

수업 내용 정리 보고서

작성일: 2026년 7월 9일

챕터 구성

- 보안 실험실 인프라 구축 및 네트워크 구성
- HTTP 프로토콜과 클라이언트 사이드 분석 기술
- WebGoat 취약점 실습 및 보안 설계 검증

보안 실험실 인프라 구축 및 네트워크 구성

보안 실습을 진행하기 위해서는 공격자와 방어자가 상호작용할 수 있는 격리된 네트워크 환경을 구축하는 것이 최우선 과제입니다. 이번 단계에서는 UTM(Unified Threat Management, 통합 위협 관리) 장비를 활용하여 IPS(Intrusion Prevention System, 침입 방지 시스템) 환경을 설계하고, 가상화 환경에서의 네트워크 토폴로지를 구성하는 과정을 다룹니다.

1. UTM 장비 구축 및 시스템 검증

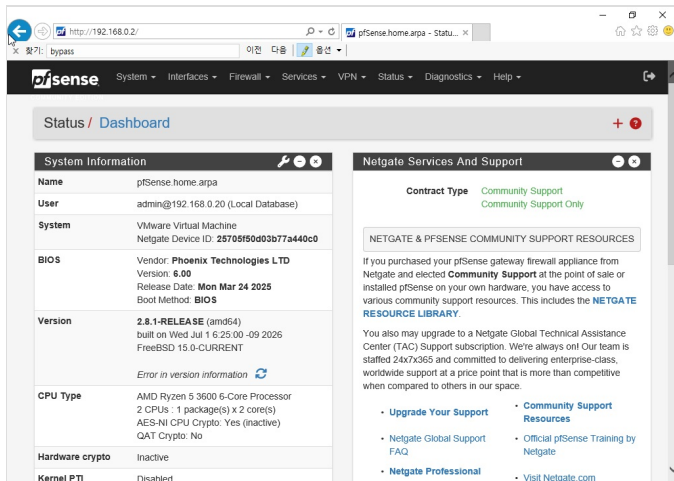
보안 실험실의 핵심 방어 장비로 pfSense를 활용하여 UTM 환경을 구축하였습니다. UTM은 방화벽, IPS, VPN 등 다양한 보안 기능을 하나의 장비에서 처리할 수 있어 실무적인 보안 환경을 시뮬레이션하기에 적합합니다. 장비 설치 후에는 시스템이 하드웨어 및 가상화 환경에 맞게 올바르게 구성되었는지 대시보드를 통해 검증해야 합니다.

검증 과정에서는 다음과 같은 주요 항목을 확인합니다:

- **System Information:** 장비의 호스트 이름(Name)과 운영 체제(FreeBSD 기반) 확인
- **BIOS & Version:** Phoenix Technologies 등 BIOS 벤더 정보와 pfSense 릴리즈 버전(예: 2.8.1-RELEASE) 확인
- **Hardware Capability:** CPU 유형(AMD Ryzen 등)과 AES-NI와 같은 하드웨어 암호화 가속 기능의 활성화 여부, 그리고 커널 PTI(Page Table Isolation) 상태 점검

시스템 대시보드를 통해 하드웨어 사양과 보안 관련 가속 기능이 정상적으로 로드되었는지 확인하는 단계입니다.

pfSense Dashboard UI, VMware Virtual Machine, 2.8.1-RELEASE 버전 표시



pfSense 대시보드를 통해 시스템 정보, BIOS 버전, CPU 유형 및 하드웨어 암호화 활성화 상태를 확인한다

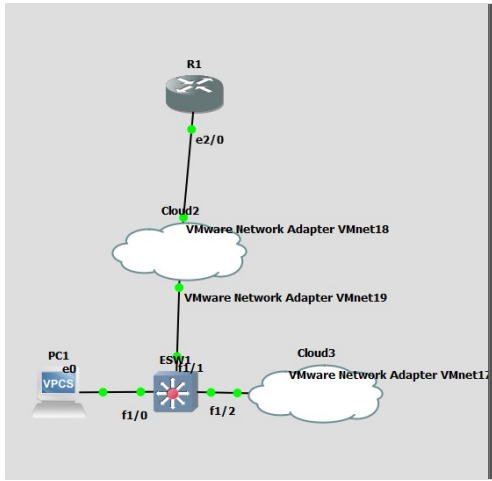
2. 네트워크 토폴로지 설계

실습 환경은 가상 네트워크 어댑터(VMnet)를 활용하여 물리적 장비가 없는 가상 환경에서도 WAN(외부망)과 LAN(내부망)이 분리된 네트워크 구조를 구현합니다. 구축된 네트워크의 논리적 흐름은 다음과 같습니다.

- **R1 (Edge Router):** 외부망(WAN)과 연결되는 최상단 라우터로, e2/0 인터페이스를 통해 Cloud2(VMnet18)와 연결됩니다.
- **ESW1 (Core Switch):** 내부망을 구성하는 스위치로, Cloud2(VMnet18)와 Cloud3(VMnet17) 사이의 가상 클라우드 환경에 연결되어 각 노드를 중계합니다.
- **PC1 (End Device):** 스위치(ESW1)의 f1/0 포트에 연결된 실습 대상 단말로, VPCS를 통해 네트워크 통신을 수행합니다.

전체적인 연결 관계는 아래의 구성도를 통해 확인할 수 있습니다.

R1, ESW1, PC1 장비 및 VMnet17/18/19 연결 관계 표시



R1 라우터, ESW1 스위치, PC1 및 VMnet 가상 어댑터를 활용한 네트워크 구성도이다

3. 네트워크 및 파일 시스템 핵심 개념

인프라 구축 시 설계 단계에서 결정해야 하는 중요한 두 가지 기준은 네트워크 인터페이스(NI) 모드와 파일 시스템의 종류입니다.

가. 네트워크 인터페이스(NI) 작동 모드

라우터 또는 UTM 장비를 설정할 때 WAN과 LAN의 네트워크 인터페이스가 어떻게 동작하는지에 따라 두 가지 모드로 구분됩니다.

- **브릿지 모드(Bridge Mode):** WAN과 LAN의 NI가 동일한 서브넷에 속하거나 물리적으로 연결되어 동일한 네트워크 세그먼트처럼 동작하는 방식입니다.
- **라우터 모드(Router Mode):** WAN과 LAN의 NI가 서로 다른 서브넷을 가지며, 장비가 L3 계층에서 라우팅 기능을 수행하는 방식입니다. 보안 실험실에서는 망 분리를 위해 주로 이 모드를 사용합니다.

나. 파일 시스템 선택 (ZFS vs UFS)

보안 장비의 안정적인 데이터 보존을 위해 어떤 파일 시스템을 사용할지 결정해야 합니다.

- **ZFS (Zettabyte File System):** 높은 안정성과 데이터 무결성을 보장하는 파일 시스템입니다. 스냅샷, 복구 기능 등이 강력하여 보안 장비의 OS 및 데이터 보존 용으로 권장됩니다.
- **UFS (Unix File System):** 일반적인 Unix 계열 시스템에서 사용되는 표준 파일 시스템입니다. 구조가 단순하지만, ZFS에 비해 데이터 보호 및 복구 기능이 제한적일 수 있습니다.

4. 실습 점검 체크리스트

인프라 구축이 완료된 후, 다음 항목들이 정상적으로 동작하는지 점검해야 합니다.

1. 네트워크 가시성: PC1에서 R1까지의 경로가 라우팅 테이블에 정상적으로 잡히는가?
2. 모드 동작 확인: 설정한 네트워크 모드(Bridge 또는 Router)에 따라 IP 대역이 분리되었는가?
3. 장비 자원 확인: pfSense 대시보드에서 CPU와 하드웨어 암호화 기능이 'Active' 상태인가?
4. 파일 시스템 안정성: 구축된 UTM 장비가 ZFS 등 권장되는 파일 시스템으로 구성되었는가?

HTTP 프로토콜과 클라이언트 사이드 분석 기술

1. 실습 환경 구축 및 네트워크 구조

본 실습은 보안 취약점을 학습하기 위해 의도적으로 취약하게 설계된 WebGoat 애플리케이션을 기반으로 진행됩니다. 실습을 위한 가상 환경은 pfSense UTM 장비와 가상 라우터, 스위치를 포함한 복합적인 네트워크 구조로 구성되어 있습니다.

먼저, 네트워크의 게이트웨이 역할을 수행하는 pfSense 장비를 구축합니다. pfSense는 오픈 소스 기반의 방화벽 및 라우터 소프트웨어로, 실습 환경 내에서 WAN과 LAN의 인터페이스를 구분하여 트래픽을 제어하는 역할을 합니다. 설치 후 대시보드를 통해 시스템의 사양을 확인합니다.

네트워크 토폴로지는 가상 네트워크 어댑터인 VMnet을 활용하여 설계되었습니다. R1 라우터를 정점으로 ESW1 스위치가 연결되어 있으며, PC1(VPCS)과 같은 엔드포인트 기기들이 각기 다른 네트워크 세그먼트에 배치되어 통신을 수행합니다.

2. HTTP 프로토콜의 이해

웹 보안 공격의 가장 기초가 되는 것은 클라이언트와 서버가 데이터를 주고받는 규칙인 HTTP(HyperText Transfer Protocol)를 이해하는 것입니다. 모든 웹 요청은 다음과 같은 세 가지 핵심 요소로 구성됩니다.

- **요청/응답 라인(Request/Response Line):** HTTP 메서드(GET, POST 등), 요청 경로, HTTP 버전을 포함합니다.
- **헤더(Header):** 클라이언트 환경, 인증 토큰, 쿠키(Cookie) 등 데이터에 대한 메타 정보를 담고 있습니다.

- 바디(Body): 실제 전송하고자 하는 데이터(Payload)가 포함되는 영역으로, 주로 POST 메서드에서 사용됩니다.

실무에서 주의해야 할 가장 큰 차이점은 GET과 POST 메서드의 데이터 노출 방식입니다. GET 방식은 파라미터를 URL에 그대로 포함시키므로 브라우저 히스토리나 서버 로그에 민감 정보가 노출될 위험이 있습니다. 반면 POST 방식은 데이터를 바디(Body)에 실어 보내므로 URL상에는 나타나지 않습니다. 하지만 데이터가 바디에 들어있다고 해서 보안이 보장되는 것은 아니며, 클라이언트 측의 검증을 우회할 수 있는 프록시 공격에 노출될 수 있습니다.

3. [실습] HTTP Basics: 숨겨진 데이터 찾기

WebGoat의 기초 레슨에서는 HTTP 요청과 응답의 구조를 직접 관찰합니다. 목표는 클라이언트 화면에는 보이지 않지만, 서버가 응답할 때 바디(Body)에 함께 보낸 '매직 넘버(Magic Number)'를 찾아내는 것입니다.

이를 위해 브라우저의 개발자 도구(Developer Tools) 내 Network 탭을 활용합니다. 사용자가 입력창에 값을 입력하고 전송할 때 발생하는 실제 트래픽을 가로채서 응답 데이터 전체를 분석해야 합니다.

실습 절차:

1. 브라우저에서 F12를 눌러 개발자 도구를 실행하고 Network 탭을 선택합니다.
2. 웹 페이지의 입력창에 테스트 값을 입력한 후 전송 버튼을 누릅니다.
3. Network 탭에 기록된 요청 목록 중 해당 요청을 클릭합니다.
4. Headers 탭에서 메서드가 GET인지 POST인지 확인하고, Response 탭에서 서버가 보내온 데이터 본문을 검사하여 숨겨진 값을 추출합니다.

4. HTTP 프록시를 이용한 요청 변조(Request Tampering)

단순히 데이터를 관찰하는 것을 넘어, 공격자는 클라이언트와 서버 사이의 통신을 중간에서 가로채는 인터셉트 프록시(Intercepting Proxy)를 사용하여 요청 내용을 조작할 수 있습니다. 이를 요청 변조(Request Tampering)라고 합니다.

예를 들어, 서버가 POST 방식으로만 데이터를 받도록 설계되어 있더라도, 공격자가 프록시를 통해 메서드를 GET으로 바꾸거나 바디에 담긴 값을 쿼리 파라미터로 변경하여 보낼 경우, 서버의 검증 로직이 부실하다면 공격이 성공할 수 있습니다. 이는 클라이언트 측 보안 통제(Client-Side Enforcement)에만 의존하는 설계 결함(Insecure Design)의 전형적인 사례입니다.

5. [실습] HTTP Proxies: 요청 가로채기 및 조작

WebGoat 실습에서는 프록시 툴을 사용하여 클라이언트가 보낸 요청을 가로챈 뒤, 실시간으로 세 가지 요소를 변조합니다.

실습 수행 단계:

1. 프록시 툴에서 인터셉트(Intercept) 기능을 활성화합니다.
2. 클라이언트에서 요청을 보낸 후, 프록시 툴에 멈춰있는 요청 패킷을 확인합니다.
3. 메서드 변조: POST로 설정된 메서드를 GET으로 변경합니다.
4. 헤더 조작: 사용자 정의 헤더(예: `x-request-intercepted: true`)를 추가합니다.
5. 데이터 이동: HTTP 바디에 있던 값을 URL의 쿼리 파라미터로 옮겨 적습니다.
6. 변조된 요청을 서버로 전송하여 서버가 조작된 요청을 정상적으로 처리하는지 확인합니다.

검증 포인트: 만약 변조된 요청에 대해 서버가 에러를 내지 않고 성공 응답을 보낸다면, 해당 애플리케이션은 서버 측에서 요청의 무결성과 형식을 검증하지 않는 심각한 취약점을 가지고 있음을 의미합니다.

6. 개발자 도구를 활용한 클라이언트 사이드 분석

공격자는 별도의 도구 없이도 브라우저에 내장된 개발자 도구(Developer Tools)만으로도 정찰(Reconnaissance)을 수행할 수 있습니다.

- Console 탭: 페이지에 로드된 JavaScript 함수를 직접 호출하거나 변수 값을 조작할 수 있습니다. UI상에 버튼이 보이지 않더라도, 함수 이름만 안다면 콘솔에 직접 입력하여 숨겨진 기능을 실행할 수 있습니다.
- Network 탭: 서버와 주고받는 모든 API 호출과 파일 리소스를 모니터링하여, 숨겨진 엔드포인트나 민감한 데이터 전송 패턴을 파악할 수 있습니다.

7. [실습] Console을 이용한 숨겨진 함수 호출

실습에서는 화면에 노출되지 않은 내부 JavaScript 함수를 콘솔에서 직접 호출하여 숨겨진 로직을 실행해 봅니다.

실습 수행 단계:

1. 개발자 도구의 Console 탭으로 이동합니다.
2. 분석된 스크립트 내에서 발견된 함수명(예: `webgoat.customjs.phoneHome()`)을 입력하고 엔터를 누릅니다.
3. 콘솔에 출력되는 리턴값(Return Value)을 확인하여 미션을 완료합니다.

주의사항: 클라이언트 사이드에서 수행되는 모든 로직은 사용자가 완전히 통제할 수 있습니다. 따라서 중요한 비즈니스 로직이나 권한 검증은 절대로 클라이언트 JavaScript에 맡겨서는 안 되며, 반드시 서버 측에서 재검증되어야 합니다.

학습 요약 및 점검 체크리스트

[핵심 요약]

- HTTP 데이터는 Line, Header, Body로 구분되며, GET은 URL에, POST는 Body에 데이터를 실어 보낸다.
- 프록시는 클라이언트와 서버 사이의 통신을 가로채서 요청(Request)과 응답(Response)을 실시간으로 변조할 수 있다.
- 클라이언트 사이드 검증(JS 함수 호출 등)은 공격자에 의해 쉽게 우회될 수 있으므로 보안의 근거가 될 수 없다.

분석 항목	주요 확인 사항	공격적 관점의 체크포인트
HTTP 메서드	GET vs POST 구분	POST 요청을 GET으로 변조 시 서버가 허용하는가?
데이터 위치	Header, Body, Query Param	Body에 있어야 할 값이 URL에 노출되거나 변조 가능한가?
클라이언트 로직	JavaScript 함수 노출 여부	콘솔에서 권한 우회나 데이터 조작 함수를 호출할 수 있는가?
트래픽 흐름	Network 탭 기록	화면에 보이지 않는 API 호출이나 민감 데이터가 전송되는가?

WebGoat 취약점 실습 및 보안 설계 검증

1. 실습 환경 구축 및 네트워크 설계

본 실습을 위해 UTM(Unified Threat Management) 장비로 pfSense를 설치하고 IPS(Intrusion Prevention System) 환경을 구축하였다. 시스템의 안정성을 위해 파일 시스템은 ZFS(Zettabyte File System)를 사용하였으며, pfSense의 대시보드를 통해 하드웨어 사양과 운영 환경을 점검하였다.

시스템 관리 화면을 통해 확인한 주요 사양은 다음과 같다. BIOS 버전은 6.0이며, CPU는 AMD Ryzen 5 3600 6-Core Processor가 구동 중임을 확인하였다. 또한, 네트워크 인터페이스(NI) 설정을 통해 WAN과 LAN의 구성을 확인하여 브릿지(Bridge) 모드 또는 라우터(Router) 모드로의 동작 여부를 판단할 수 있다.

전체 네트워크 토폴로지는 R1 라우터를 최상단에 배치하고, VMware Network Adapter(VMnet)를 활용하여 가상 네트워크를 계층적으로 구성하였다. R1은 e2/0 인터페이스를 통해 Cloud2(VMnet18)에 연결되며, Cloud2는 VMnet19를 통해 ESW1 스위치에 연결된다. ESW1의 f1/1 포트는 PC1(VPCS)의 e0 인터페이스와 연결되어 있으며, 추가적으로 Cloud3(VMnet17)를 통해 외부망과 연결되는 구조를 가진다.

2. WebGoat 개요

실습의 대상인 WebGoat는 OWASP(Open Worldwide Application Security Project)에서 개발한 의도적으로 취약하게 설계된 웹 애플리케이션이다. 이는 실제 운영 중인 서비스에 공격을 시도하는 불법 행위를 방지하고, 로컬 환경에서 웹 보안 취약점(OWASP Top 10)을 안전하게 학습하기 위한 도구로 활용된다.

3. HTTP 통신 구조 및 보안 취약점 분석

모든 웹 공격의 기초는 HTTP 요청(Request)과 응답(Response) 구조를 이해하는 데서 시작된다. HTTP 트랜잭션은 크게 세 가지 요소로 구성된다.

- 요청/응답 라인(Request/Response Line): 메서드(GET, POST 등), 경로, HTTP 버전을 포함한다.
- 헤더(Header): 클라이언트의 환경 정보, 인증 토큰, 쿠키 등 부가 정보를 전달한다.
- 바디(Body): POST 메서드 등에서 실제 전송하고자 하는 데이터가 실리는 영역이다.

보안 관점에서 GET 방식과 POST 방식의 차이는 매우 중요하다. GET 방식은 파라미터가 URL에 그대로 노출되므로, 브라우저 히스토리나 서버 로그를 통해 민감한 정보(비밀번호, 세션 토큰 등)가 유출될 위험이 크다. 따라서 설계 단계에서 민감한 데이터는 반드시 POST 방식을 사용하여 바디에 담아 보내야 한다. 이를 준수하지 않는 설계 결함은 CWE-598 및 OWASP A06:2025 Insecure Design 취약점으로 이어진다.

4. HTTP 프록시를 이용한 요청 변조(Request Tampering)

공격자는 브라우저와 서버 사이에 위치하는 인터셉트 프록시(Intercepting Proxy)를 사용하여 클라이언트가 보낸 요청을 가로채고, 서버에 전달되기 직전에 데이터를 실시간으로 수정할 수 있다. 이는 클라이언트 측에서 수행되는 검증 로직을 무력화하는 데 사용된다.

실습을 통해 다음과 같은 세 가지 방식의 변조를 수행하였다.

1. 메서드 변경: 전송 방식을 POST에서 GET으로 변경하여 서버의 처리 로직을 교란한다.
2. 헤더 조작: `x-request-intercepted: true`와 같은 임의의 헤더를 삽입하여 서버의 검증을 우회한다.
3. 데이터 위치 변경: POST 바디에 담겨야 할 데이터를 GET의 쿼리 파라미터로 옮겨 서버의 데이터 수신 로직을 테스트한다.

이러한 공격 기법은 가격 정보나 권한 값을 조작하는 eShoplifting과 같은 사례와 유사하며, 서버가 클라이언트로부터 전달받은 값을 신뢰할 수 없음을 증명한다.

5. 개발자 도구를 활용한 클라이언트 측 분석

별도의 도구 없이도 브라우저의 개발자 도구(Developer Tools)만으로도 클라이언트 측 보안 약점을 찾아내는 정찰(Recon) 활동이 가능하다. 특히 Console 탭과 Network 탭은 강력한 무기가 된다.

5.1 Console 탭을 통한 함수 직접 호출

개발자는 페이지 UI 상에 버튼을 숨겨두더라도, 자바스크립트 함수명을 알면 Console 탭에서 직접 함수를 호출할 수 있다. 실습에서는 `webgoat.customjs.phoneHome()` 함수를 직접 실행하여, UI에 노출되지 않은 숨겨진 로직이 동작하는지 확인하였다.

이러한 행위는 클라이언트 측에 비즈니스 로직이 노출되어 있거나, 서버에서 함수 호출에 대한 권한 검증이 누락되었을 때 발생하는 **Insecure Design** 결함을 보여준다.

5.2 Network 탭을 통한 숨겨진 데이터 추출

화면에 표시되는 정보가 전부가 아니다. Network 탭을 모니터링하면 서버가 응답(Response) 시 전달하는 실제 데이터 전체를 확인할 수 있다. 화면상에는 보이지 않더라도 응답 본문(Response Body)에는 매직 넘버나 관리자용 데이터가 포함되어 있을 수 있다.

실습 과정에서 HTTP 응답을 확인한 결과, 화면에 노출되지 않은 특정 값을 추출하여 인증이나 쿼즈 해결에 활용할 수 있음을 확인하였다.

6. 실습 결론 및 보안 점검 체크리스트

이번 실습을 통해 클라이언트 측의 검증은 언제든 공격자에 의해 조작될 수 있다는 점을 확인하였다. 보안 설계 시에는 다음 사항을 반드시 점검해야 한다.

점검 항목	보안 권고 사항
데이터 전송 방식	민감한 정보(비밀번호, 토큰 등)는 절대 GET 파라미터로 전달하지 말고 POST 바디를 사용할 것.
서버 측 검증	클라이언트가 보낸 모든 값(Method, Header, Body)은 신뢰하지 말고 서버에서 반드시 재검증할 것.
비즈니스 로직 보호	클라이언트 사이드 자바스크립트 함수에 중요한 비즈니스 로직을 포함하지 말 것.
정보 노출 최소화	HTTP 응답 시 클라이언트 화면에 불필요한 민감 데이터가 포함되지 않도록 Response Body를 관리할 것.