Faculté des sciences et de la technologie

Département Math et Informatique

Réalisé par Dr gattal elhachemi

TP Réseaux avancés — Protocoles applicatifs

Wireshark • GNS3 • Python

Date : 08/11/2025

1) Objectifs pédagogiques

- • Identifier le rôle et le fonctionnement des protocoles applicatifs (HTTP/HTTPS, DNS, DHCP, FTP, SMTP/IMAP/POP3, SNMP, Telnet).
- • Monter un mini-réseau sur GNS3, déployer des services et générer du trafic.
- • Analyser les échanges avec Wireshark (filtres, champs clés, chronologie).
- • Écrire de petits scripts Python pour interroger des services et automatiser des tests.
- • Produire des captures annotées et des conclusions techniques.

2) Pré■requis

- • GNS3 (version récente), Wireshark, Python 3.10+ installés.
- • Deux nœuds Linux (VM ou conteneurs) : Client■LINUX et Serveur■LINUX.
- • Droits sudo sur les nœuds, bases IP/TCP/UDP et commandes Linux.

3) Topologie GNS3

```
[Client-LINUX] ---- [Switch GNS3] ---- [Serveur-LINUX]
       \____ Capture Wireshark sur le lien client-switch (ou sur le client)

Adressage (exemple) :
- Serveur : 10.10.10.1/24
- Client : via DHCP (ex. 10.10.10.100/24)
```

4) Mise en place des services (Serveur■LINUX)

4.1 DHCP (attribution IP)

```
sudo apt update
sudo apt install -y isc-dhcp-server
sudo bash -c 'cat > /etc/dhcp/dhcpd.conf << "EOF"
default-lease-time 600;
max-lease-time 7200;
authoritative;
subnet 10.10.10.0 netmask 255.255.255.0 {
  range 10.10.10.100 10.10.10.150;
  option routers 10.10.10.1;
  option domain-name-servers 10.10.10.1;
  option domain-name "lab.local";
}
EOF'
sudo sed -i 's/INTERFACESv4=""/INTERFACESv4="eth0"/' /etc/default/isc-dhcp-server
sudo systemctl enable --now isc-dhcp-server
```

4.2 DNS (résolution des noms) via dnsmasq

```
sudo apt install -y dnsmasq
sudo bash -c 'cat > /etc/dnsmasq.d/lab.conf << "EOF"
interface=eth0
domain=lab.local
address=/srv.lab.local/10.10.10.1
listen-address=10.10.10.1
EOF'
sudo systemctl restart dnsmasq
```

4.3 HTTP/HTTPS (Nginx)

```
sudo apt install -y nginx
echo "Hello HTTP from GNS3" | sudo tee /var/www/html/index.html
sudo systemctl enable --now nginx
```

```
# Optionnel : activer TLS auto-signé pour HTTPS (certificat)
```

4.4 FTP (vsftpd)

```
sudo apt install -y vsftpd
sudo systemctl enable --now vsftpd
echo "fichier-ftp" | sudo tee /var/ftp/test.txt 2>/dev/null || true
```

4.5 SMTP/IMAP/POP3 (postfix + dovecot)

```
sudo apt install -y postfix dovecot-imapd dovecot-pop3d
sudo adduser etu --gecos "" --disabled-password
echo "etu:etu" | sudo chpasswd
sudo systemctl enable --now postfix dovecot
```

4.6 SNMP (snmpd)

```
sudo apt install -y snmp snmpd
sudo sed -i 's/^agentAddress.*/agentAddress udp:161,udp6:[::1]:161/' /etc/snmp/snmpd.conf
sudo sed -i 's/^#rocommunity public  default    -V systemonly/rocommunity public  default/' /etc/sn
sudo systemctl enable --now snmpd
```

4.7 Telnet (usage pédagogique, réseau isolé)

```
sudo apt install -y telnetd
sudo systemctl enable --now inetd || true
```

5) Génération de trafic & analyse Wireshark (Client■LINUX)

5.1 DHCP – obtenir une adresse IP

```
sudo dhclient -v eth0
```

Filtre Wireshark : **bootp || dhcp** — séquence Discover → Offer → Request → Ack.

5.2 DNS

```
dig @10.10.10.1 srv.lab.local
```

Filtre Wireshark : **dns** — Transaction ID, Questions, Answers (A).

5.3 HTTP

```
curl -i http://10.10.10.1/
```

Filtre Wireshark : **http** ou **tcp.port==80** — Request line, Status code, Headers.

5.4 FTP

```
ftp 10.10.10.1\n# utilisateur anonyme ou etu/etu
```

Filtre Wireshark : **ftp || tcp.port==21** — canal contrôle vs données (actif/passif).

5.5 SMTP / IMAP / POP3

```
# SMTP (telnet)
telnet 10.10.10.1 25
EHLO client.lab.local
MAIL FROM:<etu@lab.local>
RCPT TO:<etu@lab.local>
DATA
Sujet: test
Bonjour
.
QUIT

# IMAP (telnet)
telnet 10.10.10.1 143
a1 LOGIN etu etu
a2 LIST "" "*"
a3 SELECT INBOX
a4 FETCH 1 BODY[]
a5 LOGOUT

# POP3 (telnet)
telnet 10.10.10.1 110
USER etu
PASS etu
LIST
RETR 1
QUIT
```

Filtres : **smtp**, **imap**, **pop** — verbes, états, contenu.

5.6 SNMP

```
snmpwalk -v2c -c public 10.10.10.1 1.3.6.1.2.1.1
```

Filtre : **snmp** — GetRequest/GetResponse, OIDs et valeurs.

5.7 Telnet (non chiffré)

```
telnet 10.10.10.1 23\n# tapez un login fictif et observez le flux en clair dans Wireshark
```

Filtre : **telnet** — démontre les risques du clair ; comparer avec SSH (hors■scope).

6) Mini■scripts Python

HTTP (requests)

```
import requests
r = requests.get("http://10.10.10.1/", timeout=5)
print(r.status_code, r.headers.get("Server"), r.text[:80])
```

DNS (dnspython)

```
import dns.resolver
resolver = dns.resolver.Resolver()
resolver.nameservers = ["10.10.10.1"]
ans = resolver.resolve("srv.lab.local", "A")
for a in ans:
    print("A:", a.to_text())
```

SMTP (smtplib)

```
import smtplib
from email.mime.text import MIMEText
msg = MIMEText("Bonjour, ce message est un test SMTP depuis Python.")
msg["Subject"] = "TP Réseaux – Test SMTP"
msg["From"] = "etu@lab.local"
msg["To"] = "etu@lab.local"
with smtplib.SMTP("10.10.10.1", 25, timeout=5) as s:
    s.send_message(msg)
print("Message envoyé.")
```

Socket brut (TCP)

```
import socket
with socket.create_connection(("10.10.10.1", 80), timeout=5) as s:
    req = b"GET / HTTP/1.1\r\nHost: srv.lab.local\r\nConnection: close\r\n\r\n"
    s.sendall(req)
    data = s.recv(4096)
    print(data.decode(errors="ignore"))
```

7) Filtres Wireshark utiles

- • dns, http, ftp, smtp, imap, pop, snmp, bootp/dhcp, telnet
- • tcp.port==80, udp.port==53, tcp.port==21, tcp.port==25
- • ip.addr==10.10.10.1
- • Follow TCP stream (clic droit) pour reconstituer une session

8) Livrables attendus

- • Schéma de la topologie + plan d’adressage.
- • Pour chaque protocole : captures annotées, filtres utilisés, champs expliqués, chronologie.
- • Scripts Python et sorties console.
- • Conclusions sécurité (risques, mitigations, versions sécurisées).

9) Travail à réaliser

- • Mettre en place la topologie GNS3 proposée et vérifier la connectivité (ping).
- • Configurer sur le serveur : DHCP, DNS (dnsmasq), HTTP (Nginx), FTP (vsftpd), SMTP/IMAP/POP3 (postfix/dovecot), SNMP (snmpd), Telnet (réseau isolé).
- • Depuis le client, générer du trafic pour chaque protocole (commandes fournies) et capturer avec Wireshark.
- • Annoter au moins une capture par protocole : ports, messages/verbres clés, champs importants, chronologie.
- • Écrire et exécuter les mini■scripts Python (HTTP, DNS, SMTP, socket) ; inclure la sortie et expliquer le résultat.
- • Comparer un protocole en clair avec sa version chiffrée (ex. HTTP vs HTTPS) : impact dans Wireshark (payload lisible ou non).
- • Rédiger une analyse sécurité : risques observés (ex. mots de passe en clair), contre■mesures proposées (TLS, authentification, filtrage).
- • Assembler le dossier final (voir « Livrables attendus ») et vérifier la reproductibilité (README).

10) Grille d’évaluation (suggestion)

- • Mise en place & fonctionnement (GNS3/services) – 30 %
- • Qualité des captures & analyses (Wireshark) – 30 %
- • Scripts Python & résultats – 25 %
- • Clarté du rapport & conclusions – 15 %

11) Dépannage rapide

- • DHCP : interface service (INTERFACESv4), pare■feu, câblage GNS3.
- • DNS : dig @10.10.10.1, conf dnsmasq, écoute sur 10.10.10.1.
- • HTTP : curl -v, service nginx actif, port 80 ouvert.
- • FTP : mode passif/actif, ports data, vsftpd.conf.
- • Mail : logs /var/log/mail.log, dovecot actif.
- • SNMP : communauté public, port 161/UDP écouté.
- • Telnet : à utiliser uniquement en réseau isolé (risques de sécurité).