IRC sur tor

Introduction

Il y'a plusieurs années j'ai installé un serveur ircd-hybrid lorsque j'étais encore etudiant en bts SIO dans ma documentation j'ai exploré l'installation d'un serveur IRC l'ajout d'une identification chiffrement des échanges entre client et serveur ajout d'un operateur etc.

Aujourdhui à l'université en M2 à pantheon-sorbonne en lisant un mémoire sur tor et les hidden service j'ai vu que nous pouvons via le réseau TOR hébergé des services TCP (mode connecté et non mode déconnecté) ce qui a attiré mon attention sur le fait que TOR n'est pas juste un navigateur qui prend en charge le protocole http/HTTPS mais n'importe quelle service qui tourne sur TCP en théorie peut être hebergé sur TOR et avoir une adresse en « .onion » mais il faudra impérativement un proxy SOCKS tor

L'utilisateur devra utilisé le proxy avant de se connecter par exemple via HexChat il faudra mettre dans Paramètres réseau :

- Type : SOCKS5
- Host : `127.0.0.1`
- Port : `9050`

Ajout doc IRCD-HYBRID

Je vais mettre quelques rajouts comme mémo par rapport à la doc sur ircd-hybrid

Mise en place d'un accès opérateur sur un serveur IRC (ircd-hybrid)

1. Principe

Sur un serveur IRC basé sur ircd-hybrid, l'accès administrateur ne repose pas sur un système de compte classique.
L'élévation de privilèges se fait via la commande :

```
/OPER <nom_operateur> <mot_de_passe>
```

L'accès est conditionné par trois éléments :

- un nom d'opérateur (name)
- un mot de passe (password)
- un masque utilisateur (user@host)

2. Configuration du serveur

La configuration se fait dans le fichier principal de ircd-hybrid (généralement `ircd.conf`).

Exemple de bloc opérateur

```
operator {
    name = "superadmin";
    user = "~admin@192.168.1.10";
    password = "motdepasse123";
    encrypted = no;
}
```

Explication des champs

- `name` : identifiant utilisé dans la commande `/OPER`
- `user` : masque de correspondance basé sur ident et adresse IP
- `password` : mot de passe en clair ou hashé
- `encrypted` :
    - `no` si mot de passe en clair
    - `yes` si mot de passe hashé

3. Point critique : le masque user

Le serveur vérifie le format réel de connexion :

```
pseudo!ident@host
```

Exemple réel :

```
User!~admin@192.168.1.10
```

Le caractère `~` est souvent présent devant l'ident.
Il doit être inclus dans la configuration, sinon le matching échoue.

Exemple correct :

```
user = "~admin@192.168.1.10";
```

Exemple trop large :

```
user = "~admin@*";
```

Exemple recommandé (plus sécurisé) :

```
user = "~admin@192.168.1.10";
```

4. Configuration du client IRC

Avant la connexion, il faut définir l'ident (username) côté client.

Exemple de configuration :

```
username = admin
nickname = SuperUser
```

Lors de la connexion, le serveur verra :

```
SuperUser!~admin@IP
```

5. Connexion au serveur

Connexion classique :

```
serveur : 192.168.1.20
port    : 6667
```

Ou en TLS si configuré :

```
port    : 6697
```

6. Vérification de l'ident

Une fois connecté :

```
/whois SuperUser
```

Résultat attendu :

```
SuperUser!~admin@192.168.1.10
```

Si l'ident ne correspond pas, la commande `/OPER` échouera.

7. Passage en opérateur

Commande à exécuter :

```
/oper superadmin motdepasse123
```

Si la configuration est correcte :

```
You are now an IRC operator
```

8. Vérification des privilèges

```
/mode SuperUser
```

Résultat attendu :

```
+izo
```

Le `+o` indique que l'utilisateur est opérateur serveur.

9. Différence avec opérateur de salon

Un opérateur de salon possède uniquement des droits locaux :

- gestion d'un channel
- kick et ban dans un salon

Un opérateur serveur possède des droits globaux :

- déconnexion d'utilisateurs
- bannissement serveur
- gestion du serveur

10. Commandes de base opérateur

Déconnexion d'un utilisateur :

```
/kill pseudo raison
```

Bannissement :

```
/kline *@192.168.1.50
```

Suppression du ban :

```
/unkline *@192.168.1.50
```

Rechargement de la configuration :

```
/rehash
```

11. Sécurisation recommandée

Limiter le masque utilisateur :

```
user = "~admin@192.168.1.10";
```

Éviter :

```
user = "*@*";
```

Utiliser un mot de passe fort ou hashé.

12. Résultat

Après configuration correcte :

- connexion utilisateur fonctionnelle
- ident correctement reconnu
- accès opérateur validé via `/OPER`
- contrôle complet du serveur IRC disponible

Gestion des échanges et journalisation

Les échanges sur le serveur IRC ne sont pas conservés par défaut. Le protocole fonctionne en temps réel : les messages sont simplement relayés entre utilisateurs sans stockage persistant côté serveur. Les communications peuvent être chiffrées via TLS, ce qui protège les données en transit contre l'interception réseau. Toutefois, le serveur traite les messages en clair pour pouvoir les redistribuer ; en cas de compromission du serveur, il est donc théoriquement possible d'accéder aux échanges en cours. Par ailleurs, la journalisation dépend du client IRC utilisé : de nombreux clients enregistrent localement les conversations (salons et messages privés) sous forme de logs, généralement en clair sur la machine de l'utilisateur. Ces fichiers constituent une source sensible et doivent être protégés en conséquence.

Solution OTR

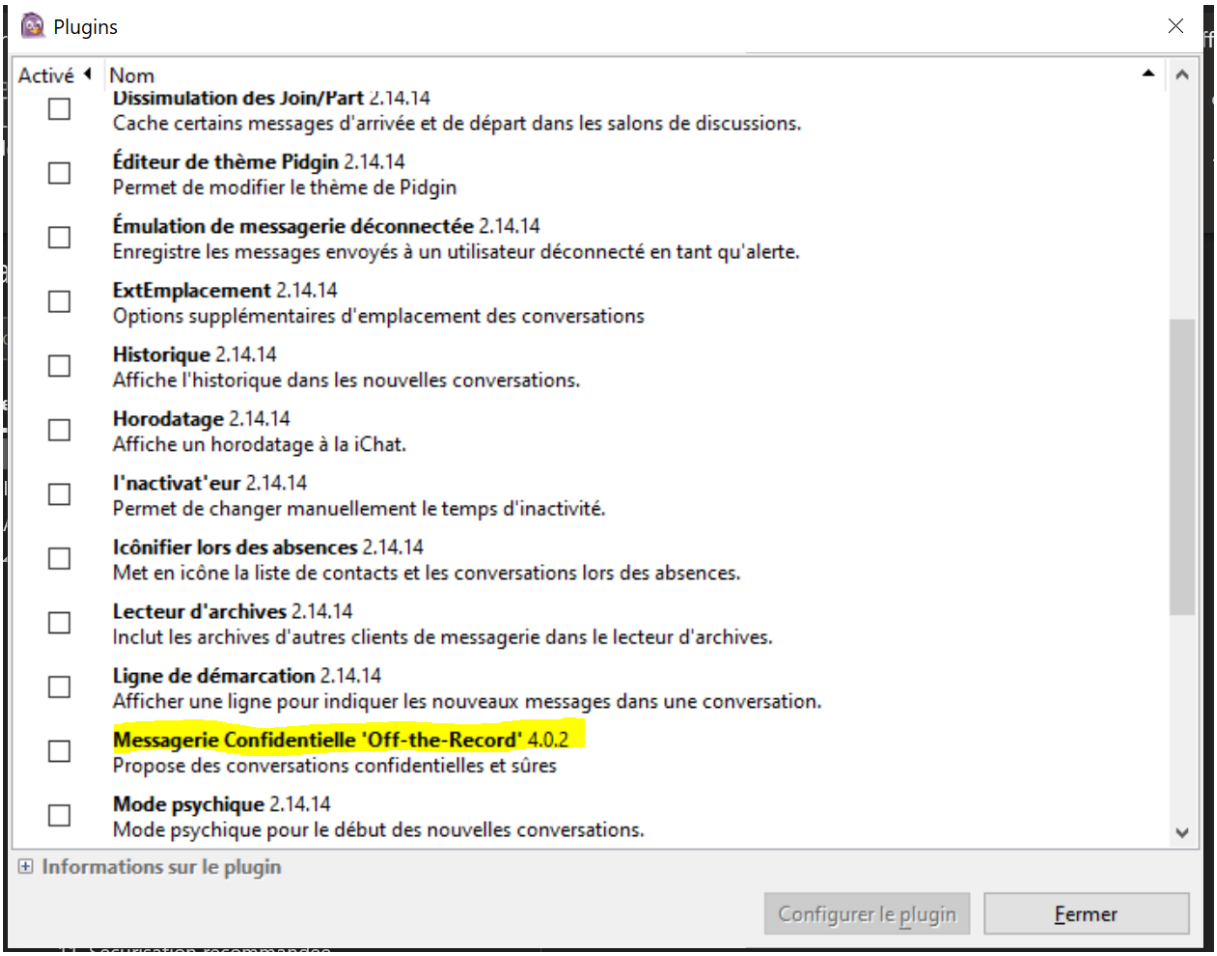
OTR (Off-the-Record Messaging) est un protocole de chiffrement de bout en bout appliqué au niveau des clients de messagerie. Dans le cadre d'IRC, il permet de sécuriser les conversations privées en chiffrant les messages directement côté utilisateur avant leur envoi au serveur. Le serveur IRC, tel qu'ircd-hybrid, ne fait que relayer les données chiffrées sans pouvoir en lire le contenu. OTR repose sur des mécanismes cryptographiques modernes (échange de clés Diffie-Hellman, chiffrement symétrique AES, authentification HMAC) et offre des garanties fortes telles que la confidentialité, la Perfect Forward Secrecy et la dénégation plausible. Ainsi, même en cas de compromission du serveur ou d'interception du trafic réseau, le contenu des échanges reste protégé, sous réserve que les deux interlocuteurs utilisent OTR et que les postes clients soient eux-mêmes sécurisés.

Utiliser un client compatible OTR, qui permettra un chiffrement de bout en bout via echange de clef symetrique via Diffie Hellman et chiffrement AES le serveur ne voit que du texte chiffré illisible et cela repond à une attaque man in middle

Installation pidgin client sur windows avec pidgin otr comme plugion

A ce lien : Off-the-Record Messaging

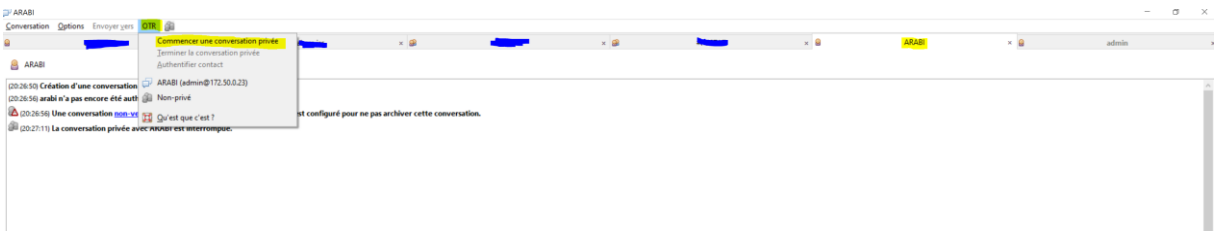
Ensuite une fois lancer le .exe + client IRC relancer on peut voir le nouveau plugin a activé



Ensuite générer une clef par compte

Initier conversation chiffrer

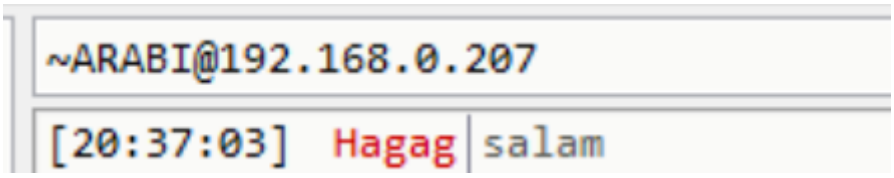
Je suis l'user Hagag et j'essaye de parler à Arabi dans ma fenetre de conversation privé j'inite la conversation OTR comme cela



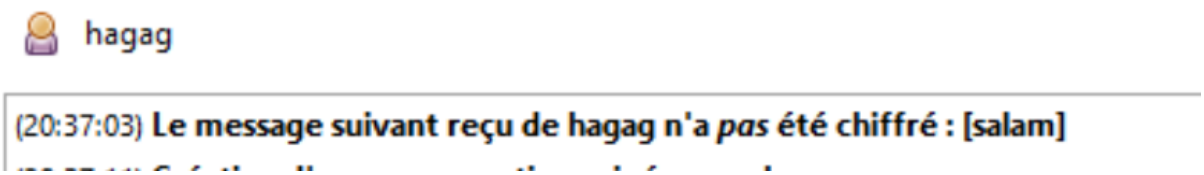
Test entre client supportant OTR et un autre ne supportant pas OTR

Hagag est connecté sur HexChat ne supportant pas OTR quand je communique avec Arabie voila ce qu'il se passe

Hagag non chiffré initie l'échange

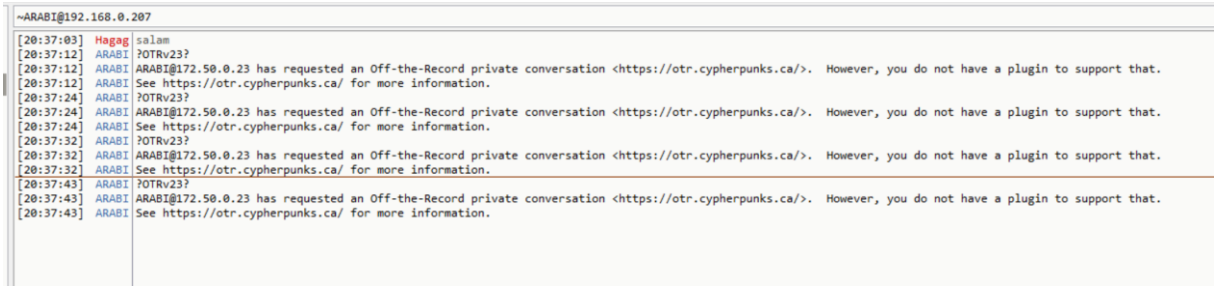


Arabi (utilise OTR) vois cela



J'initie une conversation OTR qui biensur n'aboutira pas mais fonctionnera que d'un sens

Coté Hagag non compatible OTR



Analyse via TCPDUMP

tcpdump -i any port 6667 -A

J'ai volontairement configuré le serveur pour qu'il n'utilise plus TLS + port 6667

Sans OTR

Hagag envoie un message à Arabi sans OTR

21:00:02.293048 wlan0 In IP 192.168.0.207.55173 > 172.50.0.23.ircd: Flags [.], ack 1468, win 1025, length 0

E..(.y@.....2.....A..V.~P.....

21:00:16.377283 wlan0 In IP 192.168.0.207.54585 > 172.50.0.23.ircd: Flags [P.], seq 136:156, ack 97, win 1022, length 20

E..<.|@....~.....2...9...2.>e.9KP....C..PING LAG3012923550

```
21:00:02.245144 wlan0 Out IP 172.50.0.23.ircd > 192.168.0.207.54585: Flags [.], ack 136, win 1003, length 0
E..(.x@.@....2.........9e.9K.2.>P.......
21:00:02.245626 wlan0 Out IP 172.50.0.23.ircd > 192.168.0.207.55173: Flags [P.], seq 1415:1468, ack 1707, win 1002, length 53
E..].X@.@..r.2...........V.I..A.P.......:Al-Hajjaj!~Emir@192.168.0.207 PRIVMSG ARABI :SALAM
```

Avec OTR

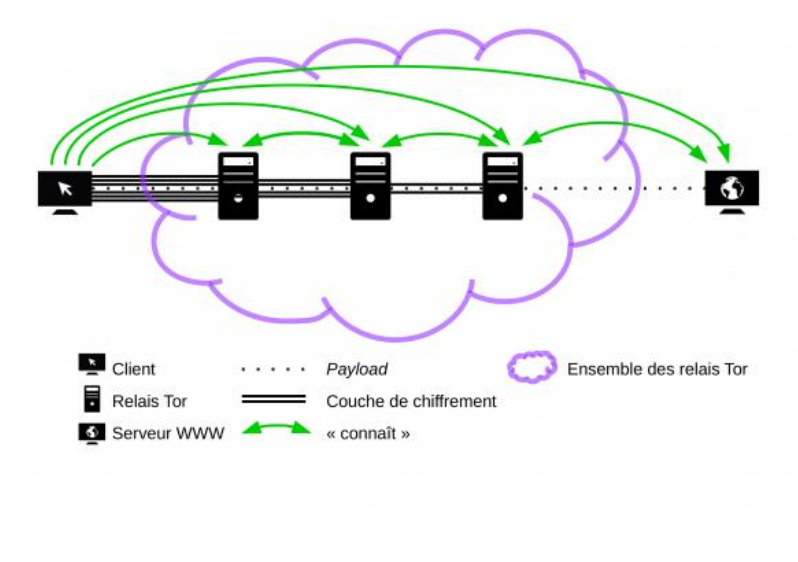
Admin envoie Salam à Arabi

Résultat dans tcpdump

On voit bien que le message ici est chiffre quand il transite par le serveur c'est excellent nous voyons juste les deux utilisateurs qui communiquent

Hidden Service TOR/IRC

Tor n'est pas qu'un moyen d'accéder à des sites. onion on peut aussi communiquer via n'importe quelle protocole d'application qui utilise TCP donc en pratique on peut utiliser un service caché qui ne révèlera pas la position originale du serveur ni celle du client un schema est beaucoup plus intéressant à ce niveau

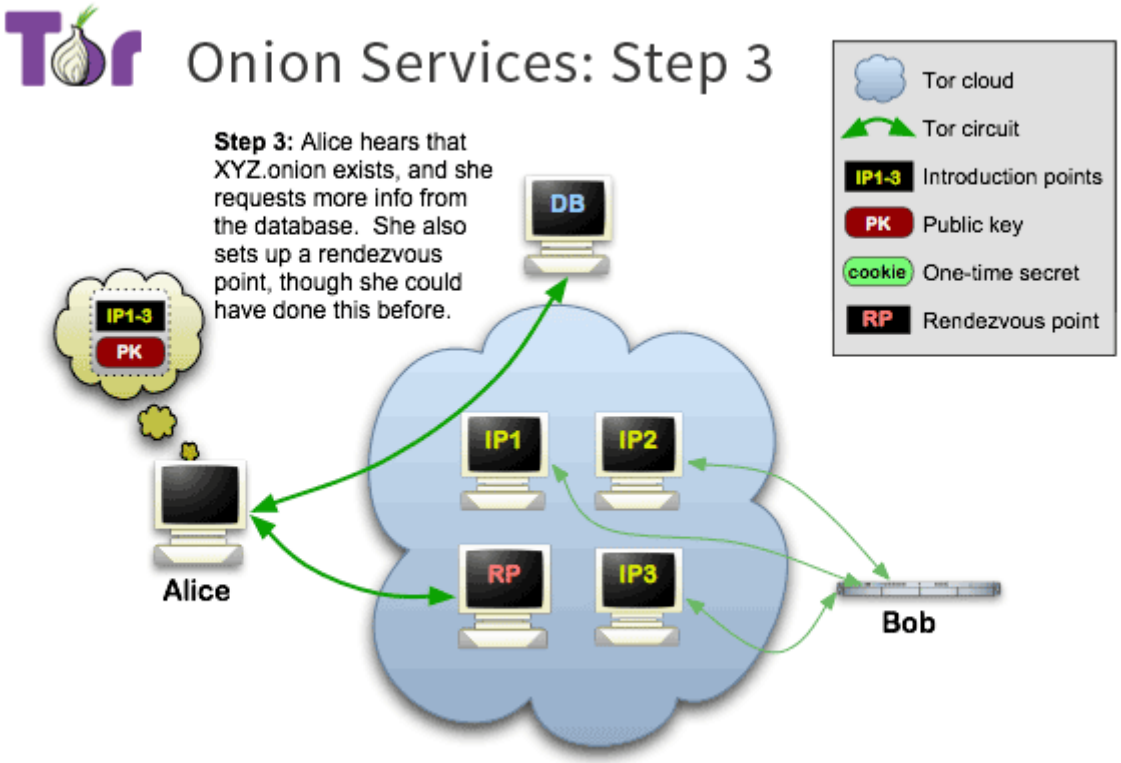


Il faut que notre service n'écoute que sur son adresse local « 127.0.0.1 » avant tout

Alice souhaite atteindre Bob, elle devra le contacter par l'intermédiaire de son adresse DNS XYZ.onion. Après cela, le client peut lancer une tentative de connexion en téléchargeant le descripteur du service caché en provenance du serveur annuaire. S'il y a un descripteur pour XYZ.onion, le client crée alors un circuit vers un autre nœud au hasard et lui demande d'agir comme un **point de rendez-vous** en lui communiquant un **secret partagé.**

Qu-est-ce qu'un serveur rendez-vous ?

Serveur rendez-vous : Nœud Tor tiré au hasard qui est chargé d'agir comme un point de rendez-vous qui **agit comme entremetteur entre les 2 hôtes**. Sans ce serveur « Rendez-vous », il est impossible d'initier une connexion entre les deux hôtes. Le but étant que **chaque partie prenante ne soit jamais directement en contact**.



Il faut ce rendre dans ce fichier `/etc/tor/torrc`

Et modifier comme tel il faut biensur decommenter le RunAsDaemon avant dans le fichier

```
############### This section is just for location-hidden services ###

## Once you have configured a hidden service, you can look at the
## contents of the file ".../hidden_service/hostname" for the address
## to tell people.
##
## HiddenServicePort x y:z says to redirect requests on port x to the
## address y:z.

HiddenServiceDir /var/lib/tor/irc/
HiddenServicePort 6697 127.0.0.1:6697

#HiddenServiceDir /var/lib/tor/other_hidden_service/
#HiddenServicePort 80 127.0.0.1:80
#HiddenServicePort 22 127.0.0.1:22
```

Ensuite coté ircd je n'écoute maintenant que en local

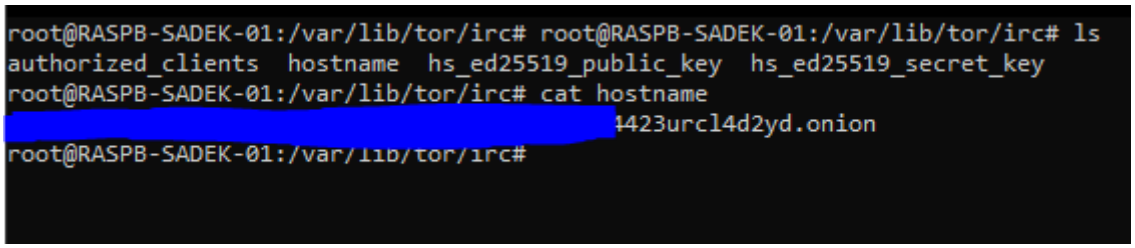
```
        */
       flags = hidden,tls
       host = "127.0.0.1"; # change this!
       port = 6697;

       /*
```

Je redémarre le service tor et ircd

Je me rends dans ce fichier « /var/lib/tor/irc »

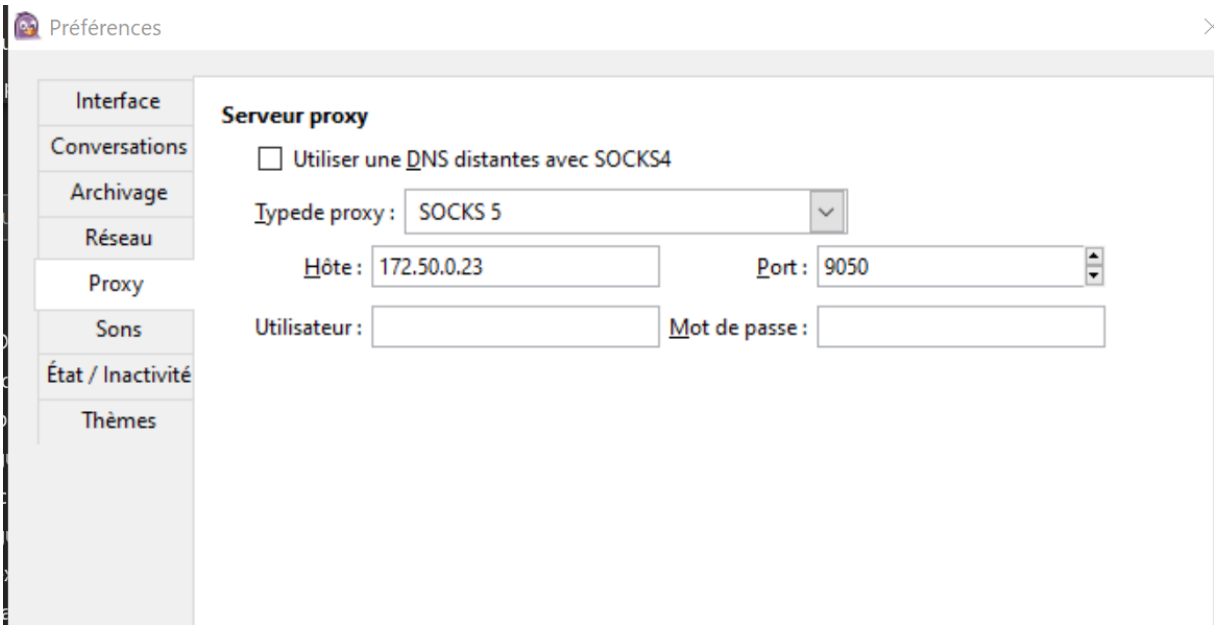
Dedans il y'a un fichier « hostname »

Et il y'aura le FQDN de ma machine sur le réseau tor

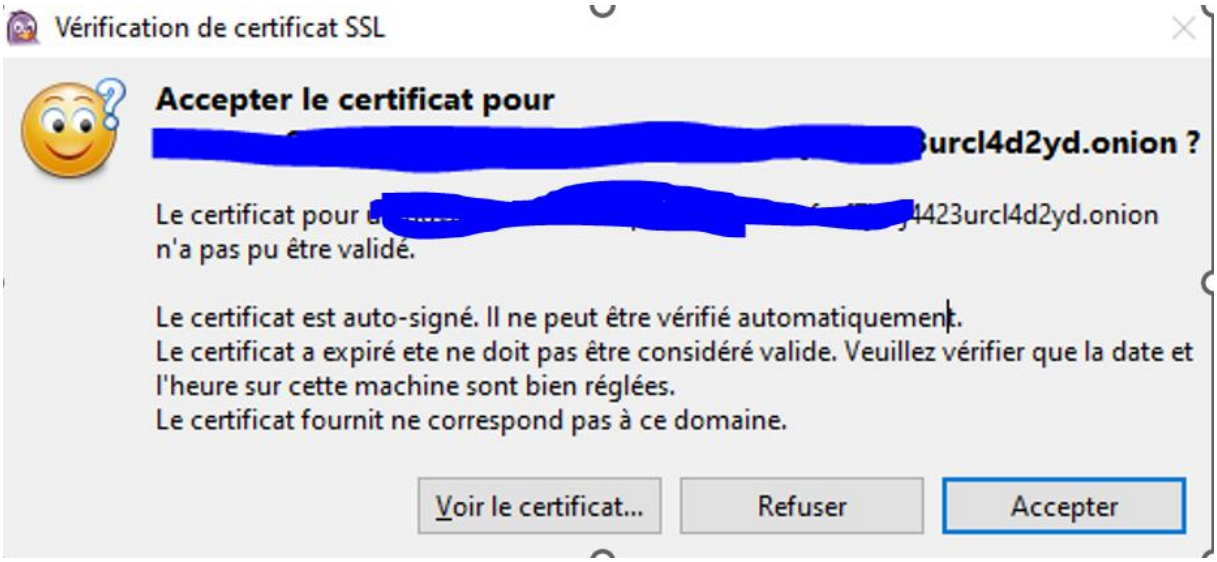


Voici mon url dans le réseau TOR

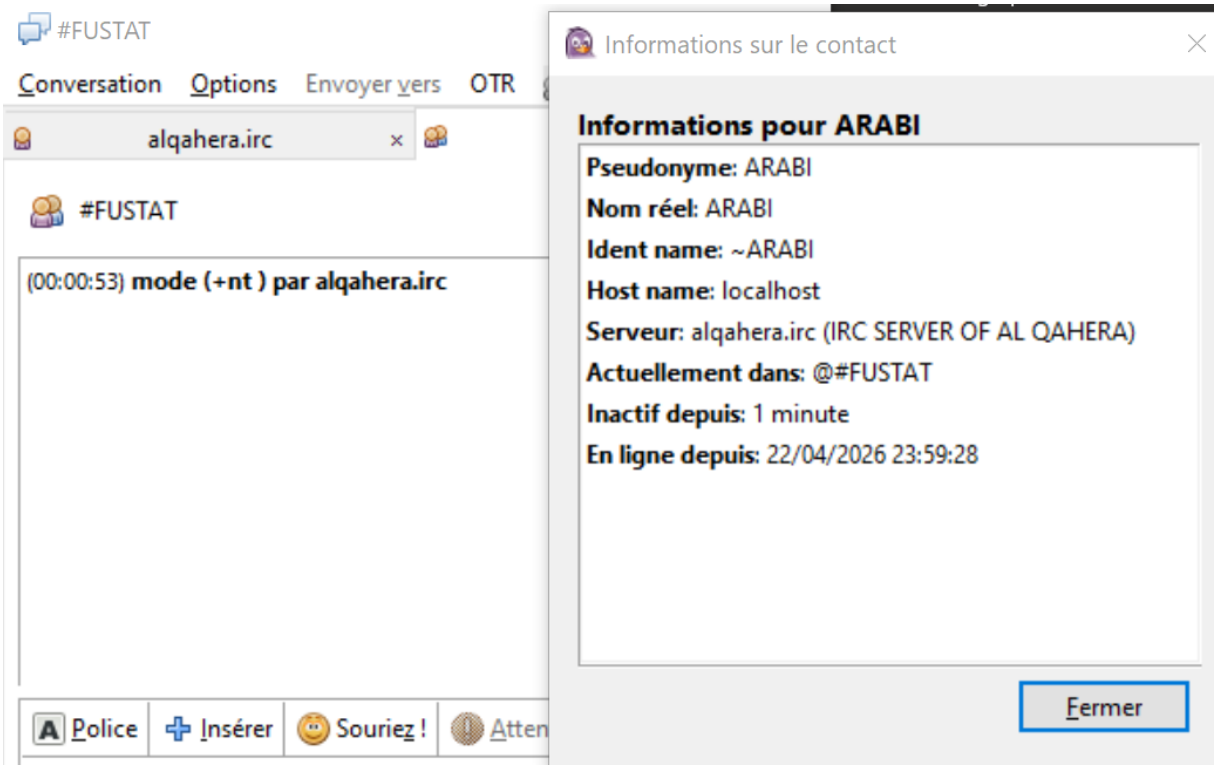
Ensuite dans pidgin je mets moin proxy socks tor comme proxy et je met l'adresse .onion comme serveur



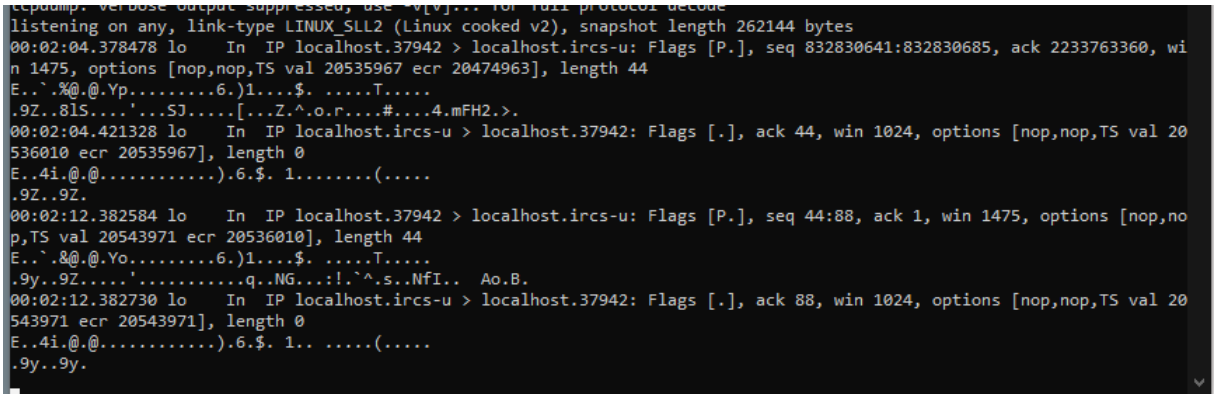
Je me connecte ensuite



Lorsque je fais un WHOIS voila ce que je vois aucune IP et l'ip source est localhost



J’ai envoyé deux messages les deux proviennent bien de localhost



Conclusion

Cette étude met en évidence les limites intrinsèques du protocole IRC en matière de sécurité, notamment l’absence de chiffrement natif des échanges. L’ajout de mécanismes comme TLS, OTR et TOR permet de renforcer significativement la confidentialité et l’anonymat, mais chaque solution répond à une problématique différente. Une approche combinée est nécessaire pour obtenir un niveau de sécurité satisfaisant.