웹 취약점 검사 시스템

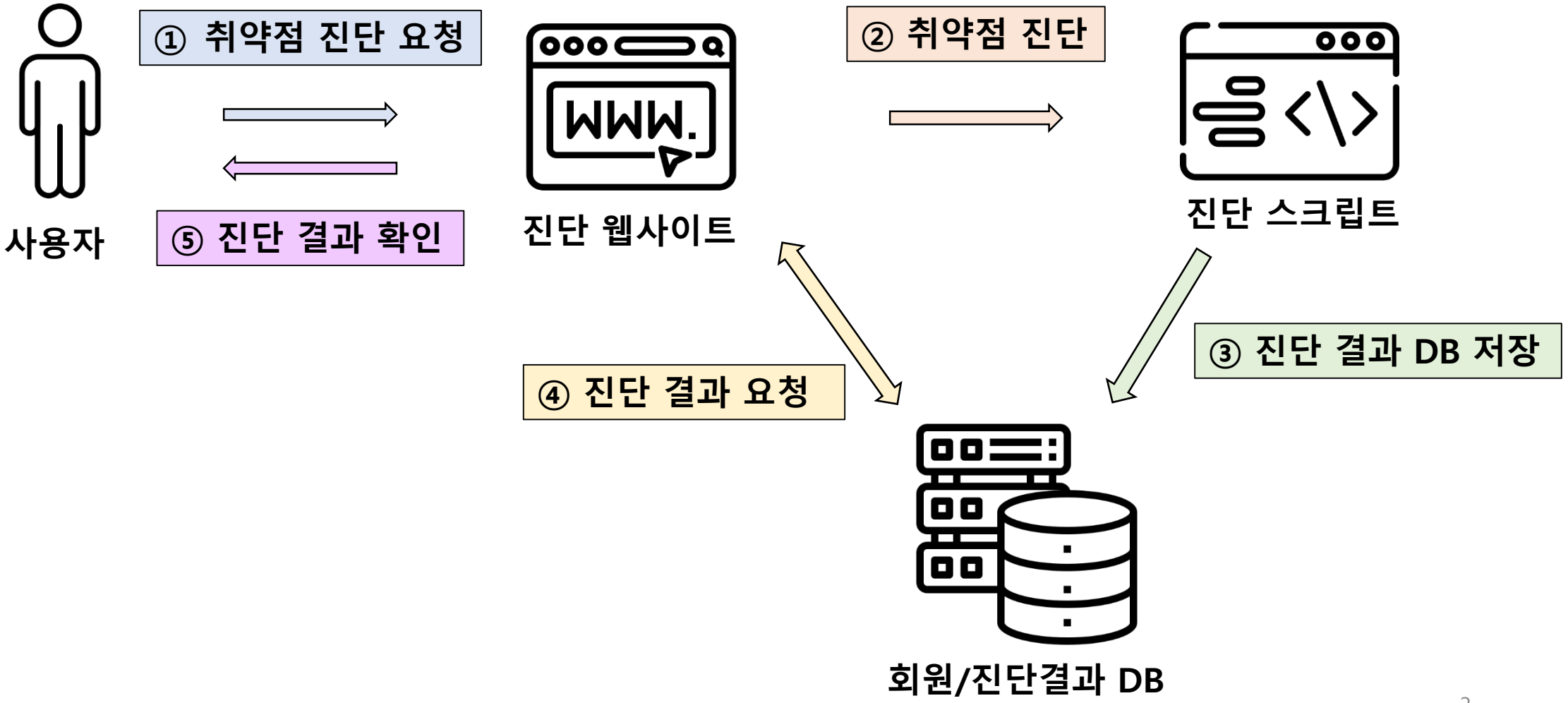
지도교수 : 양환석 교수님

코 웹 : 박시원 팀장
김난희
홍지우
윤해정
홍유진

목 차

- 구상도
- 개발 환경 및 개발 내용
- 개발 시스템 운영
- 결론 및 기대 효과

구상도


① 취약점 진단 요청
⑤ 진단 결과 확인
사용자
진단 웹사이트
② 취약점 진단
진단 스크립트
③ 진단 결과 DB 저장
④ 진단 결과 요청
회원/진단결과 DB

개발 환경 및 개발 내용(1/5)

WEB Server

DB

Front-End

진단 스크립트

개발 환경 및 개발 내용(2/5)

진단 스크립트

XSS

```
def XSS(url):
    target_url, method = getForm(url)

    insert_data = { "query" : "<script>alert('hi')</script>"}
    if method == "post":
        res=requests.post(target_url, data=insert_data).text
    else :
        res=requests.get(target_url, params=insert_data).text


    if  "<script>alert('hi')</script>" in res:
        result_list['XSS']='위험'
        result_list['XSS_info']=target_url
    else :
        result_list['XSS']='안전'
        result_list['XSS_info']=target_url
```

SQL Injection

```
def sqlInjection(url):
    target_url =url+"/doLogin"

    login_info ={
        "uid":"' or 1=1--+",
        "passw":"1234",
        "btnSubmit":"Login"
    }
    content=requests.post(target_url, data=login_info).text

    if "Welcome" in content:
        result_list['sqlin']='위험'
        result_list['sql_info']=target_url

    else:
        result_list['sqlin']='안전'
        result_list['sql_info']=target_url
```

개발 환경 및 개발 내용(3/5)

진단 스크립트

CSRF

```
def CSRF(url):
    target_url =url+"/doLogin"
    session = requests.session()
    login_info ={
        "uid":"' or 1=1--+",
        "passw":"1234",
        "btnSubmit":"Login"
    }
    res=session.post(target_url, data=login_info)
    target_url = url+"/sendFeedback"

    send_info ={
        "cfile":"comments.txt",
        "name": "<body onload='csrf.submit();'><form name='csrf'action='http://altoromutual.com:8080/logout.jsp'</form>",
        "email_addr":"1@gmail.com",
        "subject":"2",
        "comments": "3",
        "submit":"Submit"
    }
```

Brute Force

```
def bruteForce(url):
    target_url =url+"/doLogin"
    session = requests.session()

    password_list = open(os.getcwd()+"\password.txt", "r")
    password = password_list.readlines()

    for psword in password:
        passwordd = psword.strip()

        login_info ={
            "uid":"admin",
            "passw": passwordd,
            "btnSubmit":"Login"
        }

        content=session.post(target_url, data=login_info).text
```

개발 환경 및 개발 내용(4/5)

진단 스크립트

쿠키 취약점

```python
def cookie(url):
    target_url =url+"/doLogin"
    session = requests.session()

    login_info ={
    "uid":"' or 1=1--+",
    "passw":"1234",
    "btnSubmit":"Login"
    }
    session.post(target_url, data=login_info)
    cookiejar = session.cookies

    for cookie in cookiejar:
        if cookie.secure == False :
            result_cookie_secure= False
        else :
            result_cookie_secure= True

        if cookie.expires == None :
            result_cookie_expires= False
        else :
            result_cookie_expires= True
```

세션 고정 취약점

```python
def session_fixation(url):
    target_url =url+"/doLogin"
    with requests.Session() as s:

        res = s.get(target_url, data=login_info)
        cookie1 = res.headers['Set-Cookie']

    with requests.Session() as s:
        res = s.get(target_url, data=login_info)

        cookie2 = res.headers['Set-Cookie']
```

리다이렉트 취약점

```python
def redirect(url):
    target_url =url+"/doLogin"
    r = requests.get(target_url)
    try:
        session2 = r.headers['Strict-Transport-Security']
```

개발 환경 및 개발 내용(5/5)

진단 스크립트

민감한 데이터 노출

```
result_list['base64Vul'] = ''
for i in range(len(cookieValue)):
    data_bin=base64.b64decode(cookieValue[i])
    if isinstance(data_bin, str ):
        result_list['base64Vul'] = result_list['base64Vul'] + data_bin
    else :
        result_list['base64Vul'] = result_list['base64Vul'] + str(data_bin)
```

평문 전송 취약점

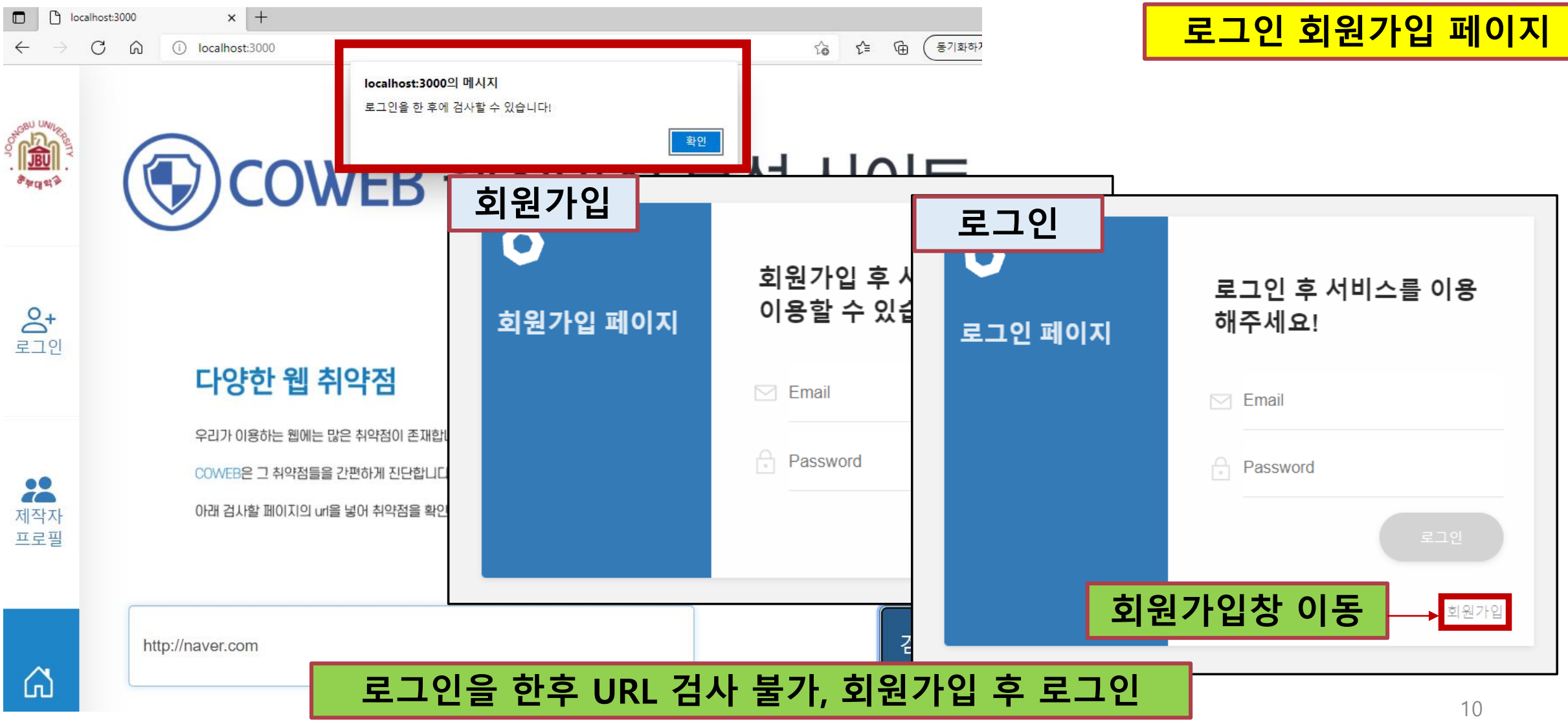
```
def sniffing():
    sniff(lfilter=showPacket, timeout=6)

temp=0
def showPacket(packet):
    scapy.all.load_layer("http")
    target_url =url+"/doLogin"
    login_info ={
        "uid":"admin",
        "passw":"admin",
        "btnSubmit":"Login"
    }
    requests.post(target_url, data=login_info)
    pk = raw(packet)
    strpk = str(pk)
```

개발 시스템 운영(1/4)


메인 페이지
- 중부대학교 정보보호학과 -
COWEB 웹취약점 분석 사이트
프로젝트 소개
로그인
제작자 프로필
다양한 웹 취약점
우리가 이용하는 웹에는 많은 취약점이 존재합니다.
COWEB은 그 취약점들을 간편하게 진단합니다.
아래 검사할 페이지의 url을 넣어 취약점을 확인해보세요.
URL 입력란
URL Address
제출

개발 시스템 운영(2/4)


로그인 회원가입 페이지
localhost:3000
localhost:3000의 메시지
로그인을 한 후에 검사할 수 있습니다!
확인
COWEB
로그인
제작자 프로필
다양한 웹 취약점
우리가 이용하는 웹에는 많은 취약점이 존재합
COWEB은 그 취약점들을 간편하게 진단합니다
아래 검사할 페이지의 url을 넣어 취약점을 확인
http://naver.com
회원가입
회원가입 페이지
회원가입 후 서
이용할 수 있습
Email
Password
로그인
로그인 페이지
로그인 후 서비스를 이용
해주세요!
Email
Password
로그인
회원가입창 이동
회원가입
로그인을 한후 URL 검사 불가, 회원가입 후 로그인

개발 시스템 운영(3/4)

진단 목록

진단 결과

진단 결과

		결과	정보
1	XSS	위험	http://altoromutual.com:8080/search.jsp
2	SQL INJECTION	위험	http://altoromutual.com:8080/doLogin
3	CSRF	위험	http://altoromutual.com:8080/sendFeedback
4	BruteForce	위험	http://altoromutual.com:8080/doLogin
5	Plaintext	위험	Https프로토콜필요
6	Cookie	위험	secure,expire 설정필요
7	Sniffing	위험	
8	Session	위험	
9	Local Encryption		b'800000~Corporate~5.239478361E7|800001~Checking~93820.44|800002~Savings~10005.42|800003~Checking~14999.39|800004~Savings~10.0|800005~Checking~25.0|800006~Savi ng~150.0|4539082039396288~Credit Card~109.92|4485983356242217~Credit Card~10000.97|'b'₩xec D₩x13a₩x02₩x03₩xb0;₩xf7₩xde}₩xec₩x0ev₩x07A}₩xd8₩x10;₩xf00₩xf4'

본인의 진단 목록에서 검사 결과 기록을 볼 수 있다.

개발 시스템 운영(4/4)

설명 페이지

세션 취약점

[불충분한 세션 관리 동작 과정]

세션 취약점이란

웹 애플리케이션에서 사용자가 로그인할 경우 매번 같은 세션 ID(일정한 패턴이 존재)를 발급하거나
세션 타임아웃을 너무 길게 설정하였을 경우 공격자가 다른 사용자의 세션을 재사용하여
세션 값을 변조하여 로그인을 시도해 다중 로그인을 허용하거나
해당 사용자의 권한을 탈취 할 수 있는 취약점입니다.

타 사용자가 로그인에 성공하도록 하는 경우 불충분한 세션 관리 취약점으로 판단 할 수 있습니다.
로그인과 로그아웃을 반복하여 수집된 세션 ID의 패턴을 분석하여
세션 타임아웃을 설정하도록 시큐어 코딩을 적용 해야 합니다.
웹 방화벽으로는 대응할 수 없으며 애플리케이션 코딩 시 시큐어 코딩을 적용하여 중복 로그인을 방지하거나

각 항목을 클릭하면 해당 취약점에 대한 설명을 볼 수 있다.

결론 및 기대 효과

◆ 결론

- 주요 웹 취약점에 대하여 진단 결과 제공

◆ 기대효과

- 진단 결과를 바탕으로 취약점 보완을 통해 안전한 웹 서비스 제공

Q & A

감사합니다