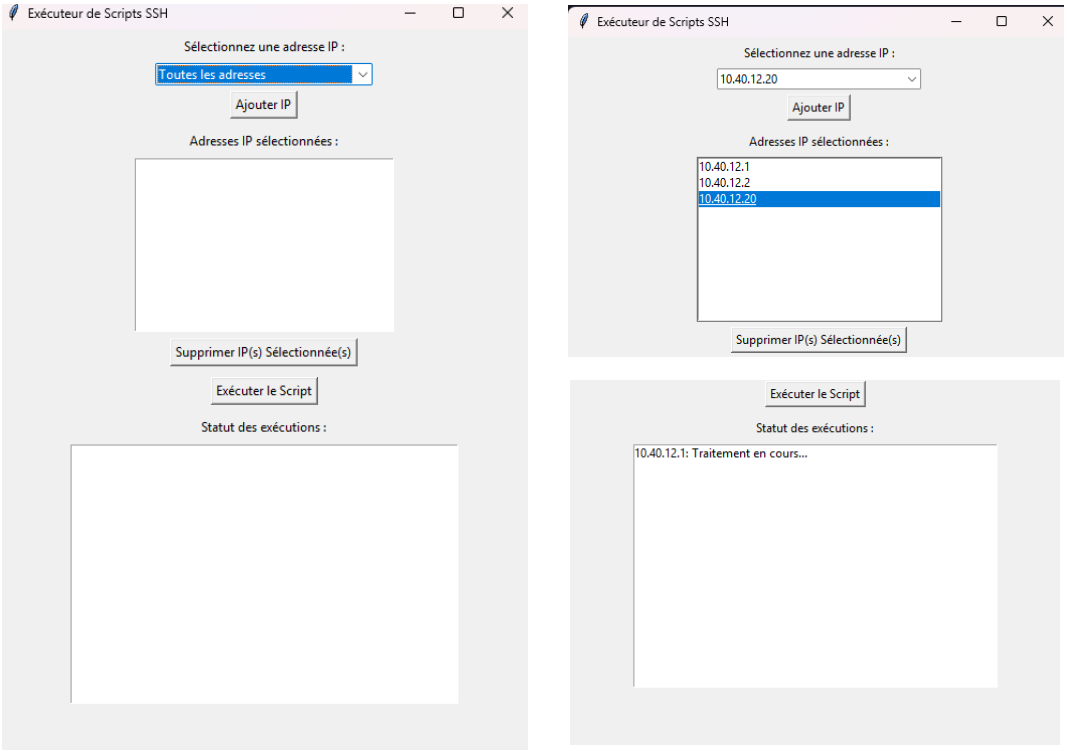


Figure 21 - Aperçu de l'application de mise à jour des configurations de référence

Sur le plan technique, l'interface repose sur la bibliothèque Tkinter de Python, et utilise Paramiko pour établir une connexion SSH sécurisée avec le serveur virtuel. Chaque administrateur dispose d'un accès configuré avec une paire de clés SSH (clé publique/clé privée), garantissant un haut niveau de sécurité sans qu'aucun identifiant ne transite en clair sur le réseau.

Lorsqu'un administrateur clique sur « Exécuter le script », deux cas se présentent :

- S'il a sélectionné « Toutes les adresses », le programme envoie la commande suivante via SSH : *« python3 programme_recuperation_conf_reference 1 »*
  Cette commande permet de sauvegarder la configuration de référence de tous les équipements actifs du réseau, comme expliqué dans le point A.
- S'il a sélectionné des adresses spécifiques, la commande exécutée est de la forme : *« python3 programme_recuperation_conf_reference 192.168.10.1 192.168.10.2… »*
  Elle permet alors de mettre à jour uniquement les configurations des équipements ciblés, comme expliqué dans le point A.

Enfin, l'application a été compilée en exécutable (.exe) à l'aide de l'outil PyInstaller, ce qui facilite son déploiement et son exécution sur les postes des administrateurs. Le code complet de l'interface est présenté en [Annexe].

III) Validation, tests et déploiement de l'outil en production

A - Méthodologie de tests et fiabilisation de l'outil :

Avant de déployer l'outil de supervision en production, une phase de test rigoureuse a été mise en place. L'objectif était de garantir son bon fonctionnement, sa robustesse face aux cas particuliers, ainsi que la fiabilité des résultats produits. Nous avons commencé par des tests fonctionnels manuels sur un environnement isolé : quelques switchs ont été configurés pour simuler différentes situations (connexion SSH réussie ou échouée, configuration absente, configuration modifiée, etc.). Ces scénarios ont permis de valider l'algorithme général et le bon enchaînement des étapes de vérification. Des tests de résilience ont également été menés pour observer le comportement de l'outil en cas d'erreur (coupure réseau, fichiers manquants, droits insuffisants…). Ces vérifications ont notamment permis de renforcer la gestion des exceptions, d'ajouter des messages explicites dans les logs, et de garantir la poursuite du traitement en cas d'incident sur un seul équipement. Cette démarche nous a permis de corriger plusieurs points avant la mise en production qui seront présentés dans la section suivante.

B – Problèmes rencontrés et ajustements apportés

Plusieurs difficultés ont émergé au cours du développement de l'outil, ce qui m'a amené à ajuster certains choix techniques afin d'assurer sa fiabilité et sa cohérence.

L'un des premiers problèmes concernait la détection des écarts de configuration. Initialement, la comparaison entre la configuration actuelle et la configuration de référence se faisait uniquement ligne par ligne, sans tenir compte du nombre d'occurrences. Cela posait problème lorsque, par exemple, une même ligne apparaissait une seule fois dans la configuration de référence, mais plusieurs fois dans la configuration actuelle : l'outil ne détectait alors aucune différence, malgré une modification réelle. Pour corriger cela, j'ai opté pour une représentation sous forme de dictionnaires, dans lesquels chaque ligne est utilisée comme clé et le nombre d'occurrences comme valeur. Ce système a permis d'identifier précisément les ajouts ou suppressions, même subtils.

J'ai également rencontré des difficultés avec l'établissement des connexions SSH sur certaines marques de switch. En effet, chaque constructeur peut implémenter des mécanismes de connexion différents, nécessitant des ajustements spécifiques (délai de réponse, prompts, comportement du shell…). Pour y répondre, j'ai mis en place un système de typage des équipements dans le fichier de configuration, ce qui permet d'adapter dynamiquement les paramètres de connexion selon la marque du switch.

L'envoi du mail de reporting a également soulevé des obstacles. Le serveur SMTP utilisé (Outlook 365) impose des exigences strictes : chiffrement TLS, port spécifique, format MIME correct, authentification renforcée… Plusieurs essais ont été nécessaires avant d'obtenir une version fonctionnelle. J'ai ajusté la configuration SMTP, renforcé la gestion des erreurs et épuré le format HTML du message pour assurer sa compatibilité et sa bonne délivrabilité. Ces ajustements successifs ont été décisifs pour garantir la stabilité de l'outil avant sa mise en production.

C – Expérience utilisateur et bilan du développement

Après plusieurs semaines d'utilisation en conditions réelles, l'outil a démontré son efficacité et sa simplicité d'usage. Le rapport généré quotidiennement est jugé lisible et compréhensible par les administrateurs, avec une présentation claire des équipements actifs et des éventuelles anomalies détectées. Les alertes sont précises et permettent d'identifier rapidement la nature du problème (connexion SSH échouée, incohérence de configuration, etc.). Le mécanisme de détection des changements de configuration a également fait ses preuves, ce qui facilite le suivi des interventions sur les équipements réseau. Par ailleurs, l'application graphique dédiée à la mise à jour des configurations de référence a été bien accueillie : elle simplifie l'opération en la rendant accessible directement depuis le poste de travail.

Globalement, l'outil a permis de renforcer la supervision du réseau tout en s'intégrant facilement dans les habitudes de travail des équipes techniques.
Le développement, les tests et le déploiement de l'ensemble des fonctionnalités ont représenté environ trois mois et demi (mi-octobre/ fin janvier) de travail intensif, réalisés en parallèle d'autres missions qui seront présenté ultérieurement.

L'objectif initial du projet, qui consistait à concevoir un outil automatisé de supervision des équipements actifs du réseau, a été pleinement atteint. L'outil permet aujourd'hui de surveiller en continu les configurations des switchs de la mairie, de détecter toute anomalie ou modification non autorisée, et de générer un reporting clair à destination des administrateurs.
Ce projet a été une expérience très enrichissante sur le plan technique et professionnel. Il m'a permis de renforcer mes compétences en programmation, mais surtout d'adopter une vision plus globale de la conception d'outils pérennes. J'ai appris à penser un projet au-delà de sa simple réalisation, en anticipant son évolution future, sa facilité d'utilisation et sa maintenabilité dans le temps. Une vision que je n'avais jamais adopté dans la réalisation de mes projets précédent. Cette première réalisation, centrée sur la supervision et la sécurisation des équipements actifs du réseau, constitue une brique technique essentielle. Dans la suite du rapport, l'approche se veut plus globale : il ne s'agira plus seulement de surveiller les switchs, mais bien d'élargir la supervision à l'ensemble du trafic réseau et de renforcer les mécanismes de sécurité à l'échelle de toute l'infrastructure.

II. Contribution à la sécurisation et à la modernisation du réseau de la collectivité (10 pages)

1. Mise en place d'un gestionnaire d'événement d'information et de sécurité(SIEM) et d'un centre d'opération de sécurité (SOC) (4 pages)

→ explication de ma contribution au projet de mise en place d'un SIEM :

- Identification des besoins de la mairie et délimitation du périmètre fonctionnel.
- Réalisation d'une étude de marché (recherche et analyse de différentes solutions du marché).
- Participation aux entretiens avec les prestataires.
- Aide au choix de la solution finale et justification de ce choix.

2. Découverte des équipements Aruba et configuration du contrôle d'accès réseau avec ClearPass (6 pages)

→ Ici tu développes :

- Le contexte et les enjeux liés à l'amélioration de la sécurité du réseau via ClearPass.
- Mon apprentissage des commandes spécifiques Aruba.
- La configuration réalisée sur les switchs pour intégrer ClearPass dans le processus d'authentification.
- La mise en production des configurations et les résultats obtenus.

III. Réalisation d'interventions techniques et de supports pour les utilisateurs internes (12 pages)

1. Déploiement automatisé d'environnements de travail pour les écoles de la collectivité (4 pages)

→ j'explique :

- Le besoin exprimé par les écoles et les services informatiques.
- La création d'un environnement de masterisation adapté.
- Le processus de déploiement des images système sur plusieurs machines.
- Les résultats obtenus et les gains de temps constatés.

2. Développement d'une application interne pour la gestion des consommables d'impression (4 pages)

→ Je présentes :

- Le besoin métier à l'origine du projet.
- Le développement du script et de l'interface graphique permettant d'automatiser la gestion des consommables.
- L'impact concret de cette application sur le travail quotidien.

3. Réalisation d'opérations de support technique et de maintenance pour les utilisateurs internes (4pages)

→ Je présente :

- Les différents types de tickets et d'incidents que tu as traités.
- Les opérations de création et de configuration d'environnements de travail.
- Les interventions ponctuelles réalisées (résolution de pannes, mises en place de nouveaux équipements, accompagnement utilisateurs).

Annexe