Cité des métiers et des Compétences l'Oriental

Module M106 : Automatiser les tâches d'administration

Rapport de :

TCP X : Communication Sécurisée avec Cryptographie et TCP

Ce projet vise à offrir une plateforme de chat sécurisée, utilisant des technologies de cryptographie, UFW et le protocole TCP pour assurer la confidentialité et l'intégrité des échanges

Préparé pour :

Lahcen SOUSSI
« Formatteur d'Automatisater les tâches »

Préparé par :

Haitam BEN DAHMANE IDRISSI
« Stagiaire en ID »

2024/2025

Résumé

Le projet TCP X est une initiative innovante visant à développer un serveur TCP permettant à plusieurs clients de communiquer entre eux de manière sécurisée. L'objectif principal de ce projet est de créer une application réseau robuste qui facilite les communications en temps réel entre des clients connectés tout en assurant la confidentialité des messages échangés grâce à un programme de cryptographie intégré, CryptiX.

Le projet se compose de trois composants principaux : un serveur TCP, des clients pouvant se connecter au serveur et communiquer entre eux, et un programme de cryptographie pour chiffrer et déchiffrer les messages des clients. Le serveur TCP est responsable de gérer les connexions des clients, de diffuser les messages entre eux et de maintenir la stabilité et la sécurité du réseau. Les clients, quant à eux, peuvent envoyer et recevoir des messages, qu'ils peuvent chiffrer ou déchiffrer en utilisant des commandes spécifiques pour garantir la confidentialité.

L'un des défis majeurs abordés dans ce projet est la sécurité des communications. Pour y répondre, le programme CryptiX a été développé pour permettre aux utilisateurs de chiffrer leurs messages avant de les envoyer et de les déchiffrer à la réception. En outre, le projet inclut des mesures pour protéger le serveur contre les attaques par déni de service (DoS), en utilisant des outils comme UFW (Uncomplicated Firewall) sous Linux.

Les résultats attendus de ce projet incluent une application fonctionnelle démontrant une communication en temps réel sécurisée entre plusieurs clients, une documentation détaillée de l'implémentation et des tests, ainsi que des recommandations pour des améliorations futures. Le projet TCP X représente une avancée significative dans la conception de systèmes de communication réseau sécurisés, offrant une base solide pour des développements ultérieurs et des applications pratiques dans divers domaines nécessitant des communications sécurisées.

Remerciements

Je tiens à exprimer ma gratitude à toutes les personnes et institutions qui ont contribué à la réalisation de ce projet TCP X.

Tout d'abord, je remercie profondément mon Formateur, LAHCEN SOUSSI, pour son soutien constant, ses conseils avisés et son encadrement tout au long de ce projet. Son expertise et ses encouragements ont été essentiels pour surmonter les défis rencontrés et pour mener ce projet à terme.

Je souhaite également remercier la Cité des métiers et des Compétences pour m'avoir offert les ressources nécessaires et l'environnement propice à la recherche et au développement. Un merci particulier à l'équipe de la bibliothèque pour leur aide précieuse dans la recherche de documents et de références.

Je suis reconnaissant envers mes collègues et amis, AMINE HABIBI et SOUFIANE LASFAR, pour leurs suggestions constructives et leur soutien moral. Leurs perspectives diverses et leurs discussions stimulantes ont enrichi mon travail et m'ont aidé à améliorer la qualité de ce projet.

Enfin, je voudrais exprimer ma reconnaissance envers ma famille, dont le soutien indéfectible et les encouragements constants m'ont permis de rester motivé et concentré. Leur patience et leur compréhension ont été des éléments cruciaux dans la réalisation de ce projet.

À tous ceux qui ont contribué de près ou de loin à l'aboutissement de ce travail, je vous adresse mes sincères remerciements.

Tables des Matières

Introduction

Contexte et Motivation

Avec l'évolution rapide de la technologie et l'augmentation exponentielle des communications numériques, la nécessité de sécuriser les échanges d'informations est devenue primordiale. Les réseaux TCP/IP, bien qu'essentiels pour la communication sur Internet, présentent des vulnérabilités qui peuvent être exploitées par des acteurs malveillants. Les attaques par déni de service (DoS) et l'interception de messages non sécurisés sont des menaces courantes qui peuvent compromettre la confidentialité et l'intégrité des données échangées.

Face à ces défis, le projet TCP X a été initié pour développer une solution robuste et sécurisée permettant des communications en temps réel entre plusieurs clients connectés à un serveur TCP. En intégrant des mécanismes de cryptographie pour protéger les messages échangés, ce projet vise à répondre aux besoins croissants de sécurité et de confidentialité dans les communications réseau.

Problématique

Les réseaux TCP traditionnels, bien qu'efficaces pour le transfert de données, manquent souvent de mécanismes intégrés pour garantir la sécurité des communications. Les principaux problèmes identifiés sont :

Vulnérabilité aux attaques DoS : Les serveurs TCP peuvent être submergés par un grand nombre de requêtes, rendant le service indisponible.

Absence de confidentialité : Les messages échangés entre clients peuvent être interceptés et lus par des tiers non autorisés, compromettant la confidentialité des données.

Ces problématiques nécessitent des solutions innovantes pour renforcer la sécurité des communications réseau.

Objectifs du Projet

Le projet TCP X vise à atteindre les objectifs suivants :

- Développer un serveur TCP capable de gérer plusieurs connexions clients simultanément, tout en maintenant une communication efficace et stable.
- Créer des clients TCP qui peuvent se connecter au serveur et échanger des messages en temps réel.
- Intégrer un programme de cryptographie (CryptiX) pour permettre le chiffrement et le déchiffrement des messages, assurant ainsi la confidentialité des échanges.
- Mettre en place des mesures de protection contre les attaques DoS en utilisant des outils comme UFW sous Linux pour sécuriser le serveur contre les attaques potentielles.
- Fournir une documentation détaillée de l'implémentation, des tests et des résultats, ainsi que des recommandations pour les améliorations futures.

Méthodologie Générale

Pour atteindre ces objectifs, le projet TCP X suivra les étapes méthodologiques suivantes :

- Étude et analyse des besoins, Comprendre les exigences fonctionnelles et non fonctionnelles du projet, y compris les aspects de sécurité.
- Conception du système, Définir l'architecture du serveur TCP, des clients, et du programme de cryptographie.
- Développement, Implémenter le serveur TCP, les clients et le programme CryptiX en utilisant le langage de programmation Python.
- Tests et validation, Effectuer des tests unitaires et d'intégration pour vérifier le bon fonctionnement du système. Simuler des attaques DoS pour évaluer la résilience du serveur et tester les fonctionnalités de cryptographie.
- Déploiement et documentation, Mettre en œuvre le système dans un environnement réel, documenter le processus de développement, les résultats des tests, et fournir des guides d'utilisation.
- Protection contre les attaques DoS, Configurer UFW sur le serveur Linux pour limiter les tentatives de connexion et bloquer les adresses IP malveillantes.

En suivant cette méthodologie, le projet TCP X vise à fournir une solution complète et sécurisée pour les communications réseau, répondant aux besoins actuels en matière de sécurité et de performance.

I- Contexte Théorique et Technique

A- Introduction au protocole TCP

Définition et historique

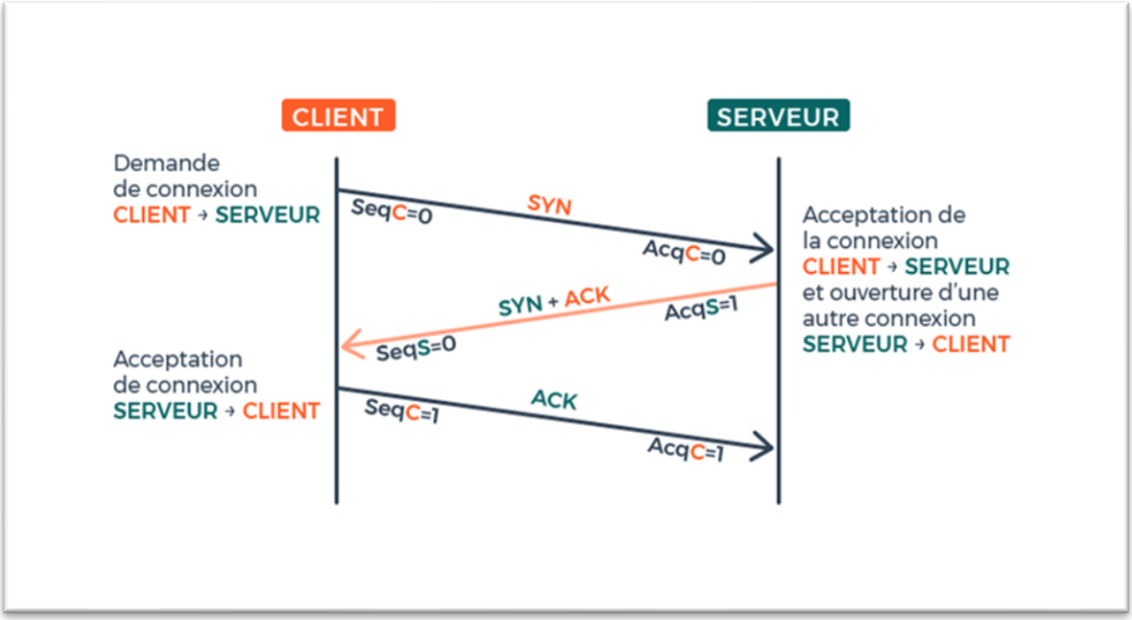
Le protocole TCP (Transmission Control Protocol) est l'un des principaux protocoles de la suite de protocoles Internet, souvent appelé TCP/IP. TCP est un protocole de communication orienté connexion, ce qui signifie qu'il établit une connexion fiable entre deux hôtes avant de transmettre des données. Introduit pour la première fois dans les années 1970, TCP a été développé par Vint Cerf et Bob Kahn et est devenu un standard en 1983 avec la publication de la norme RFC 793.

Le protocole TCP est conçu pour fournir un service de communication fiable et ordonné entre les applications fonctionnant sur des hôtes de réseau. Il est largement utilisé dans les applications nécessitant une transmission de données précise et en temps réel, comme les services web, le courrier électronique et les transferts de fichiers.

Fonctionnement général

Le protocole TCP fonctionne en trois phases principales : l'établissement de la connexion, le transfert de données et la terminaison de la connexion.

1- **Établissement de la connexion :** Cette phase utilise un processus de poignée de main en trois temps (three-way handshake) pour établir une connexion entre le client et le serveur. Ce processus implique l'échange de segments TCP avec des drapeaux SYN (synchronize), SYN-ACK (synchronize-acknowledge) et ACK (acknowledge).



« Document Détaillé de three-way handshake »

2- **Transfert de données:** Une fois la connexion établie, les données sont transmises entre le client et le serveur en segments TCP. Chaque segment contient un numéro de séquence pour assurer que les données sont reçues dans le bon ordre. TCP utilise également des mécanismes de contrôle de flux et de contrôle de congestion pour gérer la quantité de données envoyées et éviter la saturation du réseau.

3- **Terminaison de la connexion :** La connexion TCP est terminée en utilisant un processus de poignée de main en quatre temps, où les deux parties échangent des segments avec des drapeaux FIN (finish) et ACK pour s'assurer que toutes les données ont été transmises et reçues correctement avant de fermer la connexion.

B- Sécurité des communications réseau

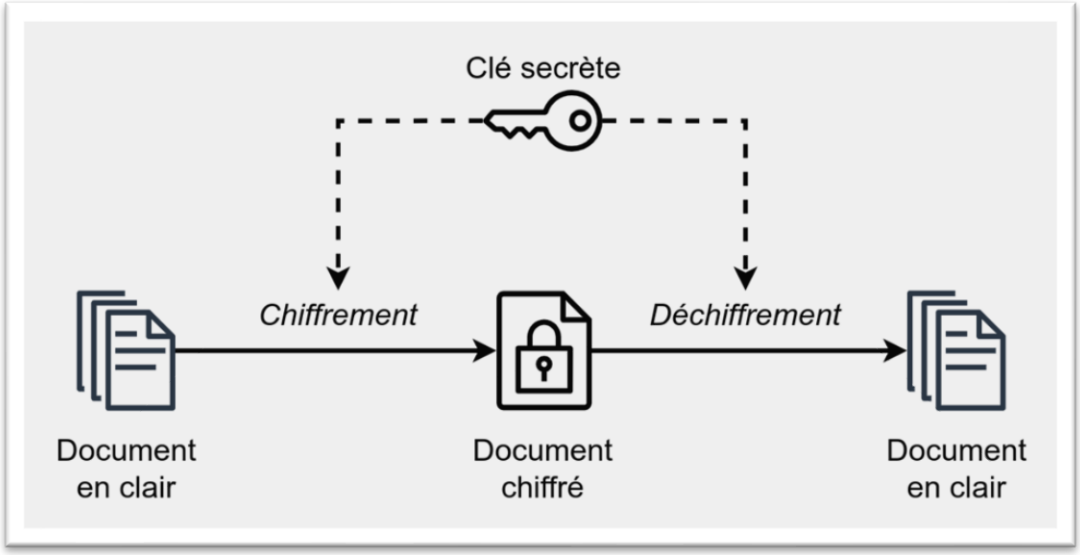
Cryptographie de base

La cryptographie est la science de la sécurisation des communications par le biais de techniques de chiffrement. Elle permet de convertir des données lisibles (texte en clair) en données illisibles (texte chiffré) à l'aide d'algorithmes et de clés de chiffrement. La

cryptographie est essentielle pour protéger la confidentialité et l'intégrité des informations transmises sur les réseaux.

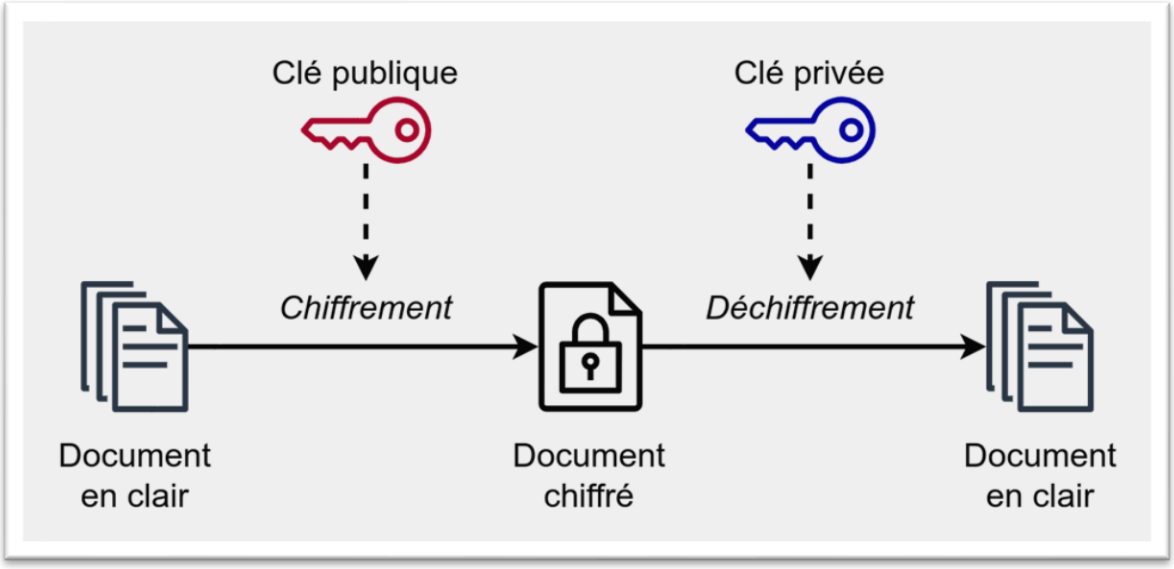
Les deux principales catégories de cryptographie sont :

Cryptographie symétrique, utilise une seule clé pour chiffrer et déchiffrer les données. L'exemple le plus connu est l'algorithme AES (Advanced Encryption Standard). La cryptographie symétrique est rapide et efficace pour chiffrer de grandes quantités de données, mais elle nécessite un moyen sécurisé pour échanger la clé de chiffrement entre les parties communicantes.



« Document Détaillé de Cryptographie symétrique »

Ensuit, ***Cryptographie asymétrique***, utilise une paire de clés, une clé publique pour le chiffrement et une clé privée pour le déchiffrement. L'exemple le plus connu est l'algorithme RSA. La cryptographie asymétrique est plus sécurisée pour l'échange de clés et pour la création de signatures numériques, mais elle est plus lente que la cryptographie symétrique.



« Document Détaillé de Cryptographie asymétrique »

Techniques de chiffrement et de déchiffrement

Les techniques de chiffrement et de déchiffrement permettent de convertir des données en texte chiffré pour les rendre illisibles à toute personne non autorisée. Les principales techniques incluent :

a. *Chiffrement par substitution :* Remplace chaque caractère du texte en clair par un autre caractère selon une règle définie. Un exemple simple est le chiffrement de César, où chaque lettre est décalée d'un certain nombre de positions dans l'alphabet.

b. *Algorithmes de chiffrement modernes :* Utilisent des combinaisons complexes de substitutions et de transpositions pour sécuriser les données. Les algorithmes AES et RSA en sont des exemples. AES utilise des blocs de 128 bits et des clés de 128, 192 ou 256 bits, tandis que RSA repose sur la factorisation de grands nombres premiers pour sa sécurité.

En intégrant ces techniques de cryptographie dans le projet TCP X, nous visons à garantir que les communications entre les clients soient sécurisées et que les données sensibles soient protégées contre les interceptions et les attaques.

II- Description du Projet et Explication de Script :

A- Présentation Générale de TCP X

Le projet "TCP X" vise à établir une plateforme de communication sécurisée basée sur le protocole TCP (Transmission Control Protocol). Conçu pour faciliter des échanges fiables et confidentiels entre plusieurs utilisateurs connectés simultanément, TCP X intègre un serveur centralisé, des clients interactifs et un module de cryptographie avancé appelé CryptiX.

B- Utilisation de Socket, Threading et Logging

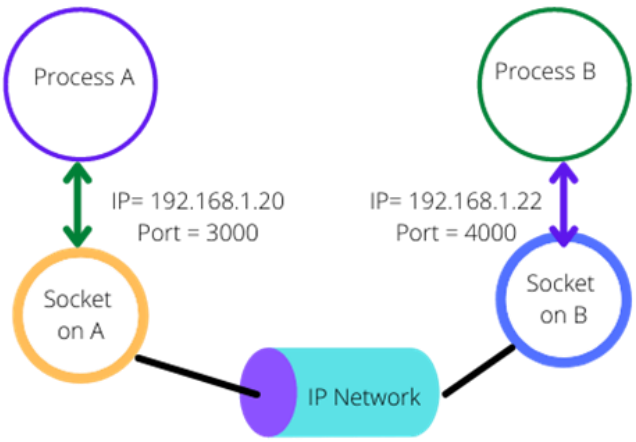
Code :

```
import socket
import threading
import logging
```

Explication :

Socket :

En Python, un socket est un concept fondamental de mise en réseau qui permet la communication entre des processus sur un réseau. Il fournit un point de terminaison pour envoyer et recevoir des données à travers un réseau, généralement en utilisant le protocole Internet (IP).



Les rôles de Socket sont :

Point de Communication : Les sockets servent de points de communication permettant à différents processus (exécutés sur le même ordinateur ou sur différents ordinateurs à travers un réseau) d'établir des connexions, d'envoyer des données et de recevoir des réponses.

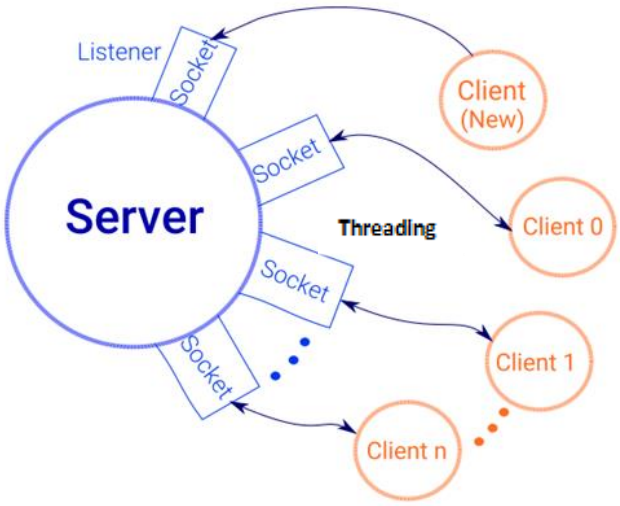
Abstraction Réseau : Ils abstraient la complexité de la communication réseau en fournissant des méthodes et des protocoles (comme TCP/IP ou UDP) qui gèrent la transmission de paquets de données de manière fiable ou selon le meilleur effort.

Threading :

Le threading en Python fait référence à la capacité d'exécuter simultanément plusieurs threads (unités plus petites d'un processus) dans le même espace de processus. Chaque thread fonctionne de manière indépendante mais partage des ressources communes telles que la mémoire et les descripteurs de fichiers.

Le threading permet l'exécution de plusieurs tâches ou opérations en parallèle. Cela signifie que différentes parties d'un programme peuvent s'exécuter de manière concurrente et indépendante les unes des autres, améliorant ainsi l'efficacité de l'exécution.

Les threads facilitent la programmation asynchrone, où plusieurs opérations peuvent progresser indépendamment, permettant au programme de continuer à s'exécuter même si un thread est bloqué ou en attente de ressources.



En utilisant des threads, un programme serveur multi-thread peut accepter une connexion d'un client, démarrer un thread pour cette communication, et continuer à écouter les requêtes des autres clients.

Logging

Le module logging en Python permet d'enregistrer des messages d'état ou de diagnostic pendant l'exécution d'une application. Il offre un moyen standardisé de capturer des informations sur l'exécution du programme, ce qui est particulièrement utile pour le débogage et le suivi des activités.

Code :

```
logging.basicConfig(filename='logs server.txt', level=logging.INFO, format='%(asctime)s - %(message)s')
```

Explication :

logging.basicConfig(...) : Configure le système de journalisation (logging) pour enregistrer les activités du serveur dans un fichier nommé 'logs_server.txt'. Le niveau de journalisation est défini sur INFO, ce qui signifie que seules les informations importantes sont enregistrées, telles que les connexions et les messages échangés.

- Rôle:
    - *Journalisation des Événements :* Le logging est utilisé pour enregistrer des événements importants, comme les connexions établies, les messages reçus, les erreurs détectées, etc. Cela permet aux développeurs de comprendre le comportement de l'application et de diagnostiquer les problèmes plus facilement.
    - *Niveaux de Gravité :* Il permet de spécifier différents niveaux de gravité pour les messages (comme DEBUG, INFO, WARNING, ERROR, CRITICAL), facilitant ainsi le filtrage et la gestion des informations en fonction de leur importance.

En résumé, les sockets et le threading sont essentiels en Python pour le réseau et la programmation concurrente, respectivement. Les sockets permettent la communication à travers les réseaux en utilisant divers protocoles, tandis que le threading facilite l'exécution simultanée des tâches au sein d'un même processus, améliorant ainsi les performances et la

réactivité dans des applications complexes. Comprendre ces concepts est crucial pour développer des applications réseau évolutives et réactives en Python.

C- Serveur TCP :

Le serveur TCP de TCP X représente le cœur du système, assurant la gestion efficace des connexions et des communications entre les clients. Voici ses principales fonctions :

- ***Écoute et Gestion des Connexions :*** Le serveur écoute en permanence sur un port spécifié (par exemple, **le port 5555**) pour les nouvelles connexions entrantes des clients. Il crée dynamiquement des threads pour chaque client connecté, permettant ainsi une gestion simultanée des requêtes sans bloquer d'autres connexions.

Code:

```
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(("0.0.0.0", 5555))
server.listen()

clients = {}
```

Explication de Code:

socket.socket(socket.AF_INET, socket.SOCK_STREAM) : Cette ligne crée un objet de socket TCP (socket.SOCK_STREAM) utilisant IPv4 (socket.AF_INET). Les sockets sont des points de terminaison de la communication réseau permettant à deux machines de communiquer entre elles.

server.bind(("0.0.0.0", 5555)) : Lie le serveur à une adresse IP (0.0.0.0 dans ce cas) et un numéro de port (5555). L'adresse 0.0.0.0 signifie que le serveur écoutera sur toutes les interfaces réseau disponibles sur la machine.

server.listen() : Met le serveur en mode d'écoute, attendant les connexions entrantes des clients. Le paramètre par défaut (backlog) spécifie le nombre maximal de connexions en attente que le serveur peut gérer à la fois.

D- Clients

Les clients de TCP X sont des applications légères destinées à fournir une interface conviviale pour les utilisateurs finaux. Voici leurs fonctionnalités principales :

- Connexion au Serveur : Les utilisateurs se connectent au serveur TCP en spécifiant son adresse IP et le port d'écoute. Une fois connectés, ils peuvent participer à des conversations et échanger des messages avec d'autres utilisateurs connectés.

Code :

```
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(("127.0.0.1", 5555))
```

Explication :

- ***socket.socket(socket.AF_INET, socket.SOCK_STREAM) :*** Crée un objet de socket client TCP.
- ***client.connect(("127.0.0.1", 5555)) :*** Établit une connexion vers le serveur TCP écoutant sur l'adresse IP '127.0.0.1' (localhost) et le port 5555. Cette étape permet au client de s'identifier auprès du serveur et d'initier des échanges de données.

D- Programme de Cryptographie CryptiX

CryptiX est le module de sécurité central de TCP X, chargé de protéger la confidentialité des échanges entre les utilisateurs. Voici comment fonctionne CryptiX :

- *Chiffrement des Messages :* Lorsqu'un utilisateur souhaite envoyer un message, CryptiX applique un chiffrement par décalage. Chaque caractère alphabétique du message est déplacé d'un nombre fixe de positions dans l'alphabet. Les caractères non alphabétiques restent inchangés.

Code :

```python
def encrypt(message, shift=3):
    encrypted_message = ""
    for char in message:
        if char.isalpha():
            shift_amount = shift % 26
            new_char = chr((ord(char.lower()) - ord('a') + shift_amount) % 26 + ord('a'))
            if char.isupper():
                new_char = new_char.upper()
            encrypted_message += new_char
        else:
            encrypted_message += char
    return encrypted_message
```

Explication:

- ***encrypt(message, shift=3) :*** Fonction qui chiffre le message donné (message) en utilisant un chiffrement par décalage avec un décalage spécifié (shift). Si aucun décalage n'est fourni, il utilise un décalage par défaut de 3.

Fonctionnement :

- **Boucle For :** Parcourt chaque caractère (char) dans le message.
- **char.isalpha() :** Vérifie si le caractère est alphabétique.
- **Calcul du Nouveau Caractère** : Applique le décalage au caractère alphabétique (char.lower()) en utilisant l'arithmétique modulo 26 pour assurer que le caractère reste dans l'alphabet.

- *Déchiffrement des Messages :* À réception, le message chiffré est déchiffré en utilisant le même décalage inverse. Cela permet aux destinataires autorisés de lire le message original sans altération.

Code :

```python
def decrypt(message, shift=3):
    return encrypt(message, -shift)
```

Explication :

decrypt(message, shift=3) : Fonction qui déchiffre le message donné (message) en utilisant le chiffrement par décalage inverse. Cela est réalisé en appelant la fonction encrypt avec un décalage négatif (-shift).

- *Intégration dans l'Interface Client :* CryptiX est intégré de manière transparente dans l'interface utilisateur des clients TCP X. Les utilisateurs peuvent activer le chiffrement en entrant le mot de passe requis, ce qui garantit que seuls les participants autorisés peuvent accéder au contenu des messages.

III- Implémentation

A. Environnement de Développement

Pour le développement de TCP X, un environnement de développement robuste et adapté aux besoins spécifiques a été mis en place. Cet environnement comprend plusieurs outils et technologies essentielles pour le développement, le test et le déploiement de l'application.

- Langage de Programmation: Python a été choisi pour sa simplicité, sa flexibilité et ses capacités avancées en matière de mise en réseau et de traitement multithreadé.

- Éditeur de Code: Un éditeur de code comme Visual Studio Code a été utilisé pour son support étendu des plugins Python, son interface utilisateur conviviale et ses fonctionnalités avancées d'édition et de débogage.

Dépendances et Bibliothèques

- Tkinter: Utilisé pour la création de l'interface utilisateur graphique (GUI) de TCP X, facilitant ainsi l'interaction utilisateur avec l'application.

- Logging : Intégré pour enregistrer les événements importants et les erreurs pendant l'exécution de l'application, ce qui est crucial pour le débogage et le suivi.

B-Détails de l’implémentation

L'implémentation de TCP X s'appuie sur plusieurs composants clés qui travaillent ensemble pour assurer le fonctionnement et la performance de l'application.

Architecture Client-Serveur

Le serveur utilise des sockets pour écouter les connexions entrantes des clients. Chaque nouvelle connexion est gérée dans un thread séparé à l'aide du module threading, permettant ainsi au serveur de gérer plusieurs clients simultanément de manière asynchrone.

Les clients se connectent au serveur via des sockets et peuvent échanger des messages cryptés à l'aide du programme de cryptographie CryptiX intégré. L'interface utilisateur des clients est développée avec Tkinter, offrant une expérience utilisateur intuitive et interactive.

Programme de Cryptographie CryptiX

CryptiX offre des fonctionnalités de cryptage et de décryptage basées sur des algorithmes simples de substitution. Il utilise Tkinter pour fournir une interface conviviale permettant aux utilisateurs de saisir des messages, de spécifier un mot de passe pour le cryptage/décryptage, et d'afficher les résultats chiffrés/déchiffrés.

IV- Test et Validation

A. Méthodologie de Test

La méthodologie de test adoptée pour TCP X visait à garantir la fiabilité, la sécurité et les performances de l'application tout au long du cycle de développement. Voici les principaux aspects de la méthodologie de test utilisée :

Test Unitaires

Des tests unitaires ont été utilisés pour vérifier le bon fonctionnement des fonctions et des modules individuels de l'application, tels que les fonctions de cryptage/décryptage, les fonctions de gestion des sockets, etc.

Test d'Intégration

Les composants de l'application, y compris le serveur TCP, les clients et le programme de cryptographie CryptiX, ont été intégrés et testés pour vérifier leur interopérabilité et leur capacité à fonctionner ensemble sans erreur.

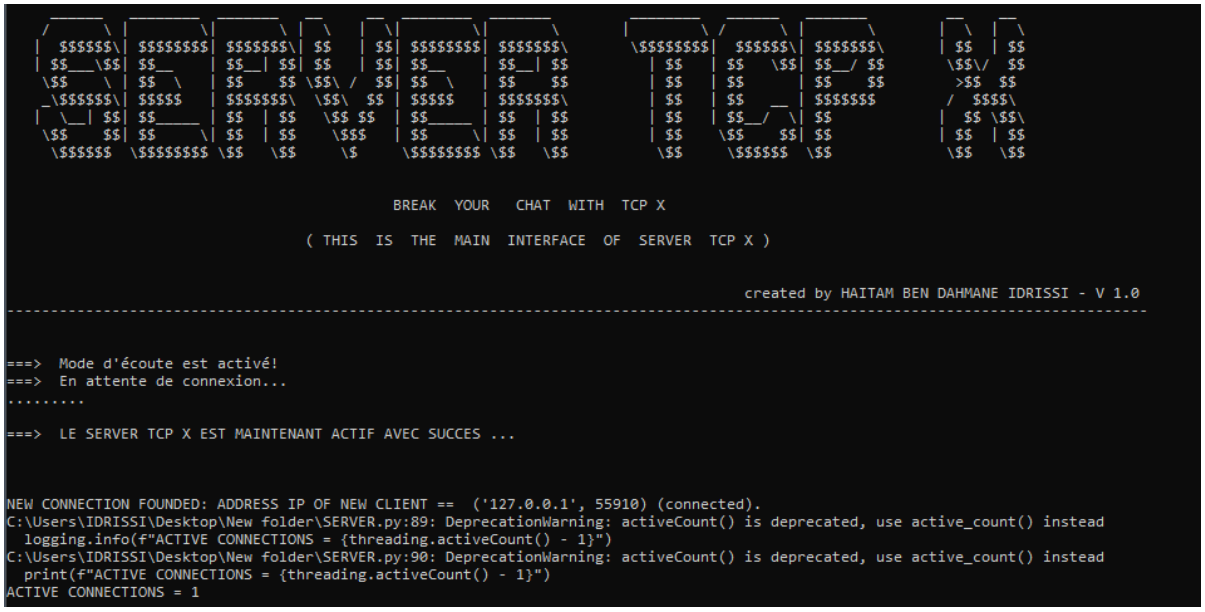
B. Cas de Test

Les cas de test ont été conçus pour couvrir divers aspects fonctionnels et non fonctionnels de TCP X, en se concentrant sur les scénarios critiques et les fonctionnalités clés de l'application :

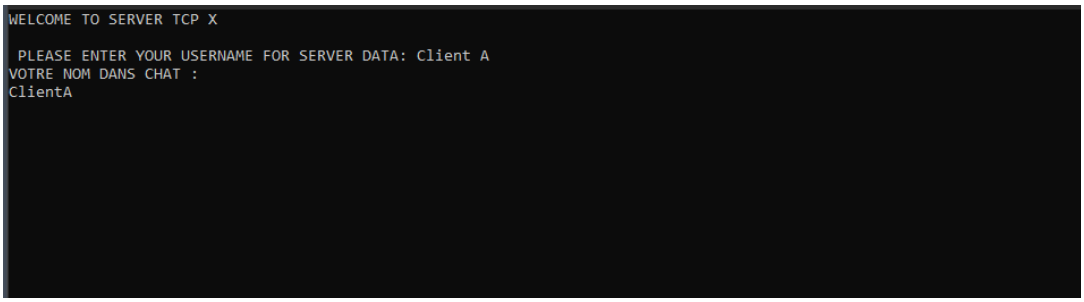
1.Test de Connexion :

- Vérification de la connexion réussie des clients au serveur TCP.

Résultat Serveur:

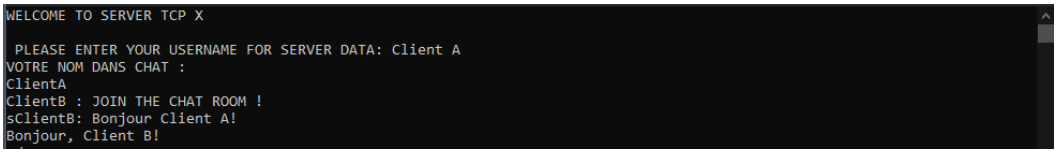


Résultat Client A:

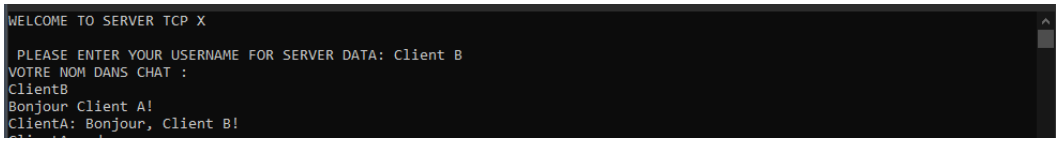


- Validation de l'établissement correct des sockets et du transfert de données.

Résultat de Communication entre Client A et B :



Client A



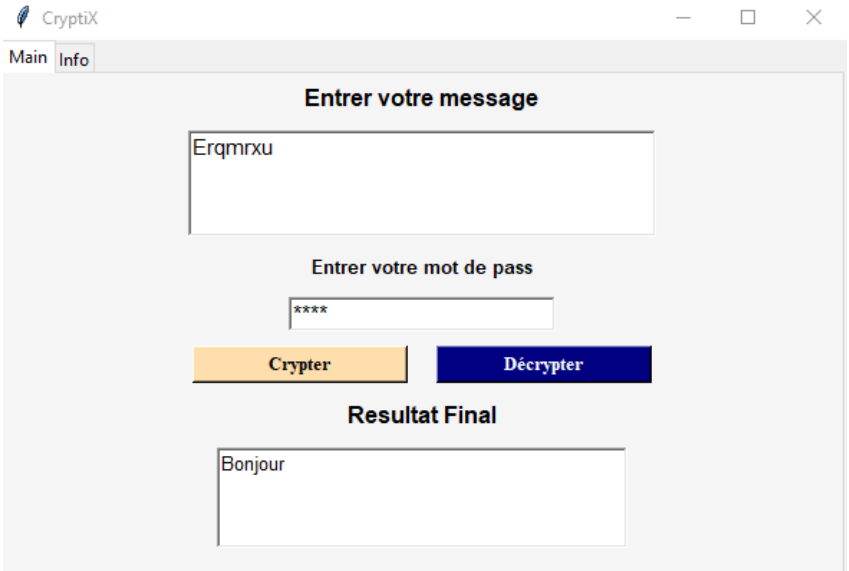
Client B

1.Test de Cryptographie :

- Vérification du cryptage et du décryptage corrects des messages à l'aide de CryptiX.
- Validation de l'exactitude des résultats chiffrés et déchiffrés.

Résultat du cryptage :

Résultat du décryptage :



VI- Sécurité et Attaques

A. Analyse de la Sécurité de TCP X

TCP X est évalué sous l'angle de la sécurité pour assurer que toutes les données échangées entre le serveur et les clients sont protégées contre les accès non autorisés et les manipulations malveillantes. Voici les principaux aspects analysés :

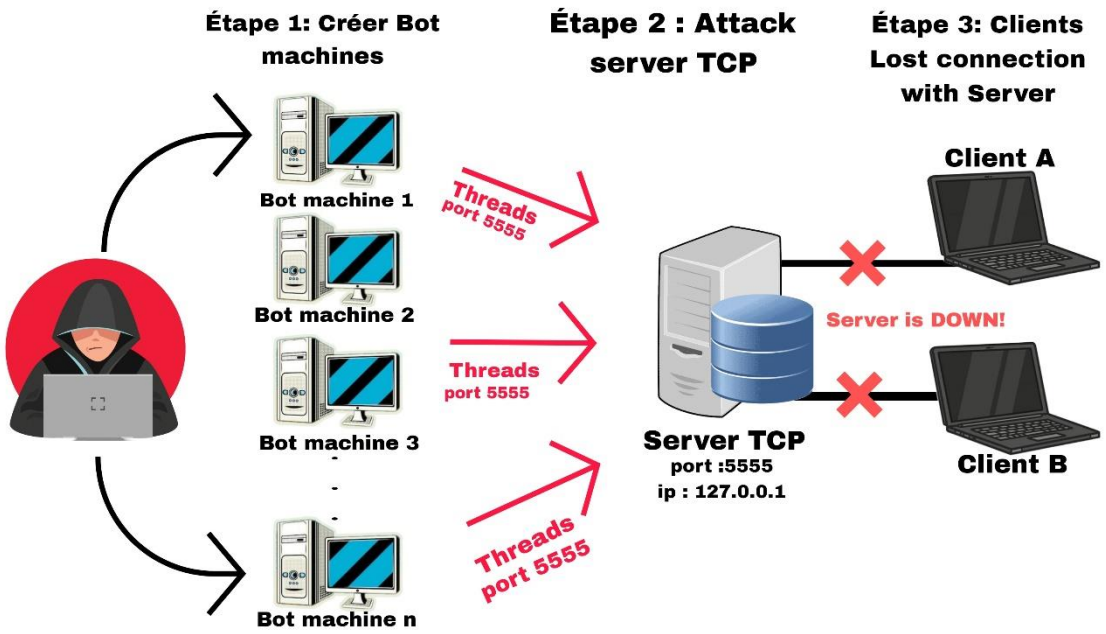
Cryptographie Robuste : Utilisation du programme CryptiX pour chiffrer les messages échangés entre les utilisateurs. Ce cryptage robuste garantit que même si les données sont interceptées, elles restent illisibles sans la clé de déchiffrement appropriée.

Gestion des Connexions : Les sockets sont utilisés pour établir et maintenir des connexions réseau sécurisées entre le serveur et les clients. Le module threading permet de gérer ces connexions de manière à éviter les blocages et à maintenir la réactivité du serveur face à de multiples utilisateurs.

Validation et Authentification : Mécanismes intégrés pour authentifier les clients lors de leur connexion au serveur. Cela inclut souvent l'utilisation de noms d'utilisateur et de mots de passe sécurisés pour vérifier l'identité des utilisateurs autorisés.

B. Simulation d’Attaque par Déni de Service (DoS)

L'objectif de cette simulation est de tester la résilience de TCP X face à une attaque par déni de service (DoS), où un attaquant tente de surcharger le serveur avec un grand nombre de requêtes de connexion simultanées.



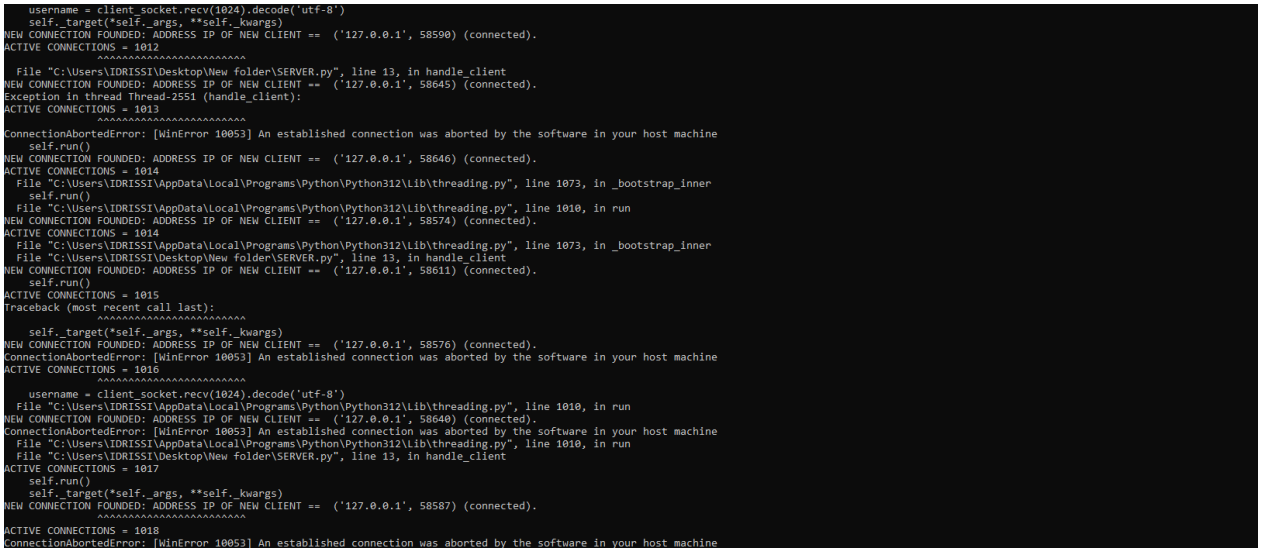
Le serveur TCP est configuré pour écouter sur **le port 5555** et accepte plusieurs connexions simultanées.

```
server.bind(("0.0.0.0", 5555))
server.listen()

clients = {}

print("Mode d'écoute de Serveur est Activé\n Serveur Active...")
```

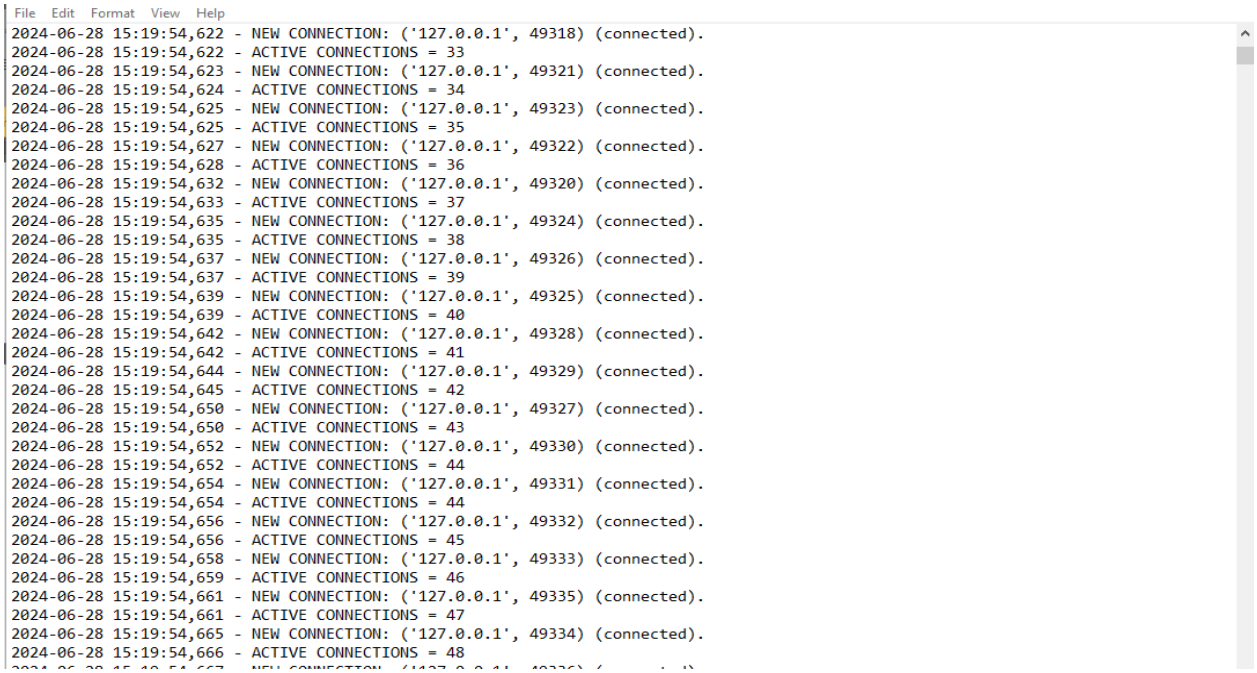
Un script d'une attaque DoS. Ce script ouvre continuellement des connexions au serveur et envoie des messages (contient port de serveur 5555 et l'adresse local) pour saturer les ressources du serveur.

```
target_ip = '127.0.0.1'
target_port = 5555

threads = []
```

Le Serveur en temp d'attaque d'un DoS (avec Logs de serveur) :



```
    username = client_socket.recv(1024).decode('utf-8')
    self._target(*self._args, **self._kwargs)
NEW CONNECTION FOUNDED: ADDRESS IP OF NEW CLIENT ==  ('127.0.0.1', 58590) (connected).
ACTIVE CONNECTIONS = 1012
              ^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\IDRISSI\Desktop\New folder\SERVER.py", line 13, in handle_client
NEW CONNECTION FOUNDED: ADDRESS IP OF NEW CLIENT ==  ('127.0.0.1', 58645) (connected).
Exception in thread Thread-2551 (handle_client):
ACTIVE CONNECTIONS = 1013
              ^^^^^^^^^^^^^^^^^^^^^^^^
ConnectionAbortedError: [WinError 10053] An established connection was aborted by the software in your host machine
    self.run()
NEW CONNECTION FOUNDED: ADDRESS IP OF NEW CLIENT ==  ('127.0.0.1', 58646) (connected).
ACTIVE CONNECTIONS = 1014
  File "C:\Users\IDRISSI\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1073, in _bootstrap_inner
    self.run()
  File "C:\Users\IDRISSI\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1010, in run
NEW CONNECTION FOUNDED: ADDRESS IP OF NEW CLIENT ==  ('127.0.0.1', 58574) (connected).
ACTIVE CONNECTIONS = 1014
  File "C:\Users\IDRISSI\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1073, in _bootstrap_inner
  File "C:\Users\IDRISSI\Desktop\New folder\SERVER.py", line 13, in handle_client
NEW CONNECTION FOUNDED: ADDRESS IP OF NEW CLIENT ==  ('127.0.0.1', 58611) (connected).
    self.run()
ACTIVE CONNECTIONS = 1015
Traceback (most recent call last):
              ^^^^^^^^^^^^^^^^^^^^^^^^
    self._target(*self._args, **self._kwargs)
NEW CONNECTION FOUNDED: ADDRESS IP OF NEW CLIENT ==  ('127.0.0.1', 58576) (connected).
ConnectionAbortedError: [WinError 10053] An established connection was aborted by the software in your host machine
ACTIVE CONNECTIONS = 1016
              ^^^^^^^^^^^^^^^^^^^^^^^^
    username = client_socket.recv(1024).decode('utf-8')
  File "C:\Users\IDRISSI\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1010, in run
NEW CONNECTION FOUNDED: ADDRESS IP OF NEW CLIENT ==  ('127.0.0.1', 58640) (connected).
ConnectionAbortedError: [WinError 10053] An established connection was aborted by the software in your host machine
  File "C:\Users\IDRISSI\AppData\Local\Programs\Python\Python312\Lib\threading.py", line 1010, in run
  File "C:\Users\IDRISSI\Desktop\New folder\SERVER.py", line 13, in handle_client
ACTIVE CONNECTIONS = 1017
    self.run()
    self._target(*self._args, **self._kwargs)
NEW CONNECTION FOUNDED: ADDRESS IP OF NEW CLIENT ==  ('127.0.0.1', 58587) (connected).
              ^^^^^^^^^^^^^^^^^^^^^^^^
ACTIVE CONNECTIONS = 1018
ConnectionAbortedError: [WinError 10053] An established connection was aborted by the software in your host machine
```



File Edit Format View Help

```
2024-06-28 15:19:54,622 - NEW CONNECTION: ('127.0.0.1', 49318) (connected).
2024-06-28 15:19:54,622 - ACTIVE CONNECTIONS = 33
2024-06-28 15:19:54,623 - NEW CONNECTION: ('127.0.0.1', 49321) (connected).
2024-06-28 15:19:54,624 - ACTIVE CONNECTIONS = 34
2024-06-28 15:19:54,625 - NEW CONNECTION: ('127.0.0.1', 49323) (connected).
2024-06-28 15:19:54,625 - ACTIVE CONNECTIONS = 35
2024-06-28 15:19:54,627 - NEW CONNECTION: ('127.0.0.1', 49322) (connected).
2024-06-28 15:19:54,628 - ACTIVE CONNECTIONS = 36
2024-06-28 15:19:54,632 - NEW CONNECTION: ('127.0.0.1', 49320) (connected).
2024-06-28 15:19:54,633 - ACTIVE CONNECTIONS = 37
2024-06-28 15:19:54,635 - NEW CONNECTION: ('127.0.0.1', 49324) (connected).
2024-06-28 15:19:54,635 - ACTIVE CONNECTIONS = 38
2024-06-28 15:19:54,637 - NEW CONNECTION: ('127.0.0.1', 49326) (connected).
2024-06-28 15:19:54,637 - ACTIVE CONNECTIONS = 39
2024-06-28 15:19:54,639 - NEW CONNECTION: ('127.0.0.1', 49325) (connected).
2024-06-28 15:19:54,639 - ACTIVE CONNECTIONS = 40
2024-06-28 15:19:54,642 - NEW CONNECTION: ('127.0.0.1', 49328) (connected).
2024-06-28 15:19:54,642 - ACTIVE CONNECTIONS = 41
2024-06-28 15:19:54,644 - NEW CONNECTION: ('127.0.0.1', 49329) (connected).
2024-06-28 15:19:54,645 - ACTIVE CONNECTIONS = 42
2024-06-28 15:19:54,650 - NEW CONNECTION: ('127.0.0.1', 49327) (connected).
2024-06-28 15:19:54,650 - ACTIVE CONNECTIONS = 43
2024-06-28 15:19:54,652 - NEW CONNECTION: ('127.0.0.1', 49330) (connected).
2024-06-28 15:19:54,652 - ACTIVE CONNECTIONS = 44
2024-06-28 15:19:54,654 - NEW CONNECTION: ('127.0.0.1', 49331) (connected).
2024-06-28 15:19:54,654 - ACTIVE CONNECTIONS = 44
2024-06-28 15:19:54,656 - NEW CONNECTION: ('127.0.0.1', 49332) (connected).
2024-06-28 15:19:54,656 - ACTIVE CONNECTIONS = 45
2024-06-28 15:19:54,658 - NEW CONNECTION: ('127.0.0.1', 49333) (connected).
2024-06-28 15:19:54,659 - ACTIVE CONNECTIONS = 46
2024-06-28 15:19:54,661 - NEW CONNECTION: ('127.0.0.1', 49335) (connected).
2024-06-28 15:19:54,661 - ACTIVE CONNECTIONS = 47
2024-06-28 15:19:54,665 - NEW CONNECTION: ('127.0.0.1', 49334) (connected).
2024-06-28 15:19:54,666 - ACTIVE CONNECTIONS = 48
```

Dans la partie D de la section VI sur la sécurité et les attaques, nous abordons l'utilisation du Firewall Linux (ufw) comme mesure de protection contre les attaques, en particulier les attaques par déni de service (DoS).

D. Utilisation du Firewall Linux (ufw) pour arrêter l'attaque

Le Firewall Linux (ufw) est un outil essentiel pour sécuriser un système en contrôlant le trafic réseau entrant et sortant. Voici comment il est utilisé dans TCP X pour contrer les attaques.

Concept et Fonctionnement du Firewall Linux (ufw)

ufw utilise des règles configurées pour spécifier quel type de trafic est autorisé ou bloqué. Ces règles peuvent être basées sur des adresses IP, des ports, des protocoles, etc. En réponse à une attaque DoS, TCP X peut configurer ufw pour bloquer les adresses IP sources suspectes qui génèrent un volume anormal de requêtes de connexion. Cela réduit la charge sur le serveur en refusant le trafic provenant de ces adresses.

ufw peut être configuré pour détecter les motifs de trafic associés aux attaques DoS, comme une augmentation soudaine du nombre de connexions ou de requêtes malveillantes.

Une fois une attaque détectée, ufw peut automatiquement bloquer le trafic en utilisant les règles préétablies, minimisant ainsi l'impact de l'attaque sur les performances du serveur TCP.

Les activités du firewall ufw sont enregistrées dans des logs système. Cela permet aux administrateurs de surveiller et d'analyser les tentatives d'attaques, facilitant la détection précoce et la réponse rapide.

Les logs de ufw permettent également aux équipes de sécurité de réviser et d'ajuster les règles de filtrage en fonction des nouvelles menaces ou des modèles de trafic anormaux observés.

Avantages de l'utilisation de ufw dans TCP X

Il offre une protection robuste contre un large éventail de menaces réseau, y compris les attaques DoS, tout en minimisant l'impact sur les performances du système. En configurant correctement ufw, TCP X peut adapter sa défense en fonction des besoins spécifiques de sécurité et des conditions du réseau.

En intégrant ufw comme mesure de sécurité dans TCP X, l'application renforce sa résilience face aux attaques réseau tout en assurant la disponibilité continue des services pour les utilisateurs légitimes.

Tester ufw contre attaque Dos

- Il n y a aucun attaque afficher

VIII- Avantages et Inconvénients de TCP X :

TCP X présente à la fois des avantages et des inconvénients significatifs qui influencent son déploiement et son utilisation dans les environnements réseau.

Du côté des inconvénients, la complexité de configuration constitue un défi majeur pour les administrateurs système inexpérimentés. La mise en place initiale de TCP X ainsi que la configuration du firewall ufw exigent une expertise spécifique pour optimiser les règles de

sécurité, ce qui peut retarder le déploiement et nécessiter des compétences avancées en gestion réseau.

En outre, TCP X est tributaire de ressources système substantielles en raison de l'utilisation intensive des threads pour la gestion des connexions. Cette exigence peut restreindre son implémentation sur des infrastructures avec des capacités limitées, limitant ainsi sa portabilité et sa scalabilité dans certains contextes.

En termes de sécurité, bien que TCP X intègre des mesures telles que CryptiX et ufw pour protéger les données contre les attaques, il reste vulnérable aux attaques réseau, notamment les attaques par déni de service (DoS), si les mesures de sécurité ne sont pas correctement configurées ou maintenues à jour.

Malgré ces défis, TCP X présente également des avantages significatifs qui le rendent attrayant pour la gestion des communications réseau. Tout d'abord, en utilisant le protocole TCP (Transmission Control Protocol), TCP X assure une livraison fiable et sans perte des données, essentielle pour les applications nécessitant une transmission sécurisée et efficace.

De plus, l'intégration de CryptiX pour le chiffrement des données renforce la sécurité des communications en garantissant la confidentialité et l'intégrité des informations échangées. Cette fonctionnalité est cruciale dans les environnements où la protection des données sensibles est une priorité.

En termes de performance, TCP X utilise efficacement des sockets et des threads pour gérer plusieurs connexions simultanées, optimisant ainsi les performances du serveur et offrant une évolutivité robuste pour répondre aux besoins croissants des utilisateurs. Cette capacité à maintenir des communications fluides et réactives contribue à une expérience utilisateur améliorée, même dans des environnements à forte charge.

Enfin, avec une interface utilisateur développée en tkinter, TCP X offre une expérience conviviale, facilitant l'accès aux fonctionnalités de chat et de cryptage même pour les utilisateurs non techniques. Cette convivialité renforce l'adoption et l'utilisation de TCP X dans divers contextes d'entreprise et de communication.

En conclusion, bien que TCP X présente des défis de déploiement et de sécurité, ses avantages en termes de fiabilité, sécurité renforcée, performance et convivialité de l'interface en font une solution attrayante pour ceux qui recherchent une gestion avancée des communications réseau.

V- Conclusion et Perspectives

A. Résumé des Accomplissements

TCP X représente une avancée significative dans la gestion sécurisée des communications réseau. En intégrant des fonctionnalités robustes telles que le cryptage avec CryptiX et une gestion efficace des connexions via l'utilisation de sockets et de threads, ce projet s'affirme comme une plateforme technologique de premier plan. Il se distingue par sa capacité à fournir une infrastructure fiable et sécurisée pour la transmission de données sensibles, garantissant la confidentialité et l'intégrité des informations échangées.

Le développement de TCP X a permis de répondre à plusieurs défis critiques dans le domaine des communications réseau, notamment en offrant une solution qui combine robustesse et flexibilité. La mise en œuvre du cryptage avec CryptiX joue un rôle crucial en assurant que les données transmises sont protégées contre les interceptions non autorisées, renforçant ainsi la confiance des utilisateurs dans la sécurité des échanges.

Par ailleurs, la gestion avancée des connexions à travers l'utilisation optimisée de sockets et de threads améliore la performance globale du système, permettant à TCP X de gérer efficacement de multiples connexions simultanées sans compromettre la qualité du service. Cette capacité à maintenir des communications fluides et réactives est essentielle dans des environnements où la disponibilité et la fiabilité sont des exigences critiques.

En outre, l'adoption d'une approche axée sur la sécurité tout au long du développement de TCP X a permis de minimiser les risques liés aux attaques et aux intrusions. Bien que toute technologie soit sujette à des vulnérabilités potentielles, TCP X a démontré sa capacité à mettre en place des mesures de sécurité robustes, notamment à travers l'intégration continue de mises à jour et de correctifs pour contrer les menaces émergentes.

En conclusion, TCP X se positionne comme une solution innovante et fiable pour la gestion des communications réseau sécurisées. Ses accomplissements dans l'intégration du cryptage avancé, de la gestion efficace des connexions et de la sécurité renforcée en font un choix pertinent pour les organisations cherchant à garantir la confidentialité et l'intégrité de leurs

données. Avec un engagement constant envers l'amélioration et l'innovation, TCP X offre des perspectives prometteuses pour l'avenir de la sécurité des communications en réseau.

B. Améliorations Futures et Perspectives

1. Une réduction de la dépendance aux ressources système et une optimisation plus poussée de l'utilisation des threads sont essentielles pour améliorer l'efficacité et la réactivité globale du serveur. Cela permettrait à TCP X de gérer efficacement une charge accrue sans compromettre la qualité du service.

2. L'intégration de nouvelles fonctionnalités avancées telles que la gestion de groupes de discussion, le partage de fichiers sécurisé et une gestion plus fine des utilisateurs enrichirait considérablement l'expérience utilisateur. Ces ajouts pourraient répondre à une demande croissante pour des solutions de communication réseau plus intégrées et multifonctionnelles.

3. La sécurité demeure une priorité absolue. Continuer à mettre à jour régulièrement les mécanismes de sécurité avec les dernières technologies et pratiques aiderait à prévenir de nouvelles menaces et à renforcer la résilience contre les attaques potentielles. Une vigilance constante et une adaptation aux évolutions des menaces sont cruciales pour maintenir la confiance des utilisateurs.

4. Développer une interface utilisateur plus intuitive et conviviale, basée sur les retours d'expérience des utilisateurs, pourrait significativement faciliter l'adoption et l'utilisation de TCP X dans divers environnements. Une interface bien conçue contribuerait à une expérience utilisateur améliorée et à une utilisation plus efficace des fonctionnalités avancées de TCP X.

Références :

Les Références sont disponibles sur le lien suivant,vous pouvez télécharger les fichiers liés au rapport :

https://github.com/Haitamidrissi/TCP-X.git