

# CTF 연습 Platform

컴퓨터공학과 | 구지원

## CTF 연습 플랫폼

로그인 후 문제를 풀고, 점수를 얻어보세요.

[문제 보러가기](#)

# 목차

## Login

아이디

비밀번호

로그인

1

프로젝트 설명

2

flask - app.py

3

templates - html

4

create\_users.py

5

create\_challenge.py

# 프로젝트 설명



사용자가 문제를 풀고 플래그를 제출하면 점수가 기록되는 시스템

- 사용자가 웹페이지(html)에서 문제 확인 (로그인 후)
- 플래그 입력 후 제출 -> app.py로 전송
- 서버에서 정답 여부 확인
- 정답 시 DB에서 점수 반영
- scoreboard 페이지에 점수 업데이트

## Challenges

### Base64 Warmup

100 pts

[Crypto] Base64 Warmup  
아래 문자열을 Base64로 디코딩하면 플래그가 나옵니다.  
  
ZmxhZ3tNaW5pX0Jhc2U2NF9YXJtdXB9  
  
플래그 형식: flag{...}

제출

### [Crypto] ROT13

150 pts

다음 문장을 ROT13으로 복호화하세요.  
  
synt{ZvavVqragFhofgne}  
  
힌트: 알파벳 13자리씩 치환.

제출

### [Forensic] Hex to ASCII

150 pts

다음 16진수를 ASCII로 변환하면 플래그가 나옵니다.  
  
66 60 61 67 78 48 65 78 5F 52 65 61 64 7D

제출

# 프로젝트 설명

## Technique 1



---

프론트엔드 / 백엔드

## Technique 2



---

사용한 기술

# templates -html

## base.html

- 모든 html 파일에서 겹치는 부분들을 base.html 파일에 모아서 기본 페이지 생성

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>CTF Platform</title>
  <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
  <header>
    <h1>Mini CTF Platform</h1>
    <nav>
      <a href="{{ url_for('index') }}">Home</a>
      <a href="{{ url_for('login') }}">Login</a>
      <a href="{{ url_for('challenges') }}">Challenges</a>
      <a href="{{ url_for('scoreboard') }}">Scoreboard</a>
    </nav>
  </header>
  <main class="container">
    {% with messages=get_flashed_messages() %}
    {% if messages %}
    <ul class="flash">
      {% for m in messages %}<li>{{ m }}</li>{% endfor %}
    </ul>
    {% endif %}
    {% endif %}
    {% block content %}{% endblock %}
  </main>
  <footer class="site-footer">
    <small>© 2025 Mini CTF</small>
  </footer>
</body>
</html>
```

# templates -html

## index.html

- 버튼 클릭 시 첫 페이지에서 challenge 페이지로 넘어감.

## Mini CTF Platform

[Home](#) [Login](#) [Challenges](#) [Scoreboard](#)

### CTF 연습 플랫폼

로그인 후 문제를 풀고, 점수를 얻어보세요.

문제 보러가기

© 2025 Mini CTF

```
{% extends "base.html" %}
{% block static %}
<h2>CTF 연습 플랫폼</h2>
<p>로그인 후 문제를 풀고, 점수를 얻어보세요.</p>
<p>
  <a class="btn" href="{{ url_for('challenges') }}">문제 보러가기</a>
  {% if not session.get('user_id') %}
    <a class="btn outline" href="{{ url_for('login') }}">로그인</a>
  {% endif %}
</p>
{% endblock %}
```

# templates -html

## login.html

- 로그인 페이지에 form형식으로 html 만들기.

## Mini CTF Platform

[Home](#) [Login](#) [Challenges](#) [Scoreboard](#)

### Login

아이디

비밀번호

로그인

© 2025 Mini CTF

```
{% extends "base.html" %}
{% block content %}
<h2>Login</h2>
<form method="post" action={{ url_for('login') }} class="form">
    <label>아이디
    <input type="text" name="username" required>
    </label>
    <label>비밀번호
    <input type="password" name="password" required>
    </label>
    <button type="submit" class="btn">로그인</button>
</form>
{% endblock %}
```

# create\_users.py

## create\_users.py

- login.html 페이지 구현 후 -> 로그인 가능한 사용자 만들기
- 성공후 db 생성하기

```
from werkzeug.security import generate_password_hash
from app import app
from models import db, User

USERNAME = "user1"
PASSWORD = "pw1234!"

with app.app_context():
    if User.query.filter_by(username=USERNAME).first():
        print(f"[SKIP] 이미 존재: {USERNAME}")
    else:
        u = User(username=USERNAME, password=generate_password_hash(PASSWORD))
        db.session.add(u)
        db.session.commit()
        print(f"[OK] 사용자 생성: {USERNAME}")
```

```
(venv) C:\vscode\hopy-CTF\CTF>venv\Scripts\activate

(venv) C:\vscode\hopy-CTF\CTF>python create_users.py
[OK] 사용자 생성: user1

(venv) C:\vscode\hopy-CTF\CTF>
```



# templates -html

## challenge.html

- 문제 목록 페이지이고, 각 문제마다 제목/점수/설명을 보여줌.
- 로그인한 사용자만 제출 폼을 볼 수 있도록 함.

```
{% extends "base.html" %}
{% block content %}
<h2>Challenges</h2>

{% if challenges|length == 0 %}
<p>등록된 문제가 없습니다.</p>
{% endif %}

<ul class="challenge-list">
  {% for c in challenges %}
    <li class="challenge-item">
      <div class="challenge-head">
        <strong>{{ c.title }}</strong>
        <span class="points">{{ c.points }} pts</span>
      </div>
      <pre class="desc">{{ c.description }}</pre>

      {% if session.get('user_id') %}
        <form method="post" action="{{ url_for('submit', challenge_id=c.id) }}" class="submit-form">
          <input type="text" name="flag" placeholder="flag{...}" autocomplete="off" required>
          <button type="submit" class="btn">제출</button>
        </form>
      {% else %}
        <p class="muted">플래그 제출은 로그인 후 가능합니다.</p>
      {% endif %}
    </li>
  {% endfor %}
</ul>
{% endblock %}
```

# create\_challenge.py

## create\_challenge.py

- 문제 생성을 그냥 미리 해서 db에 저장 해둠
- -> python create\_challenge.py 터미널에 입력

```
from app import app
from models import db, Challenge

with app.app_context():
    if Challenge.query.count() == 0:
        problems = [
            Challenge(
                title="Base64 Warmup",
                description=(
                    "[Crypto] Base64 Warmup\n"
                    "아래 문자열을 Base64로 디코딩하면 플래그가 나옵니다.\n\n"
                    "ZmxhZ3tNaW5pX0Jhc2U2NF9XYXJtdXB9\n\n"
                    "플래그 형식: flag{...}"
                ),
                flag="flag{Mini_Base64_Warmup}",
                points=100
            ),
            Challenge(
                title="[Crypto] ROT13",
                description=(
                    "다음 문장을 ROT13으로 복호화하세요.\n\n"
                    "synt{ZvavVqragFhofgne}\n\n"
                    "힌트: 알파벳 13자리씩 치환."
                ),
                flag="flag{MingHiddenSubtar}",
                points=150
            ),
            Challenge(
                title="[Forensic] Hex to ASCII",
                description=(
                    "다음 16진수를 ASCII로 변환하면 플래그가 나옵니다.\n\n"
                    "66 6C 61 67 7B 48 65 78 5F 52 65 61 64 7D"
                ),
                flag="flag{Hex_Read}",
                points=150
            ),
        ]

        db.session.add_all(problems)
        db.session.commit()
        print("[OK] 예시 문제 3개 생성 완료")
    else:
        print("[SKIP] 이미 문제가 존재합니다.")
```

# templates -html

## challenge.html 실제 페이지

- 실행 했을 때 실제 문제 목록 페이지 입니다.

### Mini CTF Platform

[Home](#) [Login](#) [Challenges](#) [Scoreboard](#)

#### Challenges

##### Base64 Warmup

100 pts

[Crypto] Base64 Warmup  
아래 문자열을 Base64로 디코딩하면 플래그가 나옵니다.  
ZmxhZ3tNaW5pX0Jhc2U2NF9XYXJtdXB9  
플래그 형식: flag{...}

제출

##### [Crypto] ROT13

150 pts

다음 문장을 ROT13으로 복호화하세요.  
synt{ZvavVqragFhoifgne}  
힌트: 알파벳 13자리씩 치환.

제출

##### [Forensic] Hex to ASCII

150 pts

다음 16진수를 ASCII로 변환하면 플래그가 나옵니다.  
66 60 61 67 7B 48 65 78 5F 52 65 61 64 7D

제출

# templates -html

## scoreboard.html

- table을 이용해서 표를 만들어서 실행했을 때 순위, 사용자, 점수 를 표시할 때 표에 표시되도록 함.

```
{% extends "base.html" %}
{% block content %}
<h2>점수판</h2>
<table class="table">
  <thead>
    <tr>
      <th>순위</th>
      <th>사용자</th>
      <th>점수</th>
    </tr>
  </thead>
  <tbody>
    {% for u in users %}
    <tr>
      <td>{{ loop.index }}</td>
      <td>{{ u.username }}</td>
      <td>{{ u.score }}</td>
    </tr>
    {% endfor %}
  </tbody>
</table>

{% endblock %}
```

# templates -html

## scoreboard.html

- 실행 했을 때 실제 스코어보드 페이지 입니다.

### Mini CTF Platform

[Home](#) [Login](#) [Challenges](#) [Scoreboard](#)

#### 점수판

순위	사용자	점수
1	user1	0

# flask - app.py

## def login():

- db에 저장한 아이디, 비밀번호를 맞게 입력 시 페이지 접속(GET 요청)

```
from flask import Flask, render_template, request, redirect, url_for, flash, session
from werkzeug.security import generate_password_hash, check_password_hash
from models import db, User, Challenge, Submission

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///ctf.db'
app.config['SECRET_KEY'] = 'secret_key_here'
app.secret_key = "dev" # flash 사용 시 필요

db.init_app(app) # DB 초기화 연결

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    # POST
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username).first()

        if user and check_password_hash(user.password, password):
            session['user_id'] = user.id
            flash('로그인 성공')
            return redirect(url_for('challenges'))
        else:
            flash('아이디 또는 비밀번호가 올바르지 않습니다.')
            return render_template('user.html')
    # GET
    return render_template('login.html')
```

# flask - app.py

## def challenges():

- db에 저장한 문제 전체 출력하기

```
@app.route('/challenges')
def challenges():
    challenges = Challenge.query.all()
    return render_template('challenge.html', challenges=challenges)

# 템플릿의 url_for('submit', challenge_id=...) 와 정확히 일치하는 라우트
@app.route('/submit/<int:challenge_id>', methods=['POST'])
def submit(challenge_id):
    if 'user_id' not in session:
        flash('로그인이 필요합니다.')
        return redirect(url_for('login'))

    raw = request.form.get('flag', '')
    submitted = normalize(raw)
    chal = Challenge.query.get_or_404(challenge_id)
    answer = normalize(chal.flag)

    user = User.query.get(session['user_id'])
    if user is None:
        flash('세션 사용자 정보를 찾을 수 없습니다.')
        return redirect(url_for('login'))

    # 이미 정답 처리했는지 확인
    already = Submission.query.filter_by(
        user_id=user.id, challenge_id=chal.id, correct=True
    ).first()

    is_correct = (submitted == answer) # 완전 일치(정규화 후). 대소문자 무시하려면 .lower()로 비교.
```

# flask - app.py

**def submit(challenge\_id):**  
**--> (앞과 연결)**

- 로그인 여부 체크 -> 제출한 답과 db의 정답을 비교
- 맞다면 점수를 정립하고 스코어보드로 이동
- 틀리면 다시 챌린지 페이지로 이동

```
# 제출 기록 저장(정답/오답 모두)
sub = Submission(
    user_id=user.id,
    challenge_id=chal.id,
    submitted_flag=raw, # 원문 그대로 보관
    correct=is_correct
)
db.session.add(sub)

if is_correct:
    if already:
        flash('이미 푼 문제입니다. 점수는 한 번만 적립됩니다.')
        db.session.commit()
        return redirect(url_for('challenges'))
    # 최초 정답 → 점수 적립
    user.score = (user.score or 0) + (chal.points or 0)
    db.session.commit()
    flash(f'정답입니다! +{chal.points}점 적립')
    return redirect(url_for('scoreboard'))
else:
    db.session.commit()
    # 디버깅용: 제출값과 정답을 살짝 보여 줌(개발 중에만)
    flash(f'오답입니다. 제출="{submitted}" / 정답="{answer}"')
    return redirect(url_for('challenges'))
```



# flask - app.py

## def scoreboard():

- users의 스코어보드 출력
- 밑에는 그냥 flask에서 app.py를 실행시키기 위해 필요한 내용

```
@app.route('/scoreboard')
def scoreboard():
    users = User.query.order_by(User.score.desc()).all()
    return render_template('scoreboard.html', users=users)

if __name__ == '__main__':
    with app.app_context():
        db.create_all()
        app.run(debug=True)
```

# Thank you!