

수업 내용 정리 보고서

작성일: 2026년 7월 21일

챕터 구성

- 시스템 및 네트워크 제어 메커니즘
- 웹 취약점 분석 및 XSS 실습
- 모의해킹 실무 점검 체크리스트

시스템 및 네트워크 제어 메커니즘

시스템과 네트워크가 안정적으로 동작하고 데이터를 정확하게 전달하기 위해서는 자원을 효율적으로 배분하고, 데이터의 변조를 방지하며, 통신 흐름을 제어하는 다양한 메커니즘이 필요합니다. 이번 장에서는 시스템의 부하를 줄이는 기술부터 네트워크 트래픽의 우선순위를 결정하는 제어 방식, 그리고 이러한 제어 메커니즘의 허점을 이용한 공격 시나리오를 분석합니다.

1. 시스템 자원 최적화 및 데이터 무결성

시스템의 성능을 극대화하기 위해서는 CPU가 모든 작업을 처리하게 두는 대신, 특정 작업을 전용 하드웨어로 분산시키는 전략이 필요합니다.

오프로딩(Offloading)

배경 및 정의: 컴퓨터 시스템에서 CPU(중앙 처리 장치)는 매우 복잡하고 다양한 연산을 수행해야 합니다. 이때 네트워크 패킷 처리나 암호화와 같이 반복적이고 특화된 작업을 CPU가 직접 수행하면 전체적인 시스템 성능이 저하될 수 있습니다. **오프로딩(Offloading)**은 이러한 특정 작업을 CPU 대신 네트워크 카드(NIC)나 전용 가속기 같은 별도의 하드웨어에 할당하여 CPU의 부하를 줄이는 기술을 의미합니다.

- 작동 원리:** CPU가 네트워크 인터페이스 카드(NIC)에 "이 패킷은 체크섬을 계산해서 보내줘"라고 명령을 내리면, NIC 내부의 하드웨어가 이를 대신 처리하여 CPU는 다른 중요한 프로세스에 집중할 수 있게 합니다.
- 실무 포인트:** 시스템 성능 모니터링 시 CPU 사용량은 낮지만 네트워크 성능이 떨어진다면, 오프로딩 설정이 잘못되어 CPU가 예상치 못한 부하를 받고 있는지 확인해야 합니다.

데이터 구조와 무결성 검증(Checksum)

데이터가 전송되거나 메모리에 저장될 때, 정해진 규칙에 따라 구조화되어야 하며 전송 과정에서 변조되지 않았음을 보장해야 합니다.

- 구조체(Structure)의 크기:** 시스템 설계 시 메모리에 할당되는 데이터의 크기는 매우 중요합니다. 예를 들어, 특정 통신 프로토콜에서 **5kb 규모의 구조체(Structure)**를 사용하도록 설계되었다면, 이 크기를 초과하는 데이터가 들어올 경우 버퍼 오버플로우(Buffer Overflow)와 같은 메모리 오염 취약점이 발생할 수 있습니다.
- 체크섬(Checksum)의 역할:** 데이터의 무결성을 확인하기 위해 데이터의 합계 값을 계산하여 데이터 끝에 붙이는 방식입니다. 만약 **체크섬 계산이 미완성**되었거나 구현이 허술하다면, 공격자가 중간에서 데이터를 변조하더라도 시스템이 이를 정상적인 데이터로 오인하여 처리할 위험이 있습니다.

주의사항: 데이터 구조의 크기와 체크섬 검증 로직은 시스템 보안의 기초입니다. 설계 단계에서 경계값 검사(Boundary Check)와 무결성 검증이 완벽하지 않으면, 이는 곧 시스템 권한 탈취로 이어지는 공격 통로가 됩니다.

2. 네트워크 트래픽 및 흐름 제어

네트워크 상에서 데이터가 충돌 없이, 그리고 중요한 데이터가 제때 전달될 수 있도록 제어하는 메커니즘이 필수적입니다.

서비스 품질(QoS, Quality of Service)

정의: 네트워크 자원이 한정되어 있을 때, 특정 트래픽에 우선순위를 부여하여 서비스 수준을 보장하는 기술입니다. 예를 들어, 실시간 영상 통화(VoIP) 데이터는 웹 서핑 데이터보다 지연(Latency)에 훨씬 민감하므로 더 높은 우선순위를 부여받아야 합니다.

- 적용 조건:** 대역폭(Bandwidth)이 부족하거나 네트워크 혼잡(Congestion)이 발생할 때 QoS 정책이 활성화되어 중요 트래픽을 보호합니다.
- 판단 기준:** 특정 서비스의 지연 시간이 급증한다면, QoS 정책이 해당 트래픽을 낮은 우선순위로 분류하고 있지는 않은지 점검해야 합니다.

슬라이딩 윈도우(Sliding Window) 정책

배경 및 정의: TCP와 같은 신뢰성 있는 프로토콜에서 송신 측과 수신 측의 속도 차이를 조절하기 위해 사용하는 흐름 제어(Flow Control) 방식입니다. **슬라이딩 윈도우(Sliding Window)**는 수신 측에서 한 번에 받을 수 있는 데이터의 양(윈도우 크기)을 알려주어, 송신 측이 그 범위 내에서 데이터를 연속적으로 보낼 수 있게 합니다.

- 작동 원리:** 수신 측이 데이터를 처리하면 윈도우를 옆으로 밀어내듯(Sliding) 다음 데이터를 받을 수 있는 구간을 업데이트합니다. 이를 통해 매번 확인 응답(ACK)을 기다리지 않고도 효율적인 전송이 가능합니다.
- 실무/시험 포인트:** 윈도우 크기가 너무 작으면 네트워크 대역폭을 충분히 활용하지 못해 속도가 느려지고, 너무 크면 수신 측의 버퍼가 넘쳐 데이터 손실이 발생할 수 있습니다.

3. 취약점 공격의 실제: 애플리케이션 계층의 침투

시스템과 네트워크 제어 메커니즘이 하위 계층(Layer 2~4)의 안정성을 담당한다면, 공격자는 이를 우회하여 상위 계층(Layer 7)의 논리적 허점을 직접 타격합니다. 아래는 웹 애플리케이션의 취약점을 노린 실제 공격 요청의 형태입니다.

```
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
[생략된 헤더 정보...]
[데이터 페이로드...]
```

위의 요청은 DVWA(Damn Vulnerable Web Application)라는 취약점 테스트 환경을 대상으로 한 XSS(Cross-Site Scripting) 공격 시도입니다. /xss_s/ 경로에 POST 방식을 사용하여 악성 스크립트가 포함된 데이터를 전송함으로써, 서버가 이를 검증 없이 브라우저에 출력하게 만드는 것이 목적입니다.

분석 및 연계:

- 네트워크 제어 관점:** 만약 네트워크 상의 침입 방지 시스템(IPS)이나 웹 방화벽(WAF)이 적절한 규칙(Rule)을 가지고 있다면, 위와 같은 특이한 패턴의 POST 요청을 차단할 수 있습니다.
- 시스템 제어 관점:** 애플리케이션이 입력값을 처리할 때, 앞서 언급한 '데이터 구조'에 대한 검증이나 '체크섬'과 같은 무결성 확인 개념이 데이터의 형식을 검증하는 데 적용된다면 이러한 공격을 방어하는 데 도움이 될 수 있습니다.

학습 확인 및 점검 사항

점검 항목	핵심 확인 내용	실무/시험 포인트
Offloading	CPU 부하 분산 여부	특정 하드웨어 가속 사용 시 성능 향상 및 CPU 점유율 변화 확인
Data Integrity	구조체 크기 및 체크섬	정해진 크기(예: 5kb) 초과 데이터 유입 시 버퍼 관리 및 체크섬 검증 누락 확인
QoS	트래픽 우선순위 정책	중요 서비스(VoIP, DB 등)에 대한 대역폭 보장 설정 여부
Sliding Window	흐름 제어 매커니즘	수신 버퍼 크기에 따른 전송 효율성 및 데이터 손실 발생 여부
XSS 공격	입력값 검증(Sanitization)	HTTP POST 요청 내 악성 스크립트 포함 여부 및 WAF 차단 여부

자가 진단 질문:

- 오프로딩 기술이 적용되었을 때와 그렇지 않았을 때, CPU의 역할은 어떻게 달라지는가?
- 체크섬이 불완전하게 구현된 시스템에서 데이터 변조를 탐지할 수 없는 이유는 무엇인가?
- 네트워크 혼잡 시 QoS가 설정되어 있지 않다면 중요 트래픽에 어떤 문제가 발생하는가?
- 슬라이딩 윈도우 크기가 고정되어 있다면, 네트워크 속도가 변할 때 효율성은 어떻게 변하는가?

웹 취약점 분석 및 XSS 실습

260721 모의해킹 수업에서는 네트워크 계층에서의 데이터 무결성 및 전송 원리를 검토한 후, 이를 바탕으로 웹 애플리케이션의 대표적인 취약점인 XSS(Cross-Site Scripting)를 분석하였다. 특히 네트워크 패킷의 구조적 특성과 웹 요청(HTTP Request)의 관계를 이해하는 것이 공격 및 방어의 핵심임을 확인하였다.

1. 네트워크 및 데이터 전송 매커니즘의 이해

웹 취약점을 심도 있게 분석하기에 앞서, 데이터가 네트워크를 통해 전달되는 과정에서 발생할 수 있는 변수들을 검토하였다. 이는 패킷 조작이나 우회 공격을 이해하는 밑바탕이 된다.

- 데이터 무결성 및 구조:** 데이터 전송 시 **체크섬(Checksum)**이 미완성되거나 불완전할 경우, 데이터의 변조 여부를 판단하기 어려워질 수 있다. 또한, 특정 크기(예: 5kb)의 **구조체(Structure)**를 사용하는 환경에서는 데이터가 구조에 맞게 정렬되지 않을 때 예기치 못한 동작이 발생할 수 있음을 학습하였다.
- 자원 활용 및 제어:** **오프로딩(Offloading)** 기술은 CPU의 부담을 줄이기 위해 특정 작업을 네트워크 카드 등 별도 장치로 넘기는 것을 의미하며, 이는 패킷 처리 속도에 영향을 미친다. 또한, **QoS(Quality of Service)** 설정이나 **슬라이딩 정책(Sliding Policy)**과 같은 트래픽 제어 매커니즘은 네트워크 혼잡 시 데이터의 우선순위를 결정하므로, 공격 트래픽의 유입이나 탐지 우회 전략 수립 시 고려해야 할 요소이다.

2. XSS(Cross-Site Scripting) 개념 및 작동 원리

웹 취약점 분석의 핵심 주제인 XSS는 공격자가 악성 스크립트를 웹 페이지에 삽입하여, 해당 페이지를 열람하는 사용자의 브라우저에서 스크립트가 실행되게 만드는 공격이다.

배경 및 정의

웹 애플리케이션은 사용자의 입력을 받아 처리하는 과정이 필수적이다. 만약 서버가 입력값에 포함된 특수 문자나 스크립트 태그를 적절히 필터링(Filtering)하거나 치환(Escaping)하지 않으면, 공격자는 이를 통해 사용자의 세션 쿠키를 탈취하거나 페이지를 변조할 수 있다.

작동 매커니즘

XSS는 스크립트가 저장되거나 전달되는 방식에 따라 구분된다. 이번 실습에서 다룬 **Stored XSS(저장형 XSS)**를 중심으로 그 흐름은 다음과 같다.

- 공격 단계:** 공격자가 게시판, 프로필 설정 등 서버 데이터베이스(DB)에 저장되는 입력창에 악성 스크립트(예: `<script>alert('XSS')</script>`)를 입력한다.
- 저장 단계:** 웹 서버는 해당 스크립트를 검증 없이 DB에 그대로 저장한다.
- 실행 단계:** 일반 사용자가 해당 게시물을 조회하면, 서버는 DB에서 스크립트를 가져와 응답(Response) 본문에 포함시킨다. 사용자의 브라우저는 이를 정상적인 코드로 인식하여 실행한다.

3. DVWA를 이용한 XSS(Stored) 실습 분석

실습에서는 DVWA(Damnable Vulnerable Web Application) 환경에서 직접 HTTP 요청을 구성하여 Stored XSS 취약점을 확인하였다.

HTTP 요청 분석

실습 과정에서 확인된 핵심적인 HTTP 요청 데이터는 다음과 같다.

```
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1\r\n
```

이 요청문의 구조를 통해 다음과 같은 공격 맥락을 도출할 수 있다.

- 메서드(Method): POST** - 데이터를 URL 파라미터가 아닌 요청 본문(Body)에 담아 전송한다. 이는 입력값이 서버 측에 저장되는 과정(Stored)에서 주로 사용된다.
- 경로(Path): /dvwa/vulnerabilities/xss_s/** - 경로명에 포함된 xss_s는 DVWA 내에서 **Stored XSS** 모듈을 타겟팅하고 있음을 의미한다.
- 프로토콜: HTTP/1.1** - 웹 통신의 표준 규약을 준수하여 요청을 전달한다.

실습 수행 및 검증 절차

- 목표:** 서버에 악성 스크립트를 삽입하여, 이후 페이지를 열람하는 모든 사용자에게 스크립트가 실행되도록 한다.
- 수행 단계:**
 - 취약한 입력 폼(예: 댓글창)을 찾는다.
 - 브라우저나 프록시 도구를 사용하여 위와 같은 **POST** 요청을 생성한다. 이때 데이터 필드에 스크립트 페이로드를 포함시킨다.
 - 요청을 전송하여 서버의 응답을 확인한다.
- 성공 여부 확인:** 다른 사용자(또는 다른 세션)로 해당 페이지에 접속했을 때, 의도한 자바스크립트(예: 경고창 팝업)가 실행되는지 확인한다.
- 실패 시 점검 포인트:**
 - 입력값이 서버에서 HTML 엔티티(예: < → <)로 치환되어 출력되는지 확인한다.
 - WAF(웹 방화벽)나 보안 솔루션이 특정 키워드(script 등)를 차단하고 있는지 검토한다.

4. 실무 및 시험 대비 핵심 포인트

구분	핵심 점검 사항	대응 방안 (Best Practice)
탐지 관점	입력값에 특수문자(< , > , ' , ")가 포함되는지 확인	입력값 검증(Input Validation) 수행
방어 관점	서버가 데이터를 출력할 때 스크립트로 해석하는지 확인	출력값 인코딩(Output Encoding) 적용
구조적 관점	POST 요청 시 페이로드가 DB에 직접 저장되는 경로 확인	Content Security Policy(CSP) 설정

주의사항: XSS 실습은 반드시 격리된 실습 환경(DVWA 등)에서만 수행해야 한다. 실제 운영 중인 서비스에 악성 스크립트를 주입하는 행위는 법적 처벌의 대상이 될 수 있다.

학습 확인 질문

- Reflected XSS와 Stored XSS의 가장 큰 차이점은 무엇인가? (힌트: 데이터의 지속성)
- POST 요청을 사용할 때, URL 파라미터에 스크립트를 넣는 방식(GET)과 비교하여 어떤 점이 보안상 더 까다로울 수 있는가?
- 체크섬(Checksum) 오류가 발생하는 패킷이 XSS 공격을 시도할 때 어떤 영향을 줄 수 있는가?

모의해킹 실무 점검 체크리스트

모의해킹(Penetration Testing)은 단순한 취약점 탐지를 넘어, 공격자가 시스템의 다양한 계층(Layer)을 어떻게 침투하고 악용할 수 있는지 전 과정을 시뮬레이션하는 작업입니다. 본 장에서는 수업에서 다룬 네트워크 프로토콜 분석부터 웹 애플리케이션 취약점까지, 실무에서 반드시 확인해야 할 핵심 점검 항목들을 계층별로 정리합니다.

1. 네트워크 및 프로토콜 계층 점검

네트워크 계층에서의 점검은 데이터가 전송되는 과정에서 발생하는 구조적 결함이나, 트래픽 제어 메커니즘의 허점을 찾는 데 집중합니다.

1.1 트래픽 오프로딩(Offloading) 설정 및 보안 경계

배경 및 정의: 오프로딩이란 메인 CPU의 부하를 줄이기 위해 특정 작업(예: 암호화/복호화, 패킷 처리)을 별도의 전용 하드웨어나 가속기(Accelerator)로 넘기는 기술을 의미합니다. SSL/TLS 오프로딩이 대표적입니다.

점검 원리: 보안 장비(WAF, IPS 등)가 트래픽을 검사하기 전에 오프로딩이 수행될 경우, 암호화된 트래픽의 내부 내용을 확인하지 못하는 '보안 사각지대'가 발생할 수 있습니다.

- **점검 포인트:** 보안 장비가 트래픽을 복호화하여 검사하는 지점과 오프로딩이 발생하는 지점의 선후 관계를 확인해야 합니다.
- **흔한 실수:** 로드 밸런서에서 SSL을 종료(Termination)할 때, 내부 네트워크로 전달되는 평문 트래픽에 대한 추가 보안 대책이 없다고 간주하는 경우입니다.

1.2 데이터 구조 및 무결성(Integrity) 검증

배경 및 정의: 프로토콜은 데이터를 주고받을 때 일정한 구조(Structure)와 오류를 확인하기 위한 체크섬(Checksum)을 사용합니다. 체크섬은 데이터 전송 중 변조나 오류가 발생했는지 판단하는 지표입니다.

점검 원리: 특정 크기(예: 5kb 구조체)를 가진 데이터 패킷을 보낼 때, 시스템이 체크섬을 불완전하게(Incomplete) 검증한다면 공격자는 데이터의 일부를 조작하여 악성 페이로드를 삽입할 수 있습니다.

- **실무/시험 포인트:**
 1. 데이터 구조체의 경계 값(Boundary Value)을 넘어서는 데이터 입력 시 버퍼 오버플로우(Buffer Overflow)가 발생하는가?
 2. 체크섬 계산 로직이 취약하여, 변조된 데이터에도 유효한 체크섬 값을 생성할 수 있는가?
- **주의사항:** 체크섬 검증이 생략되거나 일부 필드에 대해서만 수행되는 환경은 데이터 무결성 공격에 매우 취약합니다.

1.3 트래픽 제어 및 정책(QoS & Sliding Policy)

배경 및 정의: QoS(Quality of Service)는 네트워크 트래픽의 우선순위를 정하는 기술이며, 슬라이딩 정책(Sliding Policy, 주로 Sliding Window 알고리즘 기반)은 데이터 흐름을 제어하여 과부하를 방지합니다.

점검 원리: 이러한 제어 메커니즘의 허점을 이용하면 특정 트래픽을 우선적으로 처리하게 만들어 DoS(Denial of Service) 공격을 수행하거나, 속도 제한(Rate Limiting)을 우회할 수 있습니다.

- **점검 항목:**
 - QoS 설정 오류로 인해 중요 서비스가 공격 트래픽에 의해 차단될 가능성이 있는가?
 - 슬라이딩 윈도우(Sliding Window) 기반의 속도 제한 정책을 조작하여 차단 정책을 우회할 수 있는가?

2. 웹 애플리케이션 계층 점검

웹 애플리케이션 점검은 사용자 입력값이 서버 측에서 어떻게 처리되는지를 확인하는 것이 핵심입니다.

2.1 크로스 사이트 스크립팅(XSS) 취약점

배경 및 정의: XSS는 공격자가 웹 사이트에 악성 스크립트를 삽입하여, 해당 사이트를 이용하는 사용자의 브라우저에서 스크립트가 실행되게 하는 공격입니다.

실습 사례 분석: 수업에서 확인된 다음과 같은 HTTP 요청은 XSS 공격의 전형적인 시도 중 하나입니다.

```
POST /dvwa/vulnerabilities/xss_s/ HTTP/1.1
```

작업 목표 및 수행 단계:

1. **대상 식별:** DVWA(Damn Vulnerable Web Application)와 같이 취약점이 존재하는 엔드포인트를 타겟팅합니다.
2. **메서드 확인:** GET 방식이 아닌 POST 방식을 사용하는 경우, 데이터가 URL이 아닌 HTTP Body에 담기므로 데이터 전송 값을 직접 조작해야 합니다.
3. **페이로드 삽입:** 입력 파라미터에 <script>alert('XSS')</script>와 같은 스크립트를 삽입합니다.
4. **결과 확인:** 브라우저에서 경고창(Alert)이 정상적으로 출력되면 취약점이 존재하는 것으로 판단합니다.

실패 시 점검 포인트: 스크립트가 실행되지 않는다면, 서버 측에서 특수문자(<, > 등)를 HTML Entity로 인코딩(Encoding)하거나 필터링(Filtering)하고 있는지 확인해야 합니다.

3. 실무 통합 점검 항목 (Checklist)

모의해킹 수행 시 위 내용을 바탕으로 아래의 체크리스트를 활용하여 점검을 진행하십시오.

점검 영역	주요 점검 항목	판단 기준 및 확인 방법
네트워크/인프라	오프로딩 보안 경계	보안 장비가 복호화된 트래픽을 검사하는 위치에 배치되었는가?
데이터 무결성	체크섬 및 구조체 검증	데이터 변조 시 체크섬 불일치로 인해 패킷이 폐기되는가?
가용성/제어	QoS 및 속도 제한 우회	트래픽 제어 정책을 우회하여 과도한 요청을 보낼 수 있는가?
웹 애플리케이션	XSS (Reflected/Stored)	POST/GET 파라미터에 삽입된 스크립트가 브라우저에서 실행되는가?

전문가의 조언 (Pro-tip): 모의해킹은 단일 취약점을 찾는 것에 그치지 않고, "체크섬 미완성 → 데이터 변조 → 오프로딩된 환경에서의 검사 우회"와 같이 서로 다른 계층의 취약점을 연쇄적으로 결합(Chaining)하여 실제 침투 시나리오를 구성하는 것이 중요합니다.

확인 질문:

- 데이터 구조체의 크기가 고정되어 있을 때, 그 범위를 벗어나는 데이터를 보낼 경우 시스템은 어떻게 반응하는가?
- 웹 애플리케이션에서 POST 요청을 보낼 때, 서버가 입력값을 검증하는 로직이 존재하는가?