**Prise en main**

Que pensez-vous de la confidentialité des données vis à vis du serveur ?

Du coté serveur on peut avoir le message ainsi que le destinataire du message facilement.

Écrivez un script (simple\_bigbrother.py, max 40 lignes) qui vous permettra d’écouter toutes les conversations.

Écrivez un script (simple\_dos.py, max 40 lignes) qui vous permettra de créer un déni de service pour les utilisateurs.

Pouvez-vous expliquer en quoi la sérialisation pickle est certainement le plus mauvais choix ?

La sérialisation avec pickle est un très mauvais choix pour un protocole de communication réseau, surtout dans un contexte de chat non sécurisé, pour plusieurs raisons :

**1. Exécution de code arbitraire**

Pickle peut exécuter du code arbitraire lors de la désérialisation, ce qui en fait une faille de sécurité majeure. Un attaquant pourrait envoyer une charge utile malveillante qui exécute du code sur le serveur, permettant une prise de contrôle totale.

**2. Aucune vérification d’intégrité**

Contrairement à des formats comme JSON ou Protobuf, pickle n’intègre aucune vérification d’intégrité ou de structure des données. Cela signifie que des données corrompues ou malformées peuvent provoquer des erreurs ou des comportements imprévus.

**3. Non portable et difficile à inspecter**

Les objets pickle sont des binaires non lisibles par l’humain. Contrairement à JSON, qui est facilement inspectable et modifiable, pickle rend difficile le debugging et la compatibilité entre versions de Python.

**4. Attaque par relecture (Replay Attack)**

Puisque pickle ne signe ni ne chiffre les données, un attaquant pourrait capturer un paquet réseau valide et le rejouer indéfiniment pour provoquer des effets indésirables (ex. usurpation d’identité, spam de messages, etc.).

**5. Manque de standardisation**

Pickle est spécifique à Python, alors que des formats comme JSON, MessagePack ou Protobuf sont compatibles avec de nombreux langages. Cela limite l’évolutivité du protocole et rend difficile l’interopérabilité.

A partir du code ci dessous, réaliser une exécution arbitraire de code côté serveur

(simple\_exploite.py) (Remote Code Execution, RCE) :

[BONUS] A partir de l’exemple précédent, faire une RCE permettant d’exécuter une reverse

shell. Si vous y arrivez, vous êtes le boss ! (reverse\_shell.py)

Quels types de sérialisation pourrait-on utiliser pour éviter cela ? (hors CVE)

**Pour lancer client**

few1goumz@FAMILY-PC:~$ cd /mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source

few1goumz@FAMILY-PC:/mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source$ pip install names\_generator

error: externally-managed-environment

× This environment is externally managed

╰─> To install Python packages system-wide, try apt install

python3-xyz, where xyz is the package you are trying to

install.

If you wish to install a non-Debian-packaged Python package,

create a virtual environment using python3 -m venv path/to/venv.

Then use path/to/venv/bin/python and path/to/venv/bin/pip. Make

sure you have python3-full installed.

If you wish to install a non-Debian packaged Python application,

it may be easiest to use pipx install xyz, which will manage a

virtual environment for you. Make sure you have pipx installed.

See /usr/share/doc/python3.12/README.venv for more information.

note: If you believe this is a mistake, please contact your Python installation or OS distribution provider. You can override this, at the risk of breaking your Python installation or OS, by passing --break-system-packages.

hint: See PEP 668 for the detailed specification.

few1goumz@FAMILY-PC:/mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source$ source venv/bin/activate

(venv) few1goumz@FAMILY-PC:/mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source$ pip install names\_generator

Requirement already satisfied: names\_generator in ./venv/lib/python3.12/site-packages (0.2.0)

Requirement already satisfied: cmdkit>=2.7.4 in ./venv/lib/python3.12/site-packages (from names\_generator) (2.7.7)

(venv) few1goumz@FAMILY-PC:/mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source$ python3 simple\_client.py

DEBUG:asyncio:Using selector: EpollSelector

DEBUG:asyncio:Using selector: EpollSelector

INFO:pywebio.platform.tornado:Try open http://127.0.0.1:33461 in web browser

gio: http://127.0.0.1:33461: Operation not supported

DEBUG:pywebio.session.threadbased:Callback thread start

DEBUG:BaseClient:Send 'b'\x80\x04\x95-\x00\x00\x00\x00\x00\x00\x00}\x94(\x8c\x04type\x94\x8c\x04join\x94\x8c\x04nick\x94\x8c\x10eloquent\_shannon\x94u.'', recv 'b'\x80\x04\x95\x14\x00\x00\x00\x00\x00\x00\x00}\x94\x8c\x08response\x94\x8c\x02ok\x94s.''

DEBUG:BaseClient:Recv from broadcast 'b'\x80\x04\x958\x00\x00\x00\x00\x00\x00\x00}\x94(\x8c\x04type\x94\x8c\x07message\x94\x8c\x04nick\x94\x8c\x0felegant\_pasteur\x94h\x02\x8c\x04qezf\x94u.''

DEBUG:SimpleClient[eloquent\_shannon]:Received broadcast frame {'type': 'message', 'nick': 'elegant\_pasteur', 'message': 'qezf'}

**Pour lancer le BigBrother**

few1goumz@FAMILY-PC:~$ cd /mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source

few1goumz@FAMILY-PC:/mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source$ python3 simple\_bigbrother.py

**Pour lancer le Server**

few1goumz@FAMILY-PC:~$ cd /mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source

few1goumz@FAMILY-PC:/mnt/c/Users/B.FAMILY/Desktop/TP-unsecured-chat/source$ python3 simple\_server.py