HEM

Simulation Parc Informatique

Rapport du Projet

Préparé par:
AFIF Hamza – AOUNI Yassine

Encadré par :
Yassine HERNAFI, Dr
Hamza RAHHALI, Dr
Mme Zineb LANBOURI, Dr

2023/2024

Table des matières :

Introduction au Projet de Simulation de Réseau :

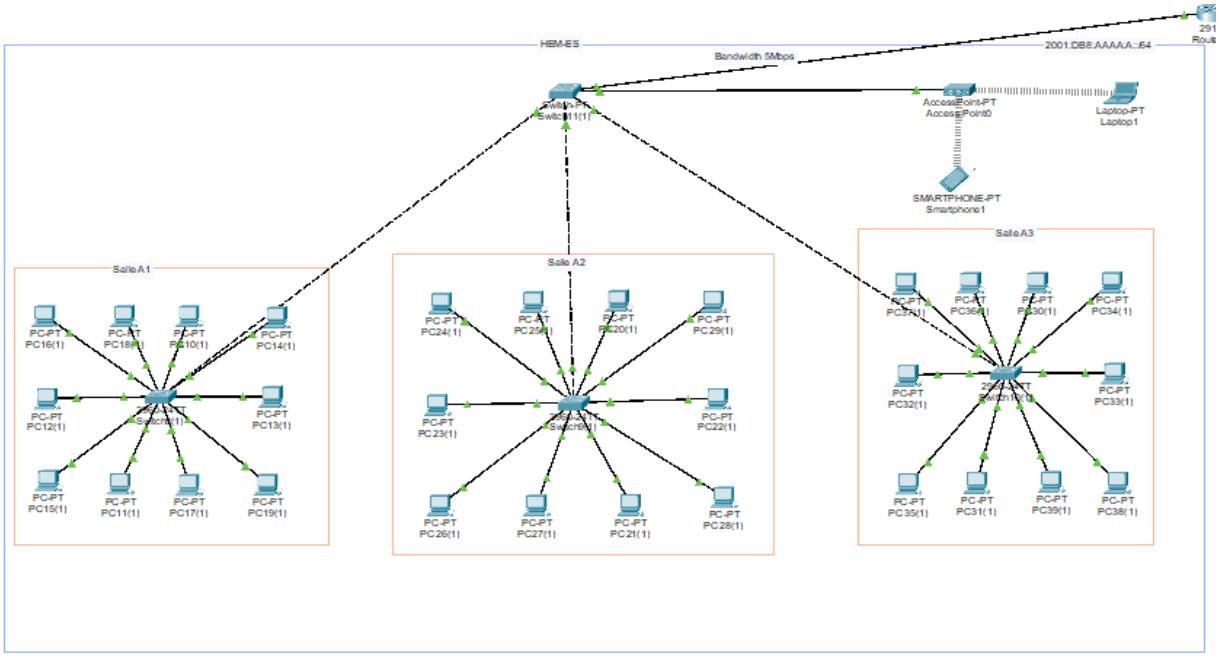
Notre simulation de réseau, étendue à un parc informatique virtuel composé des bâtiments ES, BS et DG, gagne en sophistication avec l'adoption de GNS3, un outil de simulation réseau dynamique. Cette avancée permet l'intégration d'une machine virtuelle Ubuntu, créant un environnement de test encore plus réaliste et interactif. En complément, l'utilisation du langage de programmation Python, avec sa puissante bibliothèque Scapy, nous autorise à effectuer du sniffing de paquets réseau. Ce procédé nous permet de capturer et d'analyser le trafic en temps réel, offrant une vision approfondie de la gestion des données et des opérations de sécurité au sein de notre architecture réseau. Ensemble, ces technologies modernes élèvent notre simulation à un niveau supérieur, en répliquant fidèlement les complexités et les interactions d'un réseau d'entreprise en temps réel.

La Topologie Réseau :

Cette section offre un aperçu visuel de la disposition de notre réseau simulé. À travers des captures d'écran, nous présentons la structure de chaque LAN, mettant en lumière les dispositifs et les interfaces spécifiques.

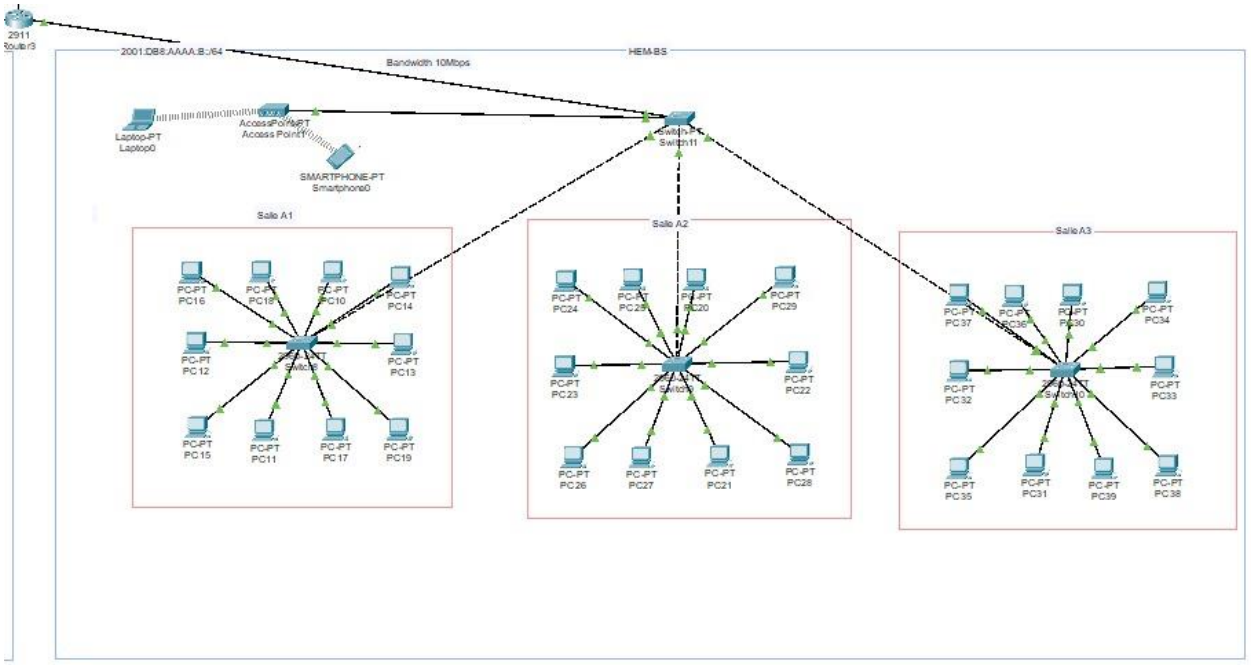
Topologie du LAN dans le Bâtiment ES :

Le premier LAN, situé dans le bâtiment ES, présente une topologie réseau soigneusement conçue pour assurer une connectivité efficace. Ce LAN démarre par une connexion directe au routeur via l'interface GigabitEthernet0/1, établissant ainsi le lien principal vers le réseau global. Le routeur est ensuite connecté à un commutateur, assurant la distribution du trafic au sein du LAN local. Un point d'accès est relié au commutateur, permettant la connectivité sans fil au sein du bâtiment. Trois salles, chacune équipée d'un commutateur dédié, sont connectées au commutateur principal. Ces commutateurs locaux, à leur tour, facilitent la connexion de 10 ordinateurs par salle via des interfaces Fast Ethernet, offrant ainsi des connexions fiables et rapides.



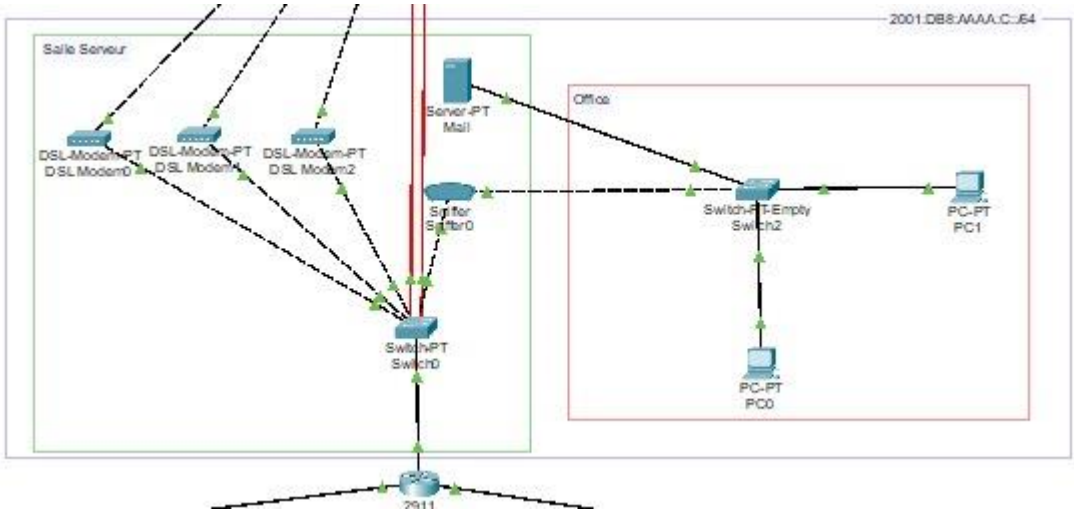
Topologie du Deuxième LAN dans le Bâtiment BS :

Le deuxième LAN, situé dans le bâtiment BS, suit la même configuration que le premier LAN pour assurer une connexion solide. Il est relié au routeur par l'interface GigabitEthernet0/2, établissant ainsi une connexion principale avec le réseau. Le routeur est connecté à un commutateur qui distribue le trafic dans le LAN local. Un point d'accès gère la connectivité sans fil, tandis que trois salles, chacune équipée d'un commutateur dédié, sont connectées au commutateur principal. Ces commutateurs locaux permettent la connexion de 10 ordinateurs par salle via des interfaces Fast Ethernet, offrant une expérience réseau fiable et rapide au sein du deuxième LAN du bâtiment BS.



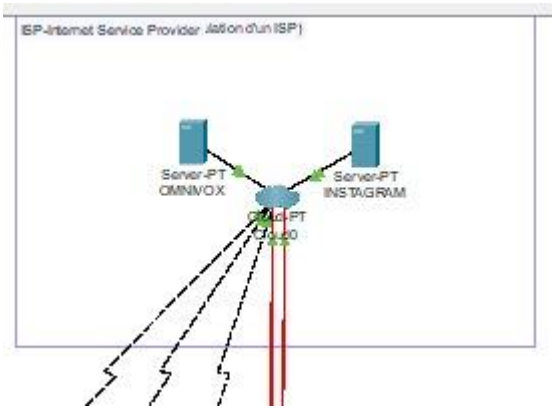
Topologie du LAN dans le Bâtiment DG :

Le bâtiment DG est équipé d'une infrastructure réseau moderne et robuste, conçue pour assurer une connectivité fiable et sécurisée pour toutes les opérations informatiques. Au cœur du réseau se trouve la salle serveur, qui abrite les serveurs de mail et d'autres équipements critiques, garantissant la continuité des services et la gestion des données. Les modems DSL assurent une connexion Internet stable, essentielle pour les communications externes et l'accès à distance. Les commutateurs réseau distribuent le trafic efficacement, facilitant la connexion des différents périphériques et postes de travail, représentés par les ordinateurs personnels. Cette configuration du réseau est stratégiquement conçue pour optimiser les performances et la gestion du trafic, tout en offrant une plateforme évolutive pour l'avenir.



Topologie de la connectivité ISP (Internet Service Provider) :

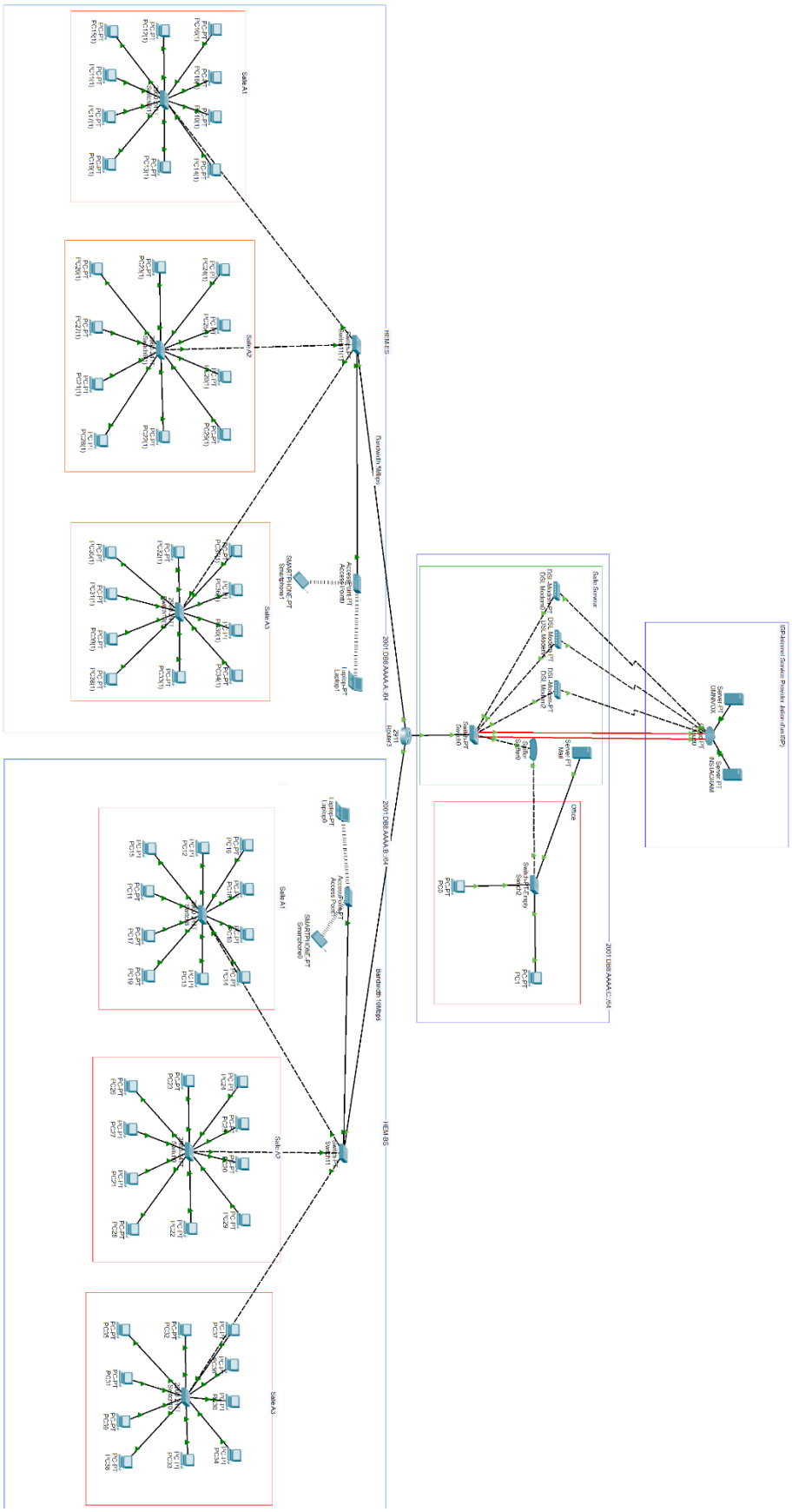
Dans la section du réseau dédiée à la connectivité ISP, nous observons une architecture stratégique qui facilite la communication entre le bâtiment DG et les autres bâtiments ES et BS. Les serveurs, étiquetés OMMNOX et INSTAGRUM, semblent être des points centraux pour la gestion du trafic réseau. Ils sont connectés à un fournisseur de services Internet (ISP), ce qui indique que ces serveurs pourraient être dédiés à l'interconnexion avec des réseaux externes, fournissant ainsi un accès Internet à l'ensemble de l'infrastructure. La présence de liaisons redondantes entre les serveurs et le point d'accès ISP souligne un design orienté vers la haute disponibilité et la résilience du réseau, minimisant ainsi les risques de coupure de connexion pour les bâtiments DG, ES et BS. Cette configuration assure que le flux de données est non seulement constant mais aussi sécurisé, ce qui est primordial pour maintenir l'efficacité des opérations et la fiabilité des communications dans un environnement d'entreprise.



Topologie complète du réseau :

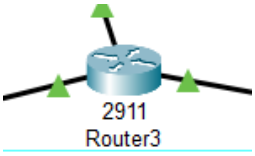
La topologie réseau complète présentée illustre un système complexe et interconnecté, conçu pour desservir une organisation étendue. L'architecture se déploie à partir d'un noyau central, probablement situé dans le bâtiment DG, et s'étend vers des sous-réseaux organisés stratégiquement dans d'autres bâtiments, identifiés comme ES et BS. Chaque sous-réseau comporte des commutateurs qui facilitent la connexion de multiples ordinateurs personnels, indiquant des zones de travail denses et fonctionnellement diversifiées.

L'ISP au sommet de la topologie sert de point d'accès principal pour l'Internet, avec des serveurs dédiés qui gèrent le trafic entrant et sortant, assurant la connectivité externe. Ces serveurs peuvent également fournir des services internes, tels que la messagerie électronique, comme indiqué par leurs étiquettes.


Salle A1
Salle A2
Salle A3
Salle A4
Salle A5
Salle Serveur
Office
ISP-Internet Service Provider (Fournisseur d'accès Internet)
HEM 65
HEM 85
2001:DB8:AAAA:A::/64
2001:DB8:AAAA:B::/64
2001:DB8:AAAA:C::/64
Switch-PT
Switch-PT-Empty
Server-PT
Server-PT Mail
Server-PT DHCPv6
Server-PT DNS
Server-PT WEB
Router-PT
Router-PT-Empty
Router-PT-Empty1
Router-PT-Empty2
Router-PT-Empty3
Router-PT-Empty4
Router-PT-Empty5
Router-PT-Empty6
Router-PT-Empty7
Router-PT-Empty8
Router-PT-Empty9
Router-PT-Empty10
Router-PT-Empty11
Router-PT-Empty12
Router-PT-Empty13
Router-PT-Empty14
Router-PT-Empty15
Router-PT-Empty16
Router-PT-Empty17
Router-PT-Empty18
Router-PT-Empty19
Router-PT-Empty20
Router-PT-Empty21
Router-PT-Empty22
Router-PT-Empty23
Router-PT-Empty24
Router-PT-Empty25
Router-PT-Empty26
Router-PT-Empty27
Router-PT-Empty28
Router-PT-Empty29
Router-PT-Empty30
Router-PT-Empty31
Router-PT-Empty32
Router-PT-Empty33
Router-PT-Empty34
Router-PT-Empty35
Router-PT-Empty36
Router-PT-Empty37
Router-PT-Empty38
Router-PT-Empty39
Router-PT-Empty40

La configuration Réseau :

Configuration du routeur :



Pour configurer le réseau IPv6 au sein de l'organisation et assurer une distribution automatique des adresses aux hôtes dans les bâtiments ES, BS et DG, nous avons mis en place un protocole DHCPv6 sur le routeur qui relie ces bâtiments. Nous avons commencé par activer le routage unicast IPv6 sur le routeur avec la commande `ipv6 unicast-routing`. Ceci est essentiel pour permettre au routeur de gérer le trafic IPv6.

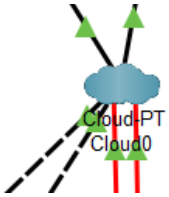
Ensuite, dans la configuration du routeur, nous avons créé des pools DHCPv6 pour chaque interface réseau. Par exemple, pour l'interface gig0/1 qui relie le bâtiment ES, nous avons utilisé la commande `ipv6 dhcp pool ES_POOL` suivie de `address prefix 2001:DB8:AAAA:A::/64` pour définir le préfixe d'adresse. Nous avons aussi spécifié un serveur DNS `dns-server 2001:DB8:AAAA:C:2E0:F7FF:FE1C:A95` et un nom de domaine à l'aide des commandes `dns-server OMNIVOX.COM` et `domain-name <omnivox.com>`, respectivement.

Pour chaque interface, nous avons attribué une adresse link-local en utilisant la commande `ipv6 address FE80::1 link-local`. Puis, nous avons configuré l'adresse unicast générale, par exemple `ipv6 address 2001:DB8:AAAA:A::1/64` pour l'interface gig0/1. Après cela, nous avons associé le serveur DHCP à l'interface avec la commande `ipv6 dhcp server ES_POOL`.

Pour informer les hôtes qu'ils doivent utiliser ces adresses pour la configuration automatique, nous avons activé le drapeau de configuration gérée sur l'interface avec `ipv6 nd managed-config-flag`. Enfin, pour s'assurer que toutes les configurations sont enregistrées dans la mémoire non volatile du routeur, nous avons exécuté la commande `wr` pour écrire la configuration.

Ainsi, chaque bâtiment se voit attribuer une plage d'adresses unique et peut communiquer de manière transparente avec le routeur central dans la salle DG et, par extension, avec l'ensemble du réseau de l'organisation.

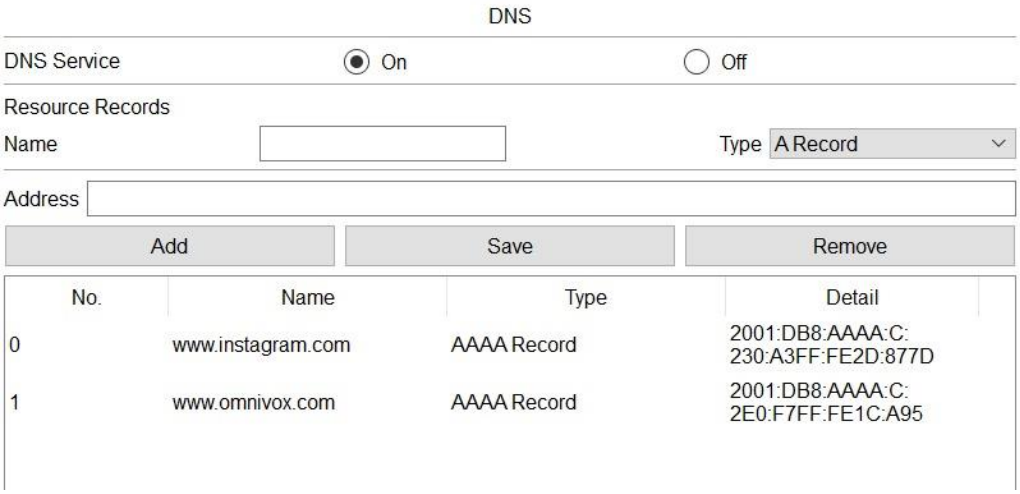
Configuration du cloud :



Avant de présenter la configuration détaillée des interfaces, il est important de noter que chaque interface du cloud a été méticuleusement configurée pour correspondre au port modem approprié, assurant ainsi une intégration transparente et une simulation réseau précise. Voici la configuration des interfaces :

Configuration du serveur DNS :

Dans le cadre de notre simulation réseau visant à reproduire un accès à Internet réaliste, nous avons mis en place un service DNS configuré avec des enregistrements AAAA. Ces enregistrements permettent la résolution de noms de domaine en adresses IPv6. Nous avons spécifiquement créé deux enregistrements : l'un pour www.instagram.com et l'autre pour www.omnivox.com, associés respectivement aux adresses IPv6 2001:DB8:AAAA:C:230:A3FF:FE2D:877D et 2001:DB8:AAAA:C:2E0:F7FF:FE1C:A95. Cette configuration est essentielle pour simuler l'accès aux services Internet par nom de domaine, une étape clé pour fournir une expérience réseau complète et fonctionnelle.

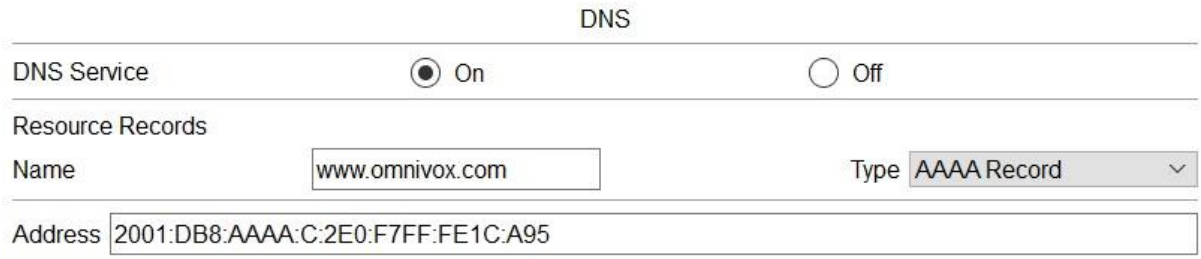


Configuration de la bande passante :

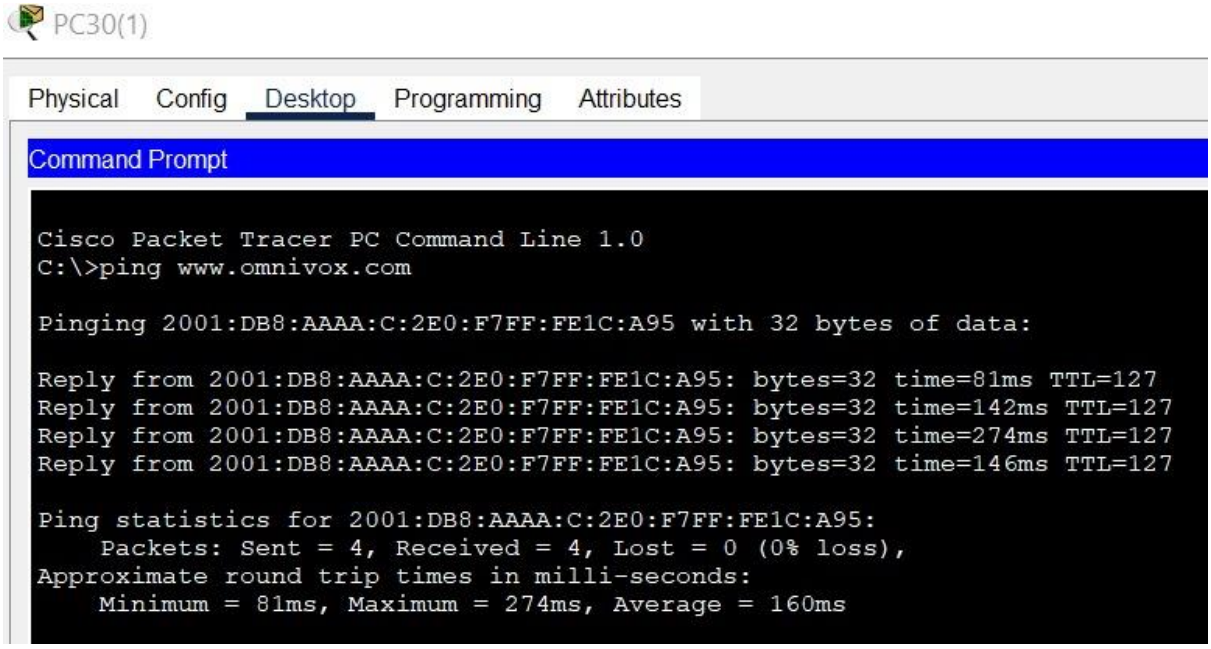
Pour réguler la bande passante dans notre réseau simulé, nous avons appliqué des politiques de limitation de débit sur les interfaces du routeur desservant les bâtiments ES et BS. Sur l'interface connectée au bâtiment ES, nous avons fixé une limite de 5 Mbps, tandis que pour le bâtiment BS, la limite est fixée à 10 Mbps. La commande de configuration sur le routeur pour établir cette limitation est ***rate-limit input 5000000 1500 2000 conform-action transmit exceed-action drop*** pour l'interface du bâtiment ES et ***rate-limit input 10000000 3000 4000 conform-action transmit exceed-action drop*** pour l'interface du bâtiment BS. Ces commandes contrôlent efficacement le trafic entrant, garantissant ainsi que la politique de bande passante spécifiée est respectée.

Le testing du réseau :

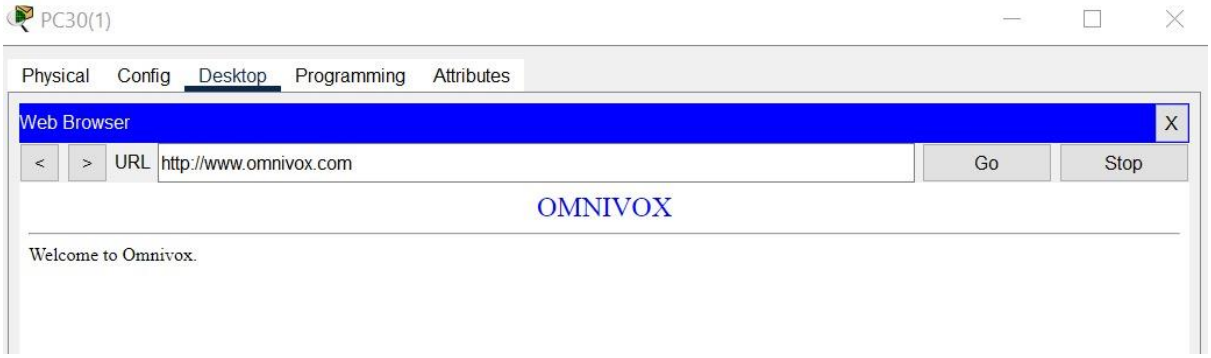
On a le serveur DNS suivant :



On vas le pinger avec un pc connecter au réseau :



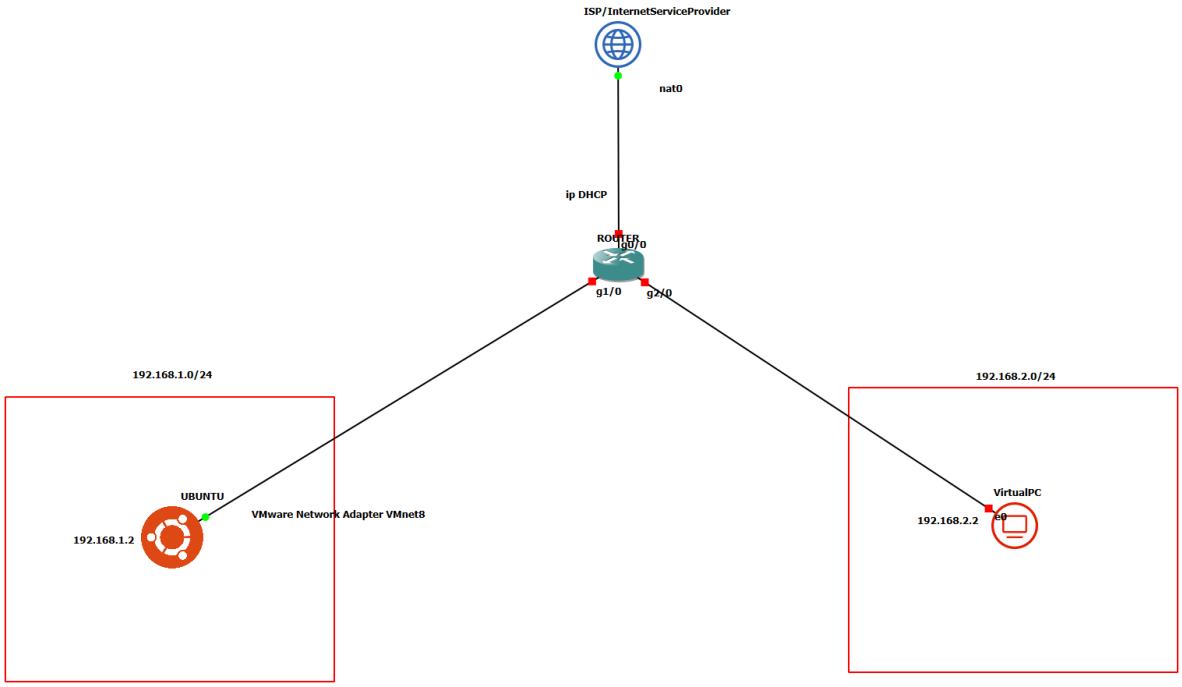
Et on va exécuter la page web www.omnivox .com :



L’intégration de GNS3:

L'évolution de notre environnement de simulation réseau a franchi une étape significative avec l'intégration de **GNS3** en remplacement de **Packet Tracer**. Cette avancée nous permet de connecter le simulateur directement à une machine virtuelle Ubuntu, offrant ainsi une expérience d'accès Internet plus réaliste et concrète. Cette configuration élargit nos capacités de test et de démonstration, en simulant de manière plus précise le comportement et la performance des réseaux dans un contexte réel d'utilisation d'Internet.

Topologie GNS3:



La topologie réalisée dans GNS3 illustre la connexion d'une machine virtuelle Ubuntu et d'un VirtualPC à un routeur configuré pour simuler une connexion Internet via un fournisseur de services Internet (ISP). Le routeur sert de point de distribution, attribuant des adresses IP aux dispositifs via DHCP. L'interface nat0 du routeur est connectée à l'ISP, permettant l'accès au réseau externe. L'interface g1/0, avec une sous-réseau IP de 192.168.1.0/24, est connectée à la machine virtuelle Ubuntu, qui utilise un adaptateur réseau VMware VMnet8 configuré avec l'IP 192.168.1.2. De l'autre côté, l'interface g2/0, avec un sous-réseau IP de 192.168.2.0/24, est connectée à un VirtualPC ayant l'IP 192.168.2.2. Cette configuration permet aux deux systèmes clients de communiquer avec le routeur et d'accéder à Internet, simulant ainsi un environnement réseau fonctionnel pour tester le comportement du réseau et la connectivité Internet.

Configuration du NAT:

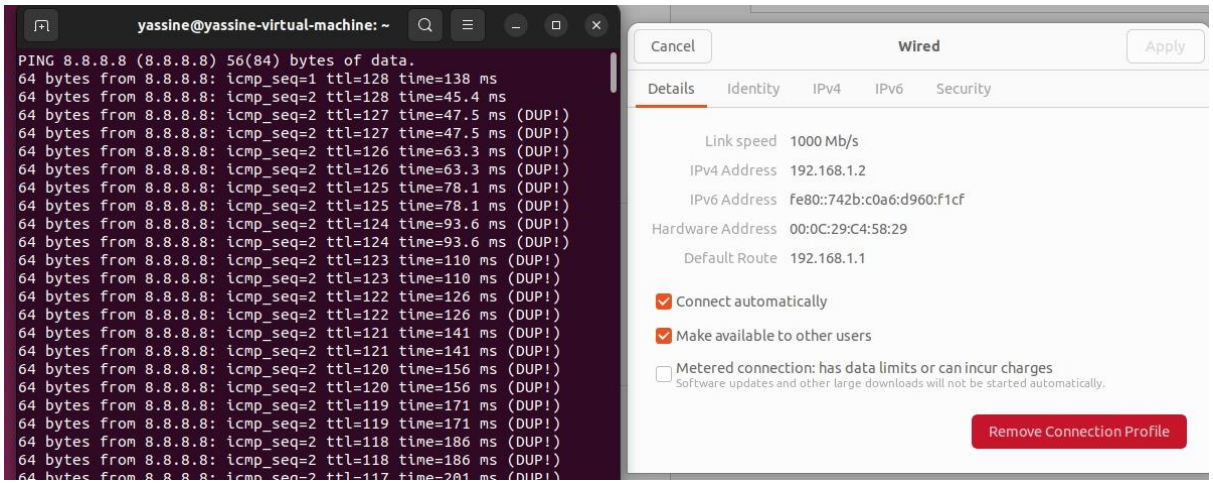
Pour configurer le Network Address Translation (NAT) sur le routeur, nous commençons par entrer en mode de configuration globale avec les commandes enable et configure terminal. Nous configurons ensuite l'interface interne **FastEthernet0/0**, qui est connectée au **VirtualPC**, en lui attribuant l'adresse **IP 192.168.1.1** et en spécifiant qu'il s'agit de l'interface interne pour le NAT avec la commande ***ip nat inside***. L'interface est également activée avec la commande ***no shutdown***. Ensuite, nous passons à l'interface externe **GigabitEthernet1/0**, qui se connecte au Cloud NAT. Nous configurons cette interface pour obtenir son adresse IP automatiquement via DHCP et la désignons comme interface externe pour le NAT avec ***ip nat outsid***e, avant de l'activer avec ***no shutdown***.

Après avoir configuré les interfaces, le routage est activé avec la commande ***ip routing***. Une liste d'accès est créée pour permettre au trafic du réseau interne ***192.168.1.0/24*** de traverser le NAT. Enfin, nous associons la liste d'accès à la source de NAT sur l'interface externe avec la commande ***ip nat inside*** source list 1 interface **GigabitEthernet1/0** ***overload***, permettant ainsi à plusieurs dispositifs de partager une seule adresse IP publique. Pour terminer, nous sortons du mode de configuration avec end et sauvegardons la configuration dans la mémoire du routeur avec ***write memory***. Cela garantit que le NAT est correctement configuré pour traduire les adresses internes en adresses utilisables sur Internet, permettant ainsi une connectivité externe.

Test GNS3:

Pour tester notre réseau nous allons pinger 8.8.8.8 depuis la Vm Ubuntu pour assurer la connectivité avec le réseau internet réel :

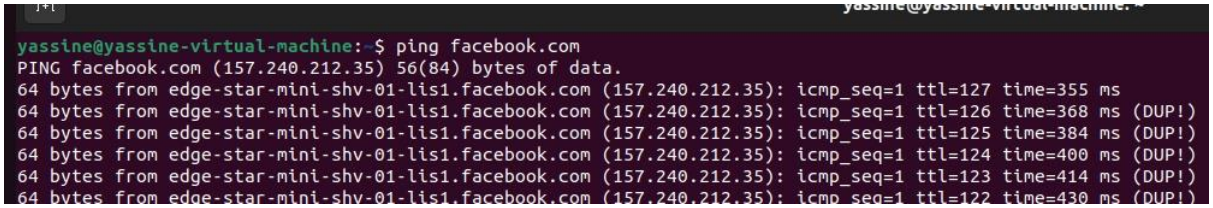
- Ping 8.8.8.8 :



Pour pinger les DNS translater au nom comme facebook.com, il faut tapper d'abbord la commande suivante :

sudo echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf > /dev/null

- Ping facebook.com :



L'intégration de Python:

L'intégration de Python dans notre environnement de simulation réseau a marqué un tournant, en particulier avec l'utilisation de la bibliothèque Scapy comme alternative à Wireshark pour l'analyse et le sniffing de paquets. Scapy est un puissant outil programmable écrit en Python, conçu pour l'interception, la décodage, et l'analyse de paquets réseau. Grâce à sa nature flexible et son architecture extensible, Scapy permet aux utilisateurs de créer leurs propres protocoles ou d'étendre les fonctionnalités existantes avec une facilité remarquable. En utilisant Scapy, nous avons été capables de capter et d'examiner le trafic réseau en temps réel, offrant une compréhension approfondie du comportement des paquets au sein de notre infrastructure simulée. Cette méthode offre un contrôle plus fin et une personnalisation avancée dans l'analyse du trafic réseau, ouvrant la voie à des diagnostics et à des démonstrations éducatives plus sophistiqués.

Pour bloquer les sites web:

La fonction get_websites_to_block invite l'utilisateur à saisir les noms des sites Web qu'il souhaite bloquer, en continuant jusqu'à ce que l'utilisateur saisisse '00' pour terminer l'opération. Ces sites sont ensuite stockés dans une liste. La fonction block_websites prend cette liste de sites Web et les ajoute au fichier hosts, qui est un fichier système utilisé par le système d'exploitation pour mapper les noms de domaine aux adresses IP. Le chemin vers le fichier hosts est défini selon le système d'exploitation cible; dans ce cas, le chemin pour Linux est utilisé, et un chemin alternatif est commenté pour Windows. Chaque site Web de la liste est écrit dans le fichier hosts avec l'adresse IP de redirection 127.0.0.1, qui est l'adresse de bouclage locale de la machine. Ainsi, toute tentative d'accès à ces sites Web sera redirigée vers l'adresse locale, empêchant effectivement l'accès au site. Ce script est un moyen simple mais efficace de restreindre l'accès à des sites Web non désirés sans l'utilisation de logiciels de contrôle parental ou de pare-feu.

```python
def get_websites_to_block():
            websites = []
            print("Enterer les sites web a blocker, tapper '00' pour terminer.")
            while True:
                website = input("Enterer le site web: ")
                if website == '00':
                    break
                websites.append(website)
            return websites


        def block_websites(websites):
            hosts_path = "/etc/hosts"  # pour Windows, il faut utiliser 'C://Windows//System32//drivers//etc//hosts'
            redirect_ip = "127.0.0.1"

            with open(hosts_path, 'a') as file:
                for website in websites:
                    file.write(f"\n{redirect_ip} {website}")
                    print(f"{website} blocked")


        websites_to_block = get_websites_to_block()
        block_websites(websites_to_block)

```

Pour flagger les paquets suspects :

```python
if packet.haslayer(TCP):
                    tcp_layer = packet[TCP]

                    if tcp_layer.flags & 0x02 and tcp_layer.flags & 0x01:
                        with open("C://Users//Hamza//Desktop//worrying_packets.txt", "a") as file:
                            file.write(f"Paquet préoccupant trouvé: {packet.summary()}\n")
                        print("Paquet préoccupant trouvé (TCP avec SYN et FIN)")
```

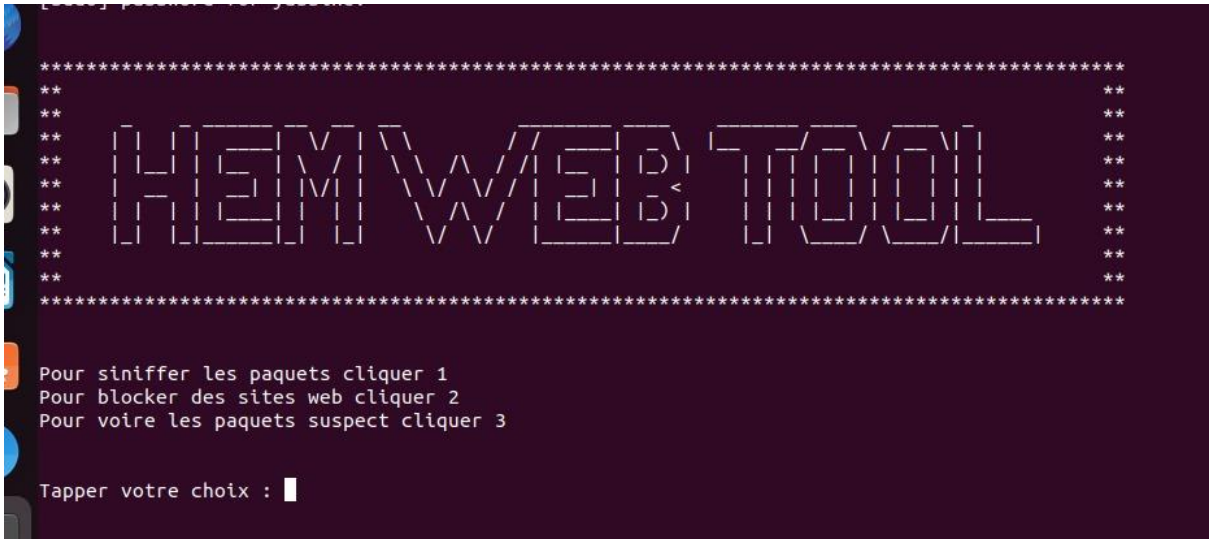
Cette partie du code Python utilise Scapy pour analyser les paquets réseau afin de détecter des paquets TCP qui présentent des caractéristiques suspectes. En particulier, le script vérifie si le paquet contient une couche TCP en utilisant packet.haslayer(TCP). Si c'est le cas, il accède à cette couche avec packet[TCP].

Le script examine ensuite les drapeaux du segment TCP pour identifier si les drapeaux SYN et FIN sont tous deux définis en utilisant tcp_layer.flags & 0x02 pour SYN et tcp_layer.flags & 0x01 pour FIN. En TCP, les drapeaux SYN et FIN sont utilisés pour établir et terminer une connexion, respectivement. La présence simultanée de ces deux drapeaux dans un seul paquet est atypique et peut indiquer une activité réseau anormale ou malveillante, telle qu'une tentative de scan de port ou un paquet malformé.

Lorsqu'un tel paquet est détecté, le script enregistre un résumé du paquet dans un fichier texte intitulé "worrying_packets.txt" situé sur le bureau de l'utilisateur Hamza. Le chemin du fichier est codé en dur dans le script. En outre, il affiche un message dans la console pour alerter l'utilisateur qu'un "Paquet préoccupant trouvé (TCP avec SYN et FIN)". Cela permet à l'utilisateur d'être informé en temps réel et d'avoir un enregistrement permanent des paquets suspects pour une analyse ultérieure.

Tester le code :

- La page home :



- Le choix des interfaces :

- Lors du pinging de 8.8.8.8 le script pyhton :

```
Ether / IP / ICMP 192.168.1.1 > 8.8.8.8 echo-request 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 8.8.8.8 > 192.168.1.1 echo-reply 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 8.8.8.8 > 192.168.1.2 echo-reply 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 192.168.1.1 > 8.8.8.8 echo-request 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 192.168.1.1 > 8.8.8.8 echo-request 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 8.8.8.8 > 192.168.1.1 echo-reply 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 8.8.8.8 > 192.168.1.2 echo-reply 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 192.168.1.1 > 8.8.8.8 echo-request 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 192.168.1.1 > 8.8.8.8 echo-request 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 8.8.8.8 > 192.168.1.1 echo-reply 0 / Raw, Size: 98 bytes
Ether / IP / ICMP 8.8.8.8 > 192.168.1.2 echo-reply 0 / Raw, Size: 98 bytes
```

- Lors du pinging de 8.8.8.8 wireshark :

Appliquer un filtre d'affichage ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	I
1348…	245.849104	192.168.179.2	192.168.1.1	ICMP	98	T
1348…	245.849145	192.168.1.1	8.8.8.8	ICMP	70	T
1348…	245.849256	192.168.1.1	8.8.8.8	ICMP	70	T
1348…	245.849360	192.168.179.2	192.168.1.1	ICMP	98	T
1348…	245.849416	192.168.179.2	192.168.1.1	ICMP	98	T
1348…	245.849612	192.168.1.1	8.8.8.8	ICMP	70	T
1348…	245.849654	192.168.1.1	8.8.8.8	ICMP	70	T
1348…	245.849707	192.168.179.2	192.168.1.1	ICMP	98	T
1348…	245.849762	192.168.179.2	192.168.1.1	ICMP	98	T

Le code source:

```
from scapy.all import conf, sniff, TCP
import time
from scapy.all import conf

print()
print()
```

```
print("*****************************************************************************************")
print("**                                                                                     **")
print("**    _   _ _____ __  __  __        _______ ____    _____ ____  ___  _              **")
print("**   | | | | ____|  \/  | \ \      / /  ___|  _ \  |__   __/ __ \/ __ \| |             **")
print("**   | |_| |  _| | |\/| |  \ \ /\ / /| |_  | |_) |    | | | |  | | |  | | |             **")
print("**   |  _  | |___| |  | |   \ V  V / |  _| |  _ <     | | | |__| | |__| | |___          **")
print("**   |_| |_|_____|_|  |_|    \_/\_/  |_____|_| \_\    |_|  \____/ \____/|_____|         **")
print("**                                                                                     **")
print("**                                                                                     **")
print("*****************************************************************************************")
```

```
print()
print()
```

```
print("Pour siniffer les paquets cliquer 1  ")
print("Pour blocker des sites web cliquer 2 ")
print("Pour voire les paquets suspect cliquer 3 ")

print()
print()


z = input("Tapper votre choix : ")

if z.isdigit() and int(z) == 1:

        def process_packet(packet):
            packet_size = len(packet)
            print(f"{packet.summary()}, Size: {packet_size} bytes")

            if packet.haslayer(TCP):
                tcp_layer = packet[TCP]

                if tcp_layer.flags & 0x02 and tcp_layer.flags & 0x01:
                    with
open("C://Users//Hamza//Desktop//worrying_packets.txt", "a") as file:
                        file.write(f"Paquet préoccupant trouvé:
{packet.summary()}\n")
                    print("Paquet préoccupant trouvé (TCP avec SYN et FIN)")

            time.sleep(1)

        def list_interfaces():

            interfaces = [(iface, conf.ifaces[iface].description,
conf.ifaces[iface].name) for iface in conf.ifaces]
            print("Les interfaces dispo:")
            for idx, (iface_idx, description, name) in enumerate(interfaces):
                print(f"Index: {idx}")
                print(f"  Description: {description}")
                print(f"  ID: {iface_idx}")
                print(f"  Nom: {name}")
                print("-" * 30)
            return interfaces


        interfaces = list_interfaces()
        choice = input("Entrer le num d'interface (tapper 'general' pour toute
les interface): ")

        if choice.lower() == 'general':
            print("Sniffing tout les interfaces...")
            sniff(prn=process_packet, store=False)
```

```
        elif choice.isdigit() and int(choice) < len(interfaces):

            selected_interface_idx = int(choice)
            selected_interface =
conf.ifaces[interfaces[selected_interface_idx][0]].name
            print(f"Sniffing L'interface: {selected_interface}")
            sniff(prn=process_packet, store=False, iface=selected_interface)

        else:
            print("Choix invalide.")




elif z.isdigit() and int(z) == 2:

        def get_websites_to_block():
            websites = []
            print("Enterer les sites web a blocker, tapper '00' pour
terminer.")
            while True:
                website = input("Enterer le site web: ")
                if website == '00':
                    break
                websites.append(website)
            return websites


        def block_websites(websites):
            hosts_path = "/etc/hosts"  # pour Windows, il faut utiliser
'C://Windows//System32//drivers//etc//hosts'
            redirect_ip = "127.0.0.1"

            with open(hosts_path, 'a') as file:
                for website in websites:
                    file.write(f"\n{redirect_ip} {website}")
                    print(f"{website} blocked")


        websites_to_block = get_websites_to_block()
        block_websites(websites_to_block)


elif  z.isdigit() and int(z) == 3:

     with open("C://Users//Hamza//Desktop//worrying_packets.txt", "r") as
file:
```

```
            data = file.read()
            if data:
                print("Paquets dangeureux enregistrees :")
                print(data)
            else:
                print("Aucun paquet dangeureux enregistree.")


else:

    print("Choix non valide")
```

Nous tenons à vous remercier sincèrement pour votre soutien et votre direction durant le développement de ce projet. Votre expertise et votre guidance ont été précieux pour nous et ont permis de mener à bien ce projet avec succès. Nous sommes reconnaissants pour votre engagement envers notre apprentissage