

수업 내용 정리 보고서

작성일: 2026년 7월 23일

챕터 구성

- DOM-based XSS의 원리와 브라우저 보안 메커니즘
- iptables를 이용한 리눅스 네트워크 방화벽 운영
- 웹 및 네트워크 보안 설정 실무 점검

DOM-based XSS의 원리와 브라우저 보안 메커니즘

DOM 기반 XSS(DOM-based Cross-Site Scripting)는 웹 애플리케이션 보안에서 매우 중요한 위치를 차지합니다. 이 공격은 서버 측의 데이터 처리 과정과는 무관하게, 클라이언트(웹 브라우저)에서 실행되는 자바스크립트가 사용자 입력값을 적절히 검증하지 않고 문서 객체 모델(DOM)에 반영할 때 발생합니다.

DOM-based XSS의 기본 원리

DOM(Document Object Model)은 브라우저가 HTML 문서를 이해하고 조작하기 위해 내부적으로 구축하는 트리 형태의 구조도입니다. 공격자는 이를 악용하여 브라우저의 자바스크립트가 실행되는 시점에 악성 스크립트를 삽입합니다.

일반적으로 서버를 거치지 않고 브라우저 환경에서만 처리가 완결되기 때문에, 서버 측 방화벽(WAF)이 요청 파라미터를 검사하더라도 우회할 가능성이 높습니다. 예를 들어, URL 파라미터 값에 `iframe` 태그를 포함하여 공격할 경우, 클라이언트 측의 자바스크립트가 `innerHTML`이나 `document.write()`를 통해 이 값을 렌더링하면 공격자의 코드가 실행됩니다.

공격자는 단순히 스크립트를 실행하는 것에 그치지 않고, `iframe` 태그를 활용하여 외부 악성 스크립트를 로드하거나, 정교하게 꾸며진 피싱 페이지를 기존 페이지 위에 덮어씌워 사용자 자격 증명을 탈취하는 등의 실질적인 피해를 유발합니다.

브라우저 보안 메커니즘과 공격의 한계

과거에는 DOM XSS가 강력한 위협이었으나, 현대 브라우저는 이를 방어하기 위한 겹겹의 보안 정책을 적용하고 있습니다. 가장 대표적인 것이 혼합된 콘텐츠(Mixed Content) 차단 정책입니다.

- **Mixed Content 정책:** HTTPS로 서비스되는 안전한 페이지에서 HTTP 통신을 사용하는 활성 콘텐츠(`iframe`, `script` 등)를 로드하려고 하면 브라우저가 이를 즉시 차단합니다. 이는 해커 서버가 HTTPS 인증서를 갖추지 못한 경우 공격을 원천 봉쇄하는 효과가 있습니다.
- **콘텐츠 보안 정책(CSP):** 서버가 브라우저에게 "우리 사이트에서는 허락된 도메인의 `iframe`만 허용하라"는 지시를 내리는 정책입니다. 해커가 HTTPS를 사용하여 Mixed Content 정책을 우회하더라도, CSP 헤더에 등록되지 않은 도메인이라면 브라우저는 렌더링을 거부합니다.
- **SameSite 쿠키 정책:** 타사 도메인에서 온 요청에 대해 쿠키 전송을 제한하여, 공격자가 `iframe` 내부에서 타겟 사이트의 세션 정보를 가로채는 것을 방지합니다.

보안 대응 전략: 시큐어 코딩

WAF를 통한 단어 필터링은 해시 기호를 활용한 우회 공격에 취약할 수 있으므로, 가장 근본적인 대책은 프론트엔드 개발 단계에서의 시큐어 코딩입니다. `innerHTML`이나 `document.write()`와 같이 입력값을 HTML로 해석하는 위험한 함수 사용을 지양해야 합니다.

대신, `innerText`나 `textContent`와 같은 함수를 사용하면 브라우저는 입력값을 단순히 텍스트로 처리합니다. 이 경우 해커가 `<iframe>` 태그를 삽입하더라도, 브라우저는 이를 HTML 요소가 아닌 글자 그대로 출력할 뿐이므로 공격은 실패하게 됩니다.

실무에서는 이러한 기술적 방어와 더불어, CSP를 적절히 설정하여 서비스에 필요한 외부 리소스(유튜브 영상, 결제 모듈 등)만 화이트리스트로 관리하는 것이 필수적입니다. 설정 파일(Nginx의 `add_header Content-Security-Policy` 등)을 통해 보안 지시서를 명확히 하되, 설정 누락으로 인한 서비스 장애가 발생하지 않도록 주의가 필요합니다.

학습 체크포인트

- **오해 주의:** 해커도 HTTPS를 쓰면 DOM XSS가 가능하다고 생각하기 쉽지만, CSP나 SameSite 쿠키 정책 등 현대 브라우저의 다중 방어 체계로 인해 실제 공격 성공률은 매우 낮습니다.
- **차단의 주체:** 서버가 아니라 피해자의 브라우저가 보안 정책을 확인하고 차단한다는 점을 이해해야 합니다.
- **점검 기준:** 웹 애플리케이션이 `innerHTML` 등을 사용할 때 사용자 입력값이 충분히 인코딩되는지, CSP 헤더가 누락되지 않았는지 정기적으로 검토해야 합니다.

iptables를 이용한 리눅스 네트워크 방화벽 운영

리눅스 시스템에서 네트워크 트래픽을 제어하고 보안 정책을 수립하는 것은 시스템 관리의 핵심입니다. 리눅스는 커널 수준에서 패킷 필터링 프레임워크인 **Netfilter**를 제공하며, 사용자는 **iptables** 명령어를 통해 이 프레임워크에 규칙을 등록하고 관리합니다.

방화벽의 핵심 구성 요소: Table, Chain, Rule

iptables는 세 가지 계층 구조를 통해 트래픽을 처리합니다. 각 계층은 패킷 처리의 목적, 시점, 동작을 명확히 구분합니다.

- **Table(테이블):** 무슨 일을 할 것인가를 결정합니다. 패킷 필터링(filter), 주소 변환(nat), 패킷 헤더 수정(mangle) 등 목적별 작업 공간을 제공합니다.
- **Chain(체인):** 언제 처리할 것인가를 결정합니다. 패킷이 시스템에 들어올 때(INPUT), 나갈 때(OUTPUT), 혹은 다른 네트워크로 전달될 때(FORWARD) 등의 처리 시점입니다.
- **Rule(규칙):** 어떻게 처리할 것인가를 정의합니다. 특정 조건(IP, 포트, 프로토콜)에 일치하는 패킷을 허용(ACCEPT), 차단(DROP), 거부(REJECT)합니다.

위의 구조도와 같이 iptables 명령은 `-A`(규칙 추가), `-p`(프로토콜 지정), `--dport`(목적지 포트), `-j`(동작 대상) 등의 옵션을 조합하여 구성됩니다. 패킷은 등록된 규칙을 위에서부터 순차적으로 검사하며, 첫 번째로 일치하는 규칙의 동작(Target)이 즉시 수행됩니다.

실습을 통한 트래픽 제어 및 운영

iptables 운영 시 가장 중요한 원칙은 '허용(ACCEPT)을 먼저, 차단(DROP/REJECT)을 나중에' 설정하는 것입니다. 이는 순차적인 규칙 검사 방식 때문입니다.

1. 포트 허용 및 기본 정책 설정

서버 서비스 운영을 위해 필요한 포트만 개방하고 나머지 트래픽은 모두 차단하는 것이 일반적인 보안 설정입니다.

```
# 기존 연결 허용
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
# HTTP(80) 및 웹 서비스(8080) 포트 허용
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 8080 -j ACCEPT
# 나머지 모든 트래픽 차단
iptables -P INPUT DROP
```

위와 같이 설정하면 지정된 포트 외의 접속 시도는 타임아웃이 발생하여 패킷이 정상적으로 차단됨을 확인할 수 있습니다.

2. SSH 접근 통제

특정 관리자 IP에서만 SSH 접속을 허용하도록 제한하여 무차별 대입 공격으로부터 시스템을 보호합니다.

```
# 특정 IP(예: 172.16.x.x)의 SSH 접속 허용
iptables -A INPUT -s 172.16.x.x -p tcp --dport 22 -j ACCEPT
# 그 외 모든 SSH 접속 차단
iptables -A INPUT -p tcp --dport 22 -j DROP
```

운영 주의사항 및 트러블슈팅

운영 중 규칙을 잘못 설정하면 서버 관리자가 스스로 접속을 차단하는 상황이 발생할 수 있습니다. 따라서 작업 전 반드시 다음 점검 사항을 확인해야 합니다.

- **DROP vs REJECT:** 외부 공개 서버는 포트 존재 여부를 숨기기 위해 DROP을, 내부 개발 환경은 거부 응답을 즉시 알 수 있는 REJECT를 주로 사용합니다.
- **현행 상태 확인:** `iptables -L -n -v` 명령을 사용하여 현재 적용된 규칙과 패킷 카운터를 주기적으로 점검합니다.
- **초기화 절차:** 실수로 규칙이 꼬였을 경우 전체 초기화(Flush)가 필요합니다. `iptables -F`로 규칙을 삭제하고 정책을 다시 ACCEPT 상태로 되돌립니다.

방화벽 운영의 완성은 규칙을 등록한 후, 의도대로 트래픽이 허용되거나 차단되는지 실제 접속 테스트(PASS/BLOCK)를 거치는 것입니다. 위 화면에서 확인되는 카운터 변화를 통해 실제 패킷이 어떤 규칙에 일치하는지 추적하는 것이 실무적인 트러블슈팅의 시작입니다.

웹 및 네트워크 보안 설정 실무 점검

현대 웹 환경에서 보안은 단순히 서버를 보호하는 것을 넘어, 브라우저의 보안 정책과 네트워크 제어 계층이 유기적으로 동작할 때 완성됩니다. 이번 실무 점검에서는 클라이언트 측 취약점인 DOM XSS와 리눅스 커널 수준의 네트워크 방화벽인 iptables를 중심으로, 보안 정책이 어떻게 설계되고 실행되는지 확인합니다.

1. 클라이언트 보안의 중심: DOM XSS 점검

DOM(Document Object Model) XSS는 서버의 개입 없이 피해자의 브라우저 내에서 자바스크립트가 안전하지 않은 방식으로 사용자 입력값을 DOM에 반영할 때 발생합니다. 공격자는 주로 URL 파라미터를 통해 악성 스크립트를 삽입하며, 이때 iframe 태그를 사용하여 피싱 페이지를 덮어씌우거나 악성 코드를 로드하는 방식을 취합니다.

최신 웹 환경에서는 HTTPS를 사용하는 타겟 사이트에서 HTTP 기반의 외부 리소스를 호출하려 할 때, 브라우저가 이를 혼합된 콘텐츠(Mixed Content)로 간주하여 차단합니다. 이는 브라우저가 Active Mixed Content(iframe, script 등)를 즉시 로드 차단하여 공격을 무력화하는 효과를 냅니다.

성공적인 방어를 위해 개발자는 innerHTML 대신 textContent나 innerText를 사용하여 데이터를 단순 텍스트로 처리해야 하며, 서버 측에서는 CSP(Content Security Policy)를 통해 허용된 도메인의 자원만 로드하도록 브라우저에 지시를 내려야 합니다.

2. 네트워크 계층의 방어막: iptables 실무

네트워크 보안의 기본은 패킷의 흐름을 제어하는 것입니다. 리눅스 환경에서는 Netfilter가 실제 방화벽 엔진 역할을 수행하며, iptables는 이 방화벽 규칙을 관리하는 도구입니다. iptables는 Table(목적), Chain(시점), Rule(동작)의 구조로 패킷을 처리합니다.

- **Table:** filter(패킷 허용/차단), nat(주소 변환), mangle(헤더 수정) 등 목적별 공간.
- **Chain:** INPUT, OUTPUT, FORWARD 등 패킷이 처리되는 시점.
- **Rule:** 조건에 일치하는 패킷을 ACCEPT, DROP, REJECT하는 실제 정책.

운영 환경에서는 서비스 가용성과 보안 사이의 균형이 중요합니다. 외부 공개 서버는 타임아웃을 유도하는 DROP 정책을 선호하며, 내부 개발 환경은 거부 응답을 보내는 REJECT를 사용하기도 합니다. 특히 SSH 등 관리용 포트는 특정 IP 대역만 허용하도록 규칙 순서를 엄격히 제어해야 합니다.

3. 실무 점검 체크리스트 및 흔한 오해

보안 정책을 점검할 때 다음의 포인트들을 확인하십시오:

- **CSP(Content Security Policy) 확인:** 서버 응답 헤더에 `frame-src 'self'` 등 명확한 허용 리스트가 포함되어 있는가?
- **규칙 순서(Order of Rules):** iptables 규칙은 위에서 아래로 검사되므로, 허용(ACCEPT) 규칙이 차단(DROP) 규칙보다 위에 위치하는가?
- **상태 기반 검사(Stateful Inspection):** ESTABLISHED, RELATED 상태의 패킷을 먼저 허용하여 불필요한 트래픽 검사를 줄이고 있는가?
- **SOP(Same-Origin Policy) 이해:** 도메인이 다르면 쿠키 접근이 불가능함을 인지하고, 피싱 공격 시나리오를 방어하고 있는가?

주의할 점은, 서버 방화벽(WAF나 iptables)만으로 모든 공격을 막을 수 없다는 것입니다. DOM XSS와 같이 브라우저 안에서 발생하는 문제는 근본적으로 프론트 엔드 코드의 안전성(시큐어 코딩)과 CSP와 같은 브라우저 보안 지침이 함께 결합되어야 실질적인 방어가 가능합니다.

마지막으로 규칙을 변경한 후에는 반드시 `iptables -L -n -v` 명령어를 사용하여 현재 적용된 규칙의 카운터 상태와 순서를 검증해야 합니다. 운영 중인 서버에서 규칙을 초기화(-F)하거나 전체 정책을 변경할 때는 원격 접속이 차단되지 않도록 사전에 주의를 기울여야 합니다.