韓國컴퓨터情報學會 論文誌
第14卷 第2號, 2009. 2.

2009-14-2-3-6

AOP를 이용한 웹 애플리케이션의 보안성 강화 방안

선 수 림*, 이 금 석**

A Method for Security Strengthening of Web Application using AOP

Soo-Rim Sun*, Keum-Suck Lee**

요 약

웹 애플리케이션과 웹 기반 정보 시스템의 이용이 증가하면서 웹 애플리케이션 공격도 증가하고 있다. 다양한 웹 공격 중에 사용자에게 큰 손해를 입힐 수 있는 공격으로 최근에 크로스 사이트 요청 변조(Cross Site Request Forgery, XSRF) 공격이 대두되고 있다. 하지만 기존에 개발되어 운영되고 있는 애플리케이션에 이런 공격을 막기 위해 보안 기능을 추가하는 것은 기업이나 조직에 많은 비용과 위험을 초래할 수 있다. 이러한 보안 기능을 레거시(legacy) 시스템에 효과적으로 적용하기 위해 관점 지향 프로그래밍(Aspect-Oriented Programming, AOP)에서 제공하는 모듈화의 장점을 이용하여 관심사(concern)를 분리한다. 본 논문에서는 JEE(Java Enterprise Edition) 환경의 시스템에서 관점 지향 프로그래밍의 애스펙트(aspect)를 이용하여 크로스 사이트 요청 변조 공격을 탐지하고 방어하는 접근방법을 제시한다.

Abstract

As use of web applications and web-based information systems increases, so web application attacks are increasing. Recently, XSRF(Cross Site Request Forgery) attacks among a variety of web attacks become important because victim's damage caused by such attacks can be severe. But adding security functions for preventing XSRF attacks to existing developed and running software systems could affect more dangerous and expensive to companies and organizations. We suggest effectively adding these security functions to legacy systems, could separate concerns using advantage of the modularity offered by AOP(Aspect-Oriented Programming) methodology. In this paper, we have presented approach for detecting and preventing XSRF in JEE systems using aspect of AOP.

▸ Keyword : 관점 지향 프로그래밍(Aspect-Oriented Programming, AOP), 크로스 사이트 요청 변조(Cross Site Request Forgery, XSRF), 웹 애플리케이션 보안(Web Application Security)

• 제1저자 : 선수림
• 투고일 : 2008. 12. 23, 심사일 : 2009. 1. 16, 게재확정일 : 2009. 2. 6.
* 동국대학교 컴퓨터공학과 ** 동국대학교 컴퓨터공학과 교수

Ⅰ. 서 론

전웹 기반 애플리케이션은 통신, 협업, 정보 검색 및 기업 경영에 많은 변화를 가져왔으며, 매일매일 수많은 사용자들이 다양한 웹 애플리케이션을 이용하고 있다. 이렇게 웹에 대한 의존성이 증가하고 있는 만큼 웹 애플리케이션을 악용하려는 공격자들도 갈수록 증가하고 있다.

기업 및 조직은 직접적으로 사용자에게 정보를 제공하기 위해 웹 서버 혹은 웹 애플리케이션 서버(Web Application Server, WAS)를 사용한다. OWASP(Open Web Application Security Project)의 10대 가장 심각한 웹 애플리케이션 보안 취약점으로 가장 상위에 있는 취약점이 크로스 사이트 스크립팅(Cross Site Scripting, XSS)이고, 그 다음이 인젝션(injection) 취약점이다[1]. 인젝션 취약점에서는 특히 SQL 인젝션 취약점이 매우 흔하다. 웹 애플리케이션 보안 분야의 이전 연구는 주로 크로스 사이트 스크립팅과 SQL 인젝션 공격의 완화에 초점을 맞췄다. 하지만 최근 사회공학적 기법을 사용하는 금융범죄자들이 피싱(Pishing) 등의 방법으로 사용자에게 금전적인 피해를 주고 있는 사례가 증가하고 있으며[2], 이것의 대표적인 공격 방식으로 크로스 사이트 요청 변조가 있다. 일반적으로 사용자 인증이 성공하게 되면 웹 애플리케이션에서는 해당 세션 구간 동안 인증된 사용자의 요청을 신뢰하기 때문에 공격자가 피해자를 대신하여 임의의 HTTP 요청을 발생시킬 수 있다. 이것은 요청이 실제 의도된 것이었는지에 대한 검증없이 실행하기 때문에 문제가 발생하며, 그 결과 크로스 사이트 요청 변조에 취약한 많은 웹 애플리케이션이 존재한다. 그러나 기존의 완화 기법들이 제시하는 방어 기술은 기존 시스템에 통합하기 위해서 상당한 수작업이 요구되기 때문에 시간이 많이 소모되고, 오류가 많이 발생한다.

관점 지향 프로그래밍은소프트웨어 시스템에서 관심사의 분리를 개선하고 소프트웨어의 비즈니스 부분을 수정할 필요없이 횡단 기능성을 추가하기 위한 기술로서 제안되었기 때문에 위의 문제점을 해결하는데 좋은 대안이 되고 있다. 관점 지향 프로그래밍은특징적인 모듈화 방법으로 보안과 같은 관심사를 다루는 것을 가능하게 하는 구체적인 언어 메커니즘을 제공한다[3, 4].

본 연구에서의 주요 목표는 크로스 사이트 요청 변조 웹 공격을 방어하기 위한 보안 애스펙트를 설계하고 구현하여 레거시 시스템에서 소스의 변경 없이 웹 공격을 방어하는 것이다. AspectJ와 AspectWerkz에서 제공되는 애스펙트프로그래밍 모델을 기반으로 제안하고 요청 트리거가 되는 서버 측에서 생성된 HTML의 하이퍼링크 및 폼 태그에 토큰을 자동으로 삽입한 후 요청시에 세션ID와 토큰을 비교하여 크로스 사이트 요청 변조 공격을 방어하는 방법으로 필요한 요소를 정의한다.

본 논문은 다음과 같이 구성된다. Ⅱ장의 관련연구에서는 크로스 사이트 요청 변조 공격과 관점 지향 프로그래밍의 개념에 대해 설명한 후 기존에 연구된 크로스 사이트 요청 변조 완화 기술 및 관점 지향 프로그래밍의 보안 관련 연구에 대해 기술한다. Ⅲ장에서는 관점 지향 프로그래밍을이용하여 보안 기능을 적용한 웹 애플리케이션 구조를 설계한다. Ⅳ장에서는 웹 애플리케이션 서버 구조에 통합된 애스펙트의 구현을 정의한다. Ⅴ장에서 실험 및 그 결과에 대해 분석하고, 마지막으로 Ⅵ장에서 결론 및 향후 과제에 대해 기술한다.

Ⅱ. 관련 연구

2.1 크로스 사이트 요청 변조

HTTP는 무상태 프로토콜로서 어떤 요청이 특정한 사용자에 속한다는 것을 인식하지 못한다. 이것은 성공적으로 로그인이 이미 수행된 사용자의 요청을 식별하기 위한 직접적인 메커니즘이 없으므로 애플리케이션이 이 문제를 극복하기 위해 클라이언트에서 쿠키로 사용자의 상태를 기억하게 하거나 서버에 데이터를 저장하여 사용자의 인증 상태를 판단하는 방법을 사용한다[5]. 이러한 방법으로 저장된 세션은 사용자의 인증 상태를 추적하기 위해 사용될 수 있다. 하지만 웹 애플리케이션이 세션에 의존하여 별도의 검증 과정을 거치지 않고 권한이 부여된 중요한 기능을 수행하기 때문에 크로스 사이트 요청 변조 공격에 노출되게 된다. 또한 크로스 사이트 요청 변조는 사용자의 인증 후 세션을 통하여 공격이 수행이 되기 때문에 추적이 불가능하다[6].

국내에서는 2008년 2월에 발생한 온라인 경매 사이트인 옥션(www.auction.co.kr)에서의 개인 정보 유출 사고가 크로스 사이트 요청 변조 공격의 피해로 보고되었다. 공격자가 익스플로잇을 내장한 대량의 스팸 메일을 발송하여 옥션의 내부 직원이 메일을 확인하는 순간 사내 ID와 비밀번호가 공격자들에게 전송되도록 하였으며, 이것이 크로스 사이트 요청 변조 공격의 한 형태이다. 이렇게 훔쳐낸 ID로 옥션 서버에 접속하여 사용자들의 개인 정보를 조회한 것이다. 이 사고로 해당 사이트의 사용자인 1800만명의 일반 개인 정보 및 거래

정보가 유출되었으며, 이로 인한 게임 및 포털 사이트에서의 명의도용 등의 2, 3차 피해가 우려되고 있다[7].

크로스 사이트 요청 변조 공격이 이루어지기 위해서는 몇 가지의 가정이 필요하다. 첫째, 공격자는 현재 피해자가 인증한 사이트의 취약점에 대해서 잘 알고 있어야 한다. 둘째, 공격자의 목표 사이트가 자동 로그인을 허용하고 있거나 피해자가 현재 로그인한 상태여야 한다. 셋째, 목표 사이트는 추가적인 인증을 요구하지 않아야 한다. 이러한 세가지 가정이 만족되면 크로스 사이트 요청 변조 공격에 매우 취약하게 된다[8].

2.2 관점 지향 프로그래밍

관점 지향 프로그래밍이 정의되게 된 것 중에 하나는 톰캣 서블릿 엔진의 연구를 통해서였다[3]. Gregor Kiczales와 그의 팀은 세션 관리나 로깅과 같이 다른 클래스로 분명하게 모듈화될 수 있는 성질의 기능들이 여러 클래스나 메서드에 포함되어 있다는 것을 발견했다. 이런 현상은 코드 분산으로 알려졌다. 개발자들이 오류를 수정하거나 기능을 개선하려면 여러 소스 파일을 검색하고 변경해야만 한다. 이것은 낮은 생산성과 또 다른 오류를 발생시킨다. 또 다른 경우는 여러 클래스 주변에 분산된 코드가 중복되어 있어 주어진 메서드가 다른 기능과 관련된 관심사와 혼합되는 코드 혼합 현상이다. 이것이 반복되게 되면 애플리케이션의 유지보수성 및 이해성이 떨어지게 된다.

관점 지향 프로그래밍○의 목적은 이런 문제에 대한 해결책을 제공하기 위한 것이다. 새로운 프로그래밍 패러다임인 관점 지향 프로그래밍은 애스펙트, 결합점(join point), 교차점(pointcut) 및 어드바이스(advice)와 같은 개념을 도입한다. 그러나 이런 개념은 클래스, 오브젝트, 프로시저나 메서드 같은 기존의 것들을 대체하지는 않는다. 오히려 관점 지향 프로그래밍은 이런 기존의 기술을 보완하는 것으로 알려졌다. 게다가 이 개념은 Java, C#, Ada, COBOL 등과 같은 객체 지향이나 절차적인 프로그래밍 스타일의 문법에 동일하게 적용된다. 또한 애스펙트는 컴파일시점(compile-time) 혹은 실행시점(run-time)에 적용될 수 있으며, 이것을 관점 지향 프로그래밍에서는 위빙(weaving) 이라고 한다.

보안과 같은 횡단 기능을 구현 또는 운영 단계에서 적용하는 데는 많은 어려움과 위험요소들이 있으며, 이를 해결하기 위해 관점 지향 프로그래밍 기술을 사용할 수 있다.

2.3 크로스 사이트 요청 변조 완화 기술

웹 개발 분야에서 흔히 크로스 사이트 요청 변조 위협을 완화하기 위한 가장 일반적인 예방책은 GET 파라미터 대신에 POST를 사용하는 것이다. 그러나 이 접근법은 크로스 사이트 요청 변조 공격을 정확하게 막지 못한다. 단지 이미지 태그의 사용과 같은 특정한 공격에 대해서 공격자에게 장벽을 제공할 뿐이다. 또한 완벽하게 GET 파라미터의 사용을 제거하는 것은 때때로 사용자에게는 브라우징의 불편을 일으킬 수 있으며, 개발자들은 구현하기가 더 어려워지는 문제가 있다[8].

HTTP Referrer 헤더를 검사하는 것은 만약 웹 애플리케이션이 정확하다면 효과적인 대응방안이 될 수 있다. 검증된 경유 주소 리스트를 관리함으로써 해당 애플리케이션의 요청이 크로스 사이트 요청 변조 공격에 의해 생성된 것인지 추론할 수 있고, 해당 트랜잭션이 수행되는 것을 거부할 수 있다. 하지만 현재 브라우저는 이 헤더를 비우거나, 임의의 값을 보내기 위해 구성될 수 있다. Referrer 헤더를 보내는 것은 제삼자에게 민감한 정보가 노출될 수 있기 때문에 추천되지 않고 있다[9]. 이것은 빈 Referrer 헤더를 어떻게 처리해야 하는지의 문제가 생긴다. 빈 Referrer 헤더를 유효한 요청으로 분류할 때 추천에 따라 Referrer 헤더의 전송을 하지 않도록 설정하는 사용자에 대한 공격의 탐지가 불가능하게 된다. 그러한 요청을 모두 크로스 사이트 요청 변조 공격으로 간주하게 되면 이러한 사용자의 모든 요청이 거부가 된다. 이 문제는 공격자가 빈 Referrer로 크로스 사이트 요청 변조 요청을 보내기 위해 다양한 브라우저의 설정을 변경하는 속임수로 사용될 수 있다[10].

쿠키가 세션ID를 저장하기 위해 사용되었을 때는 크로스 사이트 요청 변조 공격이 이루어질 수 있다. 그 이유는 사용자가 단순한 링크를 클릭했을 때 조차도 브라우저가 자동으로 요청에 쿠키를 포함한다는 것이다. URL 재작성의 경우에서 세션ID는 명시적으로 하이퍼링크나 폼 같은 요청에 삽입된다. 그런 다음 공격자가 크로스 사이트 요청 변조 공격을 수행하는 하이퍼링크를 통하여 공격을 시도할 때 이 링크는 적절한 세션ID를 포함하고 있지 않을 것이므로 공격이 성공하지 못할 것이다. 물론 공격자는 식별자에 대한 지식이 없기 때문에 정확한 세션ID로 링크를 생성할 수 없다. 하지만 다른 측면에서 보면 공격자는 인증된 사용자로 가장하기 위해 직접 이 ID를 사용할 수도 있다[6, 8].

최선의 해결책은 요청이 실제 어디서 발생했는지를 식별하기 위해 클라이언트와 서버간에 공유된 비밀 값인 토큰을 사용하는 것이다. 이 토큰은 애플리케이션에 의해 생성되어야 하며, 공격자가 쉽게 추측할 수 없어야 한다. 또한 현재 세션과 이 생성된 토큰간에 관련지을 수 있어야 한다. 금전 거래와 같이 인증이 요구되는 애플리케이션에 대한 요청은 정확한

토큰을 포함하고 있을 때만 처리가 된다. 이 접근법의 약점은 토큰을 포함시키기 위해 상당한 수작업이 필요하다는 것이며, 기존의 대규모 웹 애플리케이션에 이들 토큰 관리에 필요한 메커니즘을 보완하기 위해서는 상당한 애플리케이션 관련 지식과 안전한 코드 추가를 위한 많은 노력이 요구된다. 더 중요한 것은 개발자가 오류를 만들거나 누락시키는 부분이 있기 때문에 수정된 코드가 실제 크로스 사이트 요청 변조 취약점으로부터 안전한지 보장할 수가 없다는 것이다[6].

크로스 사이트 요청 변조 완화 메커니즘은 단지 GET 요청을 POST 요청으로 바꾸는 것과 HTTP 요청의 Referrer 헤더의 정보에 의존하는 등의 부분적인 보호를 제공하는 것, 또는 보호되어야만 하는 각각의 개별적인 애플리케이션에 대해 애플리케이션의 결과로 공유된 비밀 값을 삽입하는 것과 같이 많은 수정을 요구하는 것이 있다.

본 논문에서 크로스 사이트 요청 변조 공격을 방어하기 위해 제안하는 접근법은 Jovanovic이 제안한 프록시 기반의 크로스 사이트 요청 변조 공격 방어 기법에 기반하고 있다[6]. 프록시 기반의 방어 기법은 PHP와 Apache 웹 서버 환경에서 요청 파라미터에 토큰을 삽입하여 세션ID와의 비교를 통해 크로스 사이트 요청 변조 공격을 탐지한다. 이때 Apache 웹 서버에 애드-온 형태로 제공되므로, 어느 정도는 자동으로 공격을 방어할 수 있지만, 특별한 경우에 별도의 소스 변환 작업이 필요한 문제점이 있다.

본 논문에서는 애플리케이션 소스 코드를 변경하지 않고 공유된 비밀 값에 기반한 토큰을 삽입하는 접근법을 이용한다.

2.4 관점 지향 프로그래밍을 이용한 보안 기술

관점 지향 프로그래밍에서의 보안에 대한 연구가 이미 여러 분야에서 이루어지고 있다. 접근 제어, 암호화, 디지털 서명의 추가, 인가 및 인증 등의 연구가 있으며 대부분 구현은 AspectJ에 기반하고 있다[11].

애스펙트를 이용하여 웹 공격을 탐지하고 방어하는 연구로 AProSec이 있다[12]. AProSec은 크로스 사이트 스크립팅과 SQL 인젝션 공격을 탐지하여 파라미터의 값을 변환하거나 차단하는 방식을 사용하고 있다. 이때 애스펙트에서는 요청 이벤트를 인터셉트하여 각각의 검증 단계를 거쳐 공격을 탐지하도록 한다. 이때 2단계에 걸쳐 공격으로 분류된 파라미터를 변환하고, 변환이 불가능한 경우 요청을 차단하게 된다. 하지만 AProSec은 크로스 사이트 요청 변조 공격에 대해서는 고려하지 않고 있으며, 아직까지는 관점 지향 프로그래밍 분야에서 크로스 사이트 요청 변조 공격을 방어하는 관련 연구가 발표되지 않았다.

III. 설계

3.1 웹 애플리케이션 구조

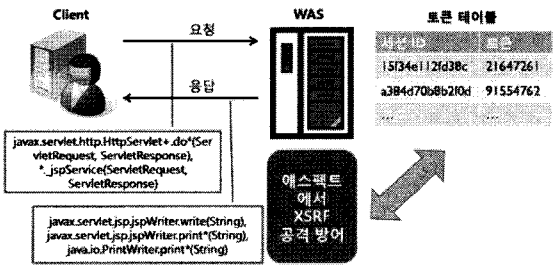

그림 1. 제안하는 보안 모델 구조
Fig. 1. Proposed security model architecture

〈그림 1〉은 제안하는 보안 모델 구조로 웹 애플리케이션 서버에서 관점 지향 프로그래밍 기술을 이용하여 크로스 사이트 요청 변조 공격을 방어하는 모델 구조를 보여준다. 클라이언트는 웹 애플리케이션 서버에 요청을 보낸다. HTTP 요청은 애스펙트에 의해 인터셉트되어 이후 요청 트리거가 되는 하이퍼링크나 해당 폼에 토큰이 삽입되게 된다. 이후에 요청이 들어오면 애스펙트에 의해 다시 인터셉트되고 검증 받게 된다. 이 때 웹 애플리케이션 서버에서는 토큰 테이블을 이용하여 해당 토큰과 세션ID가 일치하는지 검사한다. 토큰 테이블에 일치하는 토큰과 세션ID가 있다면 정상 요청으로 분류되어 애플리케이션에 요청이 전달될 것이고, 그렇지 않으면 크로스 사이트 요청 변조 공격으로 분류되어 요청이 애플리케이션에 전달 되지 않고 거부 될 것이다.

크로스 사이트 요청 변조 공격에 대한 완화 기술을 실제로 적용하기 위해서는 두가지 속성을 만족해야만 한다. 첫째, 매우 낮은 오탐율(false-positive 및 false-negative)로 크로스 사이트 요청 변조 공격을 탐지하고 방어하는데 효율적이어야만 한다. 둘째, 웹 사이트 관리자에게 쉬운 설정을 제공하고, 개발자가 기존 애플리케이션에 대한 수정이 필요하지 않도록 해야 한다. 기존의 접근법은 대부분 두가지 측면 중 최소 하나는 만족하지 못하고 있다. 크로스 사이트 요청 변조 문제에 대한 해결책은 애플리케이션으로부터 필요한 보안 메커니즘의 결합도를 낮추고, 최소의 노력으로 기존의 시스템에 적용될 수 있는 분리된 모듈을 제공하는 것이다. 그러므로 웹

애플리케이션과 대상 애플리케이션간에 요청과 응답을 관점 지향 프로그래밍을 이용하여 처리하는 접근법을 제안한다. 이 접근법은 토큰과 같은 공유된 비밀 값 기술을 적용할 수 있도록 자동으로 애플리케이션을 확장하여 애플리케이션의 요청과 응답을 투명하게 검사하고 변경하는 것이 가능하다.

이 메커니즘에서 필수적인 요구조건은 사용자의 세션에 대해 토큰이 유효하다는 것을 판단할 수 있어야 한다는 것이다. 이것을 해결하기 위해 애스펙트에서는 〈표 1〉과 같이 토큰에 세션ID로 매핑되는 토큰 테이블을 관리한다. 토큰 테이블은 메모리와 CPU 시간을 절약하기 위해 정기적으로 이전 값들을 제거해야만 하며, 이를 위해 테이블에 마지막 사용된 시간을 지정하는 타임스탬프 필드를 관리한다. 이 시간이 설정된 세션 시간보다 더 길어지면 삭제된다. 이는 별도의 배치 작업이나 데몬 프로세스를 통해 수행된다.

표 1. 토큰 테이블
Table 1. Token table

세션ID	토큰	마지막사용시간
15fb4e112fd38c	21747261	2008-07-01 17:35:24
a384d70b8b2f0d	91774762	2008-07-01 17:32:12
.	..	..

3.2 크로스 사이트 요청 변조 공격 방어 프로세스

〈그림 2〉는 요청이 들어왔을 때 애스펙트가 처리하는 단계를 보여주고 있다. 첫번째 단계에서 요청의 세션ID를 검사하게 되는데 이때 세션ID를 포함하고 있는지 여부를 검사한다. 만약 세션ID가 요청에 없다면 요청이 기존에 인증된 세션을 참조하지 않은 것이기 때문에 정상적인 요청으로 분류된다. 그러므로 여기서는 대상 애플리케이션에 요청을 안전하게 전달할 수 있다[6]. 만약 요청이 세션ID를 포함한다면 대응하는 토큰이 이미 존재하는지 여부를 검사하기 위해 토큰 테이블을 조회한다. 만약 해당 세션ID가 테이블에 존재한다면 테이블에 매핑된 토큰을 요청 파라미터에 포함할 것을 요구한다. 이 조건이 만족되지 않은 요청은 크로스 사이트 요청 변조 공격으로 분류되어 요청이 거부된다. 애플리케이션에 의해 생성된 문서는 애스펙트에 의해 인터셉트되어 토큰이 추가 되기 때문에 애플리케이션에 대한 요청에서 세션ID를 사용할 때는 항상 토큰을 포함한다는 것을 보장한다.

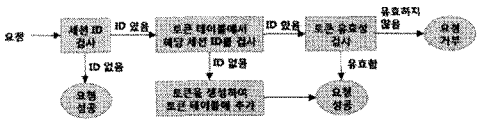

그림 2. 요청에 대한 방어 프로세스
Fig. 2. Request process for preventing

이 프로세스는 관리자의 설정에 기반하여 수행이 된다. 공격으로 분류되었을 때 피해자에게 알리기 위한 경고 메시지를 생성하고, 설정된 페이지로 이동하도록 했다. 이렇게 하면 크로스 사이트 요청 변조 공격이 탐지되었을 때 사용자의 현재 세션을 종료할 필요가 없다. 경고 메시지와 해당 링크 페이지로 이동한 후에 사용자는 정상적으로 작업을 계속 할 수 있다. 즉, 사용자 입장에서 공격이 탐지되었을 때 별다른 학습없이 상태를 파악할 수 있고 추가적인 행위를 요구하지 않고 바로 다음 작업을 수행할 수 있다는 장점이 있다. 요청이 토큰 테이블에는 존재하지 않는 세션ID를 포함하는 경우에 새로운 세션이 생성됐다는 것으로 간주한다. 애스펙트는 새로운 토큰을 생성하여 토큰 테이블에 해당 세션ID와 함께 삽입한다. 그런 후 요청은 해당 애플리케이션에 정상 요청으로 보내진다.

〈그림 3〉은 응답시에 토큰을 생성하여 URL를 수정하는 프로세스를 나타내고 있다. 응답 처리 단계는 웹 애플리케이션의 결과 문서에 토큰을 포함시키기 위한 것으로 URL 재작성과 비슷하다[6]. 유효한 세션에 대해 세션ID가 존재한다면 토큰 테이블을 조회하여 생성된 토큰이 있는지 검사한다. 만약 생성된 값이 없다면 세션이 새롭게 생성되었다는 것을 의미하므로 토큰을 생성하여 토큰 테이블에 해당 세션ID에 대응하여 값을 추가한다. 마지막으로 클라이언트에 애플리케이션의 결과 문서를 반환하기 전에 요청 트리거가 되는 하이퍼링크 및 폼에 토큰 파라미터를 확장한다.

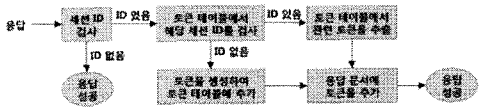

그림 3. 응답에 대한 방어 프로세스
Fig. 3. Response process for preventing

다음 〈표 2〉는 응답 문서에 토큰 파라미터가 추가되어야 할 필드를 정의한 것이다.

표 2. 토큰이 삽입되는 태그 및 속성
Table 2. Tag and attribute added token

태그	속성
a	href
form	action
frame	src
iframe	src
button	onclick
button	refresh
refresh meta	url

IV. 구현

애스펙트는 관점 지향 프로그래밍 프레임워크에 의해 사용될 수 있고 세부분으로 구성된다. 첫째, 애스펙트에서 요청에 대한 크로스 사이트 요청 변조를 탐지하고 방어하기 위해 관리자가 환경 파일상에 선택한 옵션에 의존한다. 둘째, 어드바이스는 요청과 응답 프로세스에서 세션과 토큰을 검사하고 적절한 곳에 토큰을 생성하는 것을 정의한다. 마지막으로 교차점을 정의하여 웹 애플리케이션과의 위빙을 수행한다.

4.1 애스펙트의 구성

〈그림 4〉는 본 논문에서 제안하는 관점 지향 프로그래밍을 이용하여 크로스 사이트 요청 변조 공격을 방어하기 위해 애스펙트에 기반한 보안 기능을 구현한 클래스 다이어그램이다. XSRFIntercept는 어드바이스가 정의된 애스펙트이며, AspectJ인 경우 교차점도 여기서 정의가 된다. 이 애스펙트는 공격 포인트가 되는 부분을 인터셉트하여 크로스 사이트 요청 변조 공격의 탐지 및 차단을 수행하는 클래스의 메서드를 호출하도록 구성되었다. Config 클래스에서는 〈그림 5〉의 환경 구성 정보를 로드하고, 해당 정보를 각 클래스 및 애스펙트에 제공하며, 웹 애플리케이션 서버 및 웹 애플리케이션의 프레임워크에 의존적인 세션에 대해 검사하는 기능을 제공한다. TokenMaker 클래스는 토큰을 생성 및 추출하여 토큰을 비교하고 URL을 재작성하는 기능을 제공하며, XSRFValidator 클래스는 크로스 사이트 요청 변조 공격을 탐지하는 기능을 제공한다.

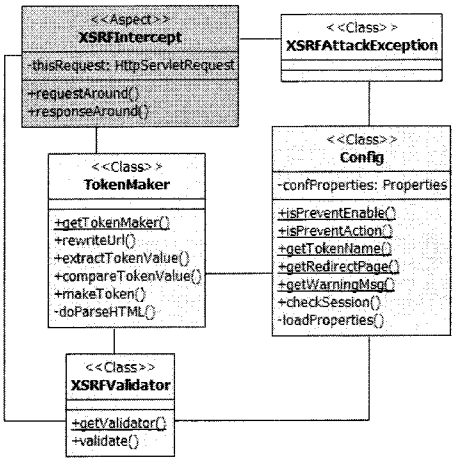

그림 4. 크로스 사이트 요청 변조 보안 기능에 대한 클래스 다이어그램
Fig. 4. Class Diagram for XSRF security functions

관리자가 크로스 사이트 요청 변조 공격 방어에 대한 설정을 할 수 있도록 〈그림 5〉와 같이 XML 형태의 환경 파일을 구성하였다.

```
<?xml version="1.0" ?>
<!DOCTYPE preventor [
<!ELEMENT preventor
(preventEnable, preventAction, tokenName,
redirectPage, warningMsg)>
        <!ELEMENT preventEnable (#PCDATA)>
        <!ELEMENT preventAction (#PCDATA)>
        <!ELEMENT tokenName (#PCDATA)>
        <!ELEMENT redirectPage (#PCDATA)>
        <!ELEMENT warningMsg (#PCDATA)>
]>
```

그림 5. 환경 구성 파일: preventor.xml
Fig. 5. Configuration file: preventor.xml

각각의 기능을 정의하기 위해 다음과 같은 의미로 ELEMENT의 집합을 정의한다.

- preventEnable: 요청/응답에 대한 방어 프로세스를 활성화할 것인지를 설정한다.
- preventAction: 이 기능이 활성화되면 응답시 토큰 삽입과 같은 방어를 위한 파라미터 확장 및 요청시 세션ID와 토큰 테이블의 검사 등을 수행하지만, 공격으로 분류되더라도 요청을 막지 않는다.
- tokenName: 파라미터에 확장될 토큰의 이름을 정의한다.
- redirectPage: 크로스 사이트 요청 변조 공격이 탐지되었을 때 해당 애플리케이션의 기능을 수행하지 않고 여

기서 지정한 페이지로 넘어간다.

○ warningMsg: 크로스 사이트 요청 변조 공격이 탐지되었을 때 사용자에게 보여질 메시지를 정의한다.

토큰을 저장하는 파라미터의 이름은 임의로 선택될 수 있으나, 애플리케이션에서 사용되는 다른 파라미터의 이름과 충돌이 나지 않아야 한다. 토큰의 값은 애스펙트가 관리하는 토큰 테이블로부터 조회된다.

크로스 사이트 요청 변조 공격은 공격자 대신에 권한있는 액션을 수행하는데 피해자가 강제로 악용될 수 있는 세션 정보가 없을 때는 공격 또한 이루어질 수 없기 때문에 애스펙트는 클라이언트가 인증이 됐다는 것을 클라이언트가 유효한 세션을 갖고 있다는 것과 동일하게 처리한다.

〈그림 6〉은 지정된 태그나 속성에 포함된 URL에 토큰 값을 확장하는 애스펙트를 나타내고 있다.

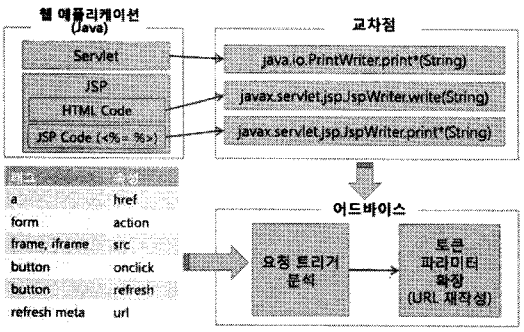

그림 6. AOP를 이용하여 토큰 파라미터 확장
Fig. 6. Extending token parameter using AOP

애스펙트를 이용하여 모든 요청을 인터셉트하기 때문에 어떠한 공격도 놓치지 않는다. 만약 세션이 존재하지 않는다면 애스펙트가 애플리케이션의 문서에 포함된 토큰이 있더라도 이미 세션이 소멸되었으므로 공격으로 분류하지 않아 애플리케이션의 동작에 영향을 미치지 않는다.

4.2 애스펙트의 적용

어드바이스 부분은 크게 두가지로 구성된다. 첫번째는 요청시에 HTTP 요청 파라미터에 대한 유효성을 검증하는 것이고, 두번째는 응답시에 애플리케이션이 생성하는 결과에 대해 토큰 파라미터를 확장하여 URL을 재작성하는 것이다.

위빙을 위한 교차점은 다음과 같이 정의 되었다.

○ doGet(ServletRequest, ServletResponse), doPost (ServletRequest, ServletResponse): 서블릿에서 발생하는 요청 이벤트를 인터셉트하기 위해 정의 되었다.

○ _jspService(ServletRequest, ServletResponse): JSP에서 발생하는 요청 이벤트를 인터셉트하기 위해 정의 되었다.

○ write(String), print*(String): java.io.PrintWriter와 javax.servlet.jsp.JspWriter의 메서드로 응답시 요청 트리거가 될 수 있는 하이퍼링크 및 폼 태그에 토큰을 삽입하기 위해서 정의 되었다.

요청시의 결합점은 컨테이너(container)에 의해 실행이 되는 것이기 때문에 교차점에서 결합점 유형을 메서드 실행(execution)으로 정의하였고, 응답시에는 출력을 하기 위해 해당 메서드를 호출하는 것이기 때문에 결합점 유형을 메서드 호출(call)로 정의하였다.

V. 실험 및 결과분석

JEE 오픈 소스 소프트웨어에 대해서 사용자의 인증이 요구되는 애플리케이션 기능을 이용하여 중요한 데이터를 변경하는 크로스 사이트 요청 변조 익스플로잇을 수행해 보았다. 실험 환경으로 서버 및 클라이언트의 CPU는 Intel Pentium M processor 1.20 GHz이며, 운영체제는 Microsoft Windows XP Professional에 Service Pack 2가 적용된 버전이다. 애플리케이션 서버는 Tomcat 6.0[13]을 사용하였고, JDK 6.0[14] 환경에서 관점 지향 프로그래밍 프레임워크로 AspectJ 1.6.2[15]에서 컴파일시점 위빙을 적용하였으며, AspectWerkz 2.0[16]을 사용하여 실행시점 위빙을 하도록 설정하였다. 또한 웹 애플리케이션의 성능을 측정하기 위해 JMeter 2.3.2[17]를 사용하였다.

실험은 JEE 애플리케이션 오픈 소스 중 프로젝트 관리를 위한 협력 도구인 mvnForum[18]을 대상으로 했다. 이 애플리케이션은 기본적인 계정 관리 기능을 제공하고, 포럼을 생성하여 게시판에 글을 작성하거나 조회할 수 있으며, 다른 사용자에게 공개 혹은 비공개로 메시지를 보내거나 메일을 전송하는 기능으로 구성되어 있다. 실험한 익스플로잇은 〈표 3〉과 같으며 향후 더 다양한 익스플로잇에 대한 연구와 다양한 도메인의 JEE 애플리케이션에서의 실험이 필요하다.

이때 각 애플리케이션을 분석하여 GET 방식의 URL 파라미터를 변조한 링크를 통해 익스플로잇을 실행하거나, POST 방식으로 파라미터를 변조한 별도의 익스플로잇 페이지를 작성하여 공격을 시도하였다. 1차 실험으로 〈표 3〉의 익스플로

잇을 수행한 결과 모두 성공적으로 차단되는 결과를 보였다. 하지만 피해자의 애플리케이션을 보호하기 위해 가장 중요한 것은 애스펙트가 정상적인 애플리케이션 행위를 방해해서는 안된다는 것이다. 이것을 검사하기 위해 2차 실험으로 크로스 사이트 요청 변조 보호를 켜놓은 상태에서 공격이 탐지되었을 때 차단되지 않도록 환경을 설정한 후 (preventAction을 이용) 해당 기능을 수행한 결과와 애스펙트를 적용하지 않고 수행한 결과를 비교해 보았다. 만약 두 결과가 동일하다면 애스펙트가 정상적인 요청의 상황에서 애플리케이션의 동작에 영향을 미치지 않고 올바르게 작동한다는 것을 의미한다. 실험 결과 애스펙트를 적용하지 않았을 때와 적용했을 때 애플리케이션의 실행 결과가 동일하다는 것을 확인하였다.

표 3. 실험에 사용한 익스플로잇
Table 3. Exploits used in experiment

애플리케이션 기능	익스플로잇	결과
포럼 생성 (관리자 권한)	GET 방식의 URL	차단
게시판 글 게시 및 수정	POST 방식의 페이지	차단
메시지 및 메일 전송	POST 방식의 페이지	차단
메시지 및 메일 삭제	GET 방식의 URL	차단
폴더 생성	GET 방식의 URL	차단
계정 정보 변경	POST 방식의 페이지	차단

이렇게 관점 지향 프로그래밍의 애스펙트를 구현하여 애플리케이션을 보호하는 것은 기존 시스템에 대한 소스코드의 변경 없이도 애플리케이션의 보안을 향상시킬 수 있으므로, 비용 및 수정으로 인한 위험부담을 줄일 수 있다는 것을 실험을 통해 알 수 있었다.

성능 실험은 Non-Security(보안 기능을 적용하지 않은 애플리케이션), Non-Aspect(보안 기능을 원래 소스를 수정하여 적용한 애플리케이션), Compile-Time Weaving(AspectJ의 컴파일시점 위빙을 적용하여 애스펙트 보안 기능을 적용한 애플리케이션), Load/Run-Time Weaving(AspectWerkz의 실행시점 위빙을 사용하여 애스펙트 보안 기능을 적용한 애플리케이션)으로 나눠서 수행하였다.

〈그림 7〉의 위에 있는 그림은 단일 스레드 환경에서 각각 100번의 HTTP 요청을 수행한 후 응답시간을 추출한 결과이고, 아래 그림은 그 중에 60ms 미만의 요청 결과를 확대한 것이다. 가로축은 HTTP 요청 횟수를 의미하고 세로축은 요청에 대한 응답시간을 Milliseconds 단위로 나타낸 것이다. 4가지 방식 모두 1번째 요청시에 가장 긴 응답 시간을 보이는데 이는 JEE 환경의 애플리케이션이기 때문에 클래스가 로드되는 시간이 포함되어 있다는 것을 의미한다. 일단 클래스가 로드 된 후에는 일정한 응답시간을 보이고 있지만, Load/Run-Time Weaving은 중간에 1번째와 비슷한 응답시간을 보이는 것을 확인할 수 있다. 이는 실행시점에 애스펙트의 변경사항이 적용되는 지를 확인하기 위해 51번째 요청시에 애스펙트를 재배포했기 때문이다. 새로 적용된 애스펙트에 의해 관련 애플리케이션이 로드 되면서 위빙이 수행되었기 때문에 긴 응답시간을 요구하게 되는 것이다.

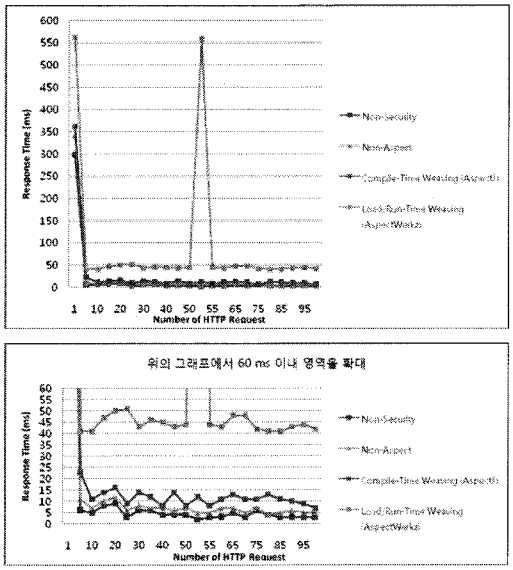

그림 7. 보안 기능 적용 방식별 HTTP 응답 결과
Fig. 7. HTTP response result by methods for applying security functions

Non-Security, Non-Aspect 및 Compile-Time Weaving은 모든 값이 큰 차이를 보이고 있지 않지만, Load/Run-Time Weaving은 상대적으로 긴 응답시간을 보이고 있다.

〈표 4〉는 보안 기능 적용 방식에 따라 상대적으로 평균 응답시간이 얼마나 차이가 나는지를 나타내고 있다. 보안 기능을 애플리케이션의 핵심 기능을 직접 수정하여 적용해 본 결과 보안 기능을 적용하기 전보다 약 1.34배 정도 응답시간이 길어졌다. 하지만 이는 개발자의 능력에 따라 달라질 수 있는 결과이다. 여기서 주목해야 할 부분은 애스펙트를 이용하지 않고 보안기능을 적용한 방식과 위빙을 수행하여 보안기능을 적용한 방식의 비교이다.

표 4. 적용 방식 대비 성능 비교
Table 4. Performance comparison of relative applying methods

기준 방식	Non-Aspect	Compile-Time	Load/Run-Time
Non-Security	1.34x	1.94x	7.22x
Non-Aspect	1x	1.45x	5.41x
Compile-Time	0.69x	1x	3.72x

Non-Aspect에 대해서 Compile-Time은 약 1.45배 정도 응답시간이 길어졌고, Load/Run-Time은 5.41배 정도 응답시간이 길어졌다. 인증을 요구하는 애플리케이션 기능들의 특성상 많은 요청에 의한 부하가 발생하지 않기 때문에 이 수치는 컴파일시점 위빙을 적용한 애플리케이션을 실행했을 때의 지연이 성능에 있어서 크게 영향을 미치지 않는다는 것을 의미한다. 이것은 위빙에 의한 성능의 지연 정도에 비해 기존 코드의 수정에 대한 노력과 위험도가 낮아지고, 보안 기능의 전체적인 적용 및 비적용을 신속하게 할 수 있다는 장점이 의미가 있다는 걸 보여준다. 하지만 Compile-Time과 비교했을 때 Load/Run-Time 은 약 3.72배 정도 응답시간이 길어지는데, 51번째 요청시에 재배포를 하지 않았다고 가정하면 약 3.2배 정도로 크게 줄지는 않는다. 하지만 Load/Run-Time은 시스템을 정지하지 않고도 보안 기능을 적용할 수 있고, 다시 해제할 수도 있는 장점이 있기 때문에 시스템의 상황에 따라 적응적으로 보안 기능을 적용해야 하는 특정 애플리케이션에 대해 부분적으로 적용하면 유용할 것이라고 판단된다.

좀 더 다양한 환경에서 발생할 수 있는 요소들을 고려한 실험이 필요하겠지만, 실제 전체적으로 실행을 해보았을 때 애스펙트에 의해 보호된 애플리케이션이 눈에 띄는 지연이 있다는 것을 관찰하지 못하였으며, 향후 성능에 대한 최적화가 더 이루어질 수 있을 것이라고 판단된다.

VI. 결론

본 논문에서 제시한 크로스 사이트 요청 변조 공격 방어 기법은 AOP를 이용하기 때문에 애플리케이션 내에 공격이 발생하는 지점을 놓치지 않고 보안 기능을 적용하여 공격을 방어할 수 있다. 또한 Java 웹 애플리케이션 환경에서 고유의 세션 관리 정책에 따라 보안 애스펙트 관련 라이브러리의 간단한 수정만으로 적용이 가능하기 때문에 유연성이 높다는 장점이 있다.

실험 결과는 이 보안 모델이 JEE 환경의 레거시 시스템에 적용하는 것이 가능하다는 것을 보여주고 있고, 기존 애플리케이션의 동작에 악의적인 영향을 미치는 것 없이 안전하게 할 수 있다는 것을 보여주고 있다.

현재 크로스 사이트 요청 변조 공격은 웹 개발자 및 공격자에게 비교적 잘 알려져 있지 않다. 하지만 금전적인 이익을 노리는 공격이 증가하는 추세에 있어서 향후 크로스 사이트 요청 변조 공격에 대한 인식도 증가할 것이라고 예상된다.

본 논문에서 제시한 모델도 크로스 사이트 스크립팅 공격이 가능하게 되면 결국 크로스 사이트 요청 변조 공격도 가능하게 되므로, 크로스 사이트 스크립팅 공격에 대한 방어가 이루어졌을 때 완벽하다는 것을 가정한다. 그러나 제시한 모델을 이용하여 크로스 사이트 스크립팅 및 각종 인젝션 공격에 대해 웹 애플리케이션을 보호하는데 적용할 수 있다. 향후 이를 실현하기 위해 본 논문에서 제안한 모델에 더하여 크로스 사이트 스크립팅 및 각종 인젝션 공격을 탐지하고, 공격 코드를 변환하거나 차단하는 연구가 필요하다.

참고문헌

[1] OWASP, TOP Ten Most Critical Web Application Security Vulnerabilities, *http://www.owasp.org*, 2007.

[2] 김영수, 조선구, "그레이박스를 사용한 컴포넌트의 관심사 분리 보안 모델," 한국컴퓨터정보학회논문지, 제 13권, 제 5호, 163-170쪽, 2008년 9월.

[3] G. Kiczales and et al, "Aspect-Oriented Programming," in proceedings of European Conference for Object-Oriented Programming, LNCS Vol. 1241, pp.220-243, 1997.

[4] 박대우, 서정만, "Phshing, Vishing, SMiShing 공격에서 공인인증을 통한 정보침해 방지 연구," 한국컴퓨터정보학회논문지, 제 12권, 제 2호, 171-180쪽, 2007년 5월.

[5] D. Kristol, L. Montulli, "RFC 2965: HTTP State Management Mechanism," *http://tools.ietf.org/rfc/rfc2965.txt*, 2000.

[6] N. Jovanovic, E. Kirda, C. Kruegel, "Preventing Cross Site Request Forgery Attacks," in workshop of the 1st International Conference on Security and Privacy for Emerging Areas in

Communication Networks, pp.1-10, Baltimore, USA, 2006.

[7] Web Application Security Consortium - Web Hacking Incidents Database(WHID), "WHID 2008-10: Chinese hacker steals user information on 18 MILLION online shoppers at Auction.co.kr," *http://www.webappsec.org/projects/whid/byid_id_2008-10.shtml*, 2008.

[8] Cross-site request forgery, *http://en.wikipedia.org/wiki/Cross_site_request_forgery*, 2008.

[9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "RFC 2616: Security Considerations," *http://www.w3.org/Protocols/rfc2616/rfc2616-sec15.html*, 1999.

[10] M. Johns, J. Winter, "RequestRodeo: Client Side Protection against Session Riding," in proceedings of the OWASP Europe 2006 Conference, Report CW448, pp.5-17, Leuven, Belgium, 2006.

[11] Workshop for Application-level Security (AOSDSEC) in the 3rd International Conference on Aspect-Oriented Software Development (AOSD'04),TW387, Lancaster, UK, 2004.

[12] G. Hermosillo, R. Gomez, L. Seinturier, L. Duchien, "AProSec: an Aspect for Programming Secure Web Applications," in Availability, Reliability and Security on the 2nd International Conference (ARES'07), pp.1026-1033, Vienna University of Technology, Austria, 2007.

[13] Apache Tomcat, *http://tomcat.apache.org/*

[14] Developer Resources for Java Technology, *http://java.sun.com/*

[15] The AspectJ Project, *http://www.eclipse.org/aspectj/*

[16] The AspectWerkz Project, *http://aspectwerkz.codehaus.org/*

[17] Apache JMeter, *http://jakarta.apache.org/jmeter/* SourceForge: mvnForum, *http://sourceforge.net/ projects/mvnForum/*[7]

저 자 소 개

선 수 림

2009년 2월 : 동국대학교 컴퓨터공학과 공학석사

관심분야 : 소프트웨어 보안, 적응적 프로그래밍, 웹 공격 방어

이 금 석

2001년 2월 : 건국대학교 컴퓨터공학과 공학박사

1981~현재 : 동국대학교 정보산업대학 컴퓨터공학과 교수

관심분야 : 소프트웨어 품질 평가, 분산 운영체제, 시스템 성능 평가