Laboratorul 5: Securitatea sistemelor vocale

Obiective

- Identificarea principalelor tipuri de atac asupra sistemelor vocale: replay, spoofing, atacuri adversariale asupra ASR.
- Înțelegerea modului în care ASR si TTS open-source modern crește suprafața de atac.
- Construirea unui set de date minimal și organizarea lui pentru experimente reproducibile.
- Implementarea unui detector de bază anti-replay (heuristic) și a unui clasificator anti-spoofing (ML).
- Evaluarea performanței prin Accuracy, ROC-AUC.
- Propunerea unui plan de mitigare pentru un sistem.

Partea I – Amenințări și model de risc

1. Tipuri de atac

- Replay: un atacator redă o înregistrare autentică a vocii utilizatorului pentru a păcăli sistemul.
- Spoofing: o voce este generată sintetic (TTS) sau convertită (voice conversion) pentru a semăna cu o identitate țintă.
- Atac adversarial asupra ASR: perturbații audio (adesea greu de sesizat) pot induce transcrieri greșite sau comenzi nedorite.
- Atacuri organizaționale: social engineering, colectare de mostre vocale, abuz de procese (confirmări telefonice, aprobări).

2. Sarcini: construirea setului de date (mini-dataset)

1. Alegeți o frază de 10–15 cuvinte (de ex.: o propoziție neutră, fără date personale).
2. Înregistrați 10 exemple LIVE (propria voce) folosind același microfon și o distanță constantă.
3. Creați 10 exemple REPLAY: redați fiecare fișier live pe un difuzor și înregistrați din nou (simulați un atac de redare).
4. Obțineți 10 exemple SINTETICE (TTS): folosiți voci generice / demo publice sau generați local cu un TTS. Evitați clonarea vocii altcuiva.
5. Organizați fișierele astfel: data/live/*.wav, data/replay/*.wav, data/synthetic/*.wav.
6. Creați un fișier metadata.csv cu: filename, label, text, device, distanță aproximativă, observații (zgomot, ecou).

Partea II – Detecție anti-replay și anti-spoofing

1. Instalare și încărcare audio

Instalați bibliotecile necesare:

pip install librosa numpy scipy matplotlib scikit-learn soundfile

Încărcarea unui fișier audio și normalizare:

```
import librosa
import numpy as np

y, sr = librosa.load('data/live/exemplu.wav', sr=None)
y = librosa.util.normalize(y)
print(y.shape, sr)
```

2. Detector de bază anti-replay: raport de energie în frecvențe înalte

Idee: semnalul „replay” trece printr-un lanț difuzor-aer-microfon, care poate modifica componentele înalte și introduce zgomot/rezonanțe. Un baseline simplu este să măsurăm ponderea energiei peste o frecvență prag (ex. 7 kHz pentru sr=16 kHz).

```
import librosa
import numpy as np

def hf_energy_ratio(y, sr, f0=7000, n_fft=2048, hop_length=512):
    S = np.abs(librosa.stft(y, n_fft=n_fft, hop_length=hop_length)) ** 2
    freqs = librosa.fft_frequencies(sr=sr, n_fft=n_fft)
    hf = S[freqs >= f0].sum()
    tot = S.sum() + 1e-12
    return float(hf / tot)
```

Sarcină: calculați acest scor pentru fiecare fișier LIVE și REPLAY și alegeți un prag (threshold) care separă cât mai bine cele două clase pe un set de validare.

3. Clasificator anti-spoofing (ML) pe trăsături acustice

Construiți un vector de trăsături pentru fiecare fișier. Un baseline robust include statistici (medie/deviație) ale MFCC și câteva trăsături spectrale simple (centroid, roll-off, flatness).

```
import librosa
import numpy as np

def extract_features(y, sr):
    y = librosa.util.normalize(y)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=20)
```

```
    mfcc_mean = mfcc.mean(axis=1)
    mfcc_std = mfcc.std(axis=1)

    centroid = librosa.feature.spectral_centroid(y=y, sr=sr).mean()
    rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr, roll_percent=0.85).mean()
    flatness = librosa.feature.spectral_flatness(y=y).mean()
    zcr = librosa.feature.zero_crossing_rate(y).mean()
    hf = hf_energy_ratio(y, sr)

    return np.concatenate([mfcc_mean, mfcc_std, [centroid, rolloff, flatness, zcr,
hf]]).astype(np.float32)
```

Antrenare și evaluare (binary: bonafide vs spoof):

```
import os, glob
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, roc_auc_score, confusion_matrix,
roc_curve

def load_dataset(root='data'):
    rows = []
    for label in ['live', 'replay', 'synthetic']:
        for fn in glob.glob(os.path.join(root, label, '*.wav')):
            rows.append({'path': fn, 'label': label})
    return pd.DataFrame(rows)

df = load_dataset('data')

X, y = [], []
for _, r in df.iterrows():
    y_audio, sr = librosa.load(r['path'], sr=None)
    X.append(extract_features(y_audio, sr))
    y.append(0 if r['label'] == 'live' else 1)  # 0=bonafide, 1=spoof

X = np.vstack(X)
y = np.array(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42, stratify=y)
```

```
clf = Pipeline([
    ('scaler', StandardScaler()),
    ('lr', LogisticRegression(max_iter=2000))
])
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
y_score = clf.predict_proba(X_test)[:, 1]

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred, digits=3))
print('ROC-AUC:', roc_auc_score(y_test, y_score))

# EER (opțional)
fpr, tpr, thr = roc_curve(y_test, y_score)
fnr = 1 - tpr
eer = fpr[np.nanargmin(np.abs(fnr - fpr))]
print('EER (aprox):', float(eer))
```

Sarcini de raportare

- Afișați spectrograme pentru câte un exemplu LIVE, REPLAY și SINTETIC. Notați diferențe observabile.
- Comparați detectorul heuristic (hf_energy_ratio + prag) cu clasificatorul ML (LogReg).
- Raportați metrici: Accuracy, Precision/Recall pe clasa spoof, ROC-AUC și (opțional) EER.
- Descrieți 2–3 cazuri unde modelul greșește și propuneți ipoteze (zgomot, microfon, text, lungime).

Partea III – Studiu de caz: modele TTS moderne și impactul asupra securității

1. De ce TTS open-source „modern” schimbă jocul

Modelele recente de sinteză pot produce vorbire expresivă și, în unele cazuri, pot controla timbrul și stilul suficient de bine încât să devină o amenințare pentru sistemele care se bazează exclusiv pe voce ca factor de autentificare.

Exemple (pentru documentare / analiză):

- Qwen3-TTS– serie open-source cu generare stabilă/expresivă, streaming și funcții de voice design / voice cloning.

- Orpheus-TTS (Canopy AI) – sistem open-source „Speech-LLM” bazat pe un backbone de tip Llama, cu capabilități de control al intonației/emoției și (în unele variante) clonare zero-shot.

2. Sarcină: test de generalizare

Adăugați în setul de test câteva mostre sintetice provenite dintr-o sursă diferită față de cea folosită la antrenare (de ex. un alt motor TTS sau mostre demo publice). Observați dacă detectorul ML rămâne robust sau „învață” doar un anumit tipar.

3. Contramăsuri recomandate în sisteme reale

- Liveness + challenge-response: solicitarea rostirii unei fraze aleatorii și verificarea coerenței temporale (dificil de „redat” perfect).
- Autentificare multi-factor (MFA): vocea trebuie să fie cel mult un factor auxiliar, nu unic.
- Detecție anti-spoofing dedicată: modele antrenate pe corpuri de date anti-spoofing (ex. seturi tip ASVspoof), actualizate periodic.
- Politici și procese: confirmări independente pentru acțiuni critice, limitarea tranzacțiilor prin voce, logare și audit.
- Trasabilitate / watermarking pentru conținut generat: marcarea și păstrarea provenienței pentru audio sintetic (acolo unde este disponibil).

Test Formativ 5 – Securitatea Sistemelor Vocale

Întrebări de reflecție

- De ce autentificarea vocală este mai greu de securizat decât parolele, chiar dacă putem antrena detectoare anti-spoofing?
- Care este compromisul între ușurința de utilizare (UX) și securitate într-un asistent vocal?
- Ce tip de date suplimentare (metadate, semnale multimodale, context) ar ajuta la reducerea atacurilor?

Secțiunea I – Definiții și Concepte

7. 1) Definiți, pe scurt, un atac de tip replay și un atac de tip spoofing în contextul autentificării vocale.
8. 2) De ce vocea este considerată o caracteristică biometrică dificil de „resetat” după compromitere?

Secțiunea II – Întrebări cu răspuns multiplu

3) O contramăsură de tip challenge-response are rolul de a:

a) crește volumul semnalului pentru a reduce zgomotul

b) solicita o frază/serie de cuvinte aleatorii pentru a verifica „liveness”

c) elimina complet nevoia de detecție anti-spoofing

d) comprima semnalul pentru streaming

4) Într-un sistem ASR, un atac adversarial urmărește în principal:

a) să reducă dimensiunea modelului

b) să introducă perturbații care schimbă transcrierea (fără a fi evident pentru utilizator)

c) să îmbunătățească acuratețea pe date zgomotoase

d) să mărească rata de eșantionare

5) Care practică reduce cel mai mult riscul într-un sistem critic controlat vocal?

a) parole fixe rostite de utilizator

b) vocea ca unic factor de autentificare

c) MFA + confirmare independentă pentru acțiuni critice + limitări de risc

d) dezactivarea completă a logării și auditului

Secțiunea III – Tehnici și evaluare

9. 6) Numiți două metrici folosite frecvent pentru evaluarea detecției spoofing/verificării vorbitorului (ex.: EER, FAR/FRR, ROC-AUC).
10. 7) De ce raportul de energie peste 7 kHz poate fi diferit între un semnal live și unul replay?

Secțiunea IV – Etică și bune practici

11. 8) Enumerați două reguli de utilizare responsabilă atunci când lucrați cu tehnologii de clonare vocală.
12. 9) De ce este important ca mostrele audio sintetice să fie etichetate și trasabile (provenance)?

Secțiunea V – Reflecție scurtă

10) Argumentați (3–5 fraze) de ce autentificarea vocală nu ar trebui să fie singurul mecanism de securitate într-un sistem critic.