오픈 소스 웹 방화벽의 발전 방향

신대용, 홍석원, 이명호

명지대학교 컴퓨터 소프트웨어학과

Consideration for Improving Open Source Web Application Firewall

Sin, Dae Yong, Hong, Sugwon, Lee, Myungho

Myongji University

E-mail : dosuser@naver.com, swhong@mju.ac.kr, myunghoi@mju.ac.kr

요 약

웹의 사용의 확대와 함께 웹 어플리케이션에 대한 공격도 증가하고 있으며 그 형태도 다양화하고 지능화하고 있다. 웹 어플리케이션 방화벽은 이러한 공격으로부터 웹 어플리케이션을 보호하기 위한 시스템이다. 본 논문에서는 현재의 웹 어플리케이션 위협 요소 중 취약한 크로스 사이트 스크립팅과 세션 보호의 문제점을 분석하고 해결 방안을 제시한다. 그리고 일반적인 웹 어플리케이션 방화벽이 앞으로 발전되어야 할 방안을 제시한다.

1. 서론

웹의 발전을 이끈 주요한 특징 중 하나는 바로 뛰어난 접근성이다. 이러한 접근성은 오늘날 많은 정보 시스템을 웹으로 구축하게 하였고 일반적인 비즈니스 사이트에서부터 전자 상거래, 금융 서비스까지 웹을 적용하게 되었다.

웹은 더 이상 정적인 페이지가 아니다. 각종 서버 측의 언어들, 자바 스크립트, 그리고 플래쉬는 웹을 매우 동적인 페이지로 만들어 주었으며 각종 서비스들이 가능하도록 하였다. 서버에서 제공되는 이러한 서비스를 통칭하여 웹 어플리케이션이라고 부르고 있다. 해커들은 이런 웹 어플리케이션을 공격하여 웹 사이트를 해킹하고 정보를 훔쳐가거나 금전적인 이득을 취하려고 한다.

이러한 웹 해킹의 증가에 따라 이를 막기 위한 전문 방화벽이 등장 하게 되었고 이를 웹 어플리케이션 방화벽(Web Application Firewall)이라고 부르고 있다.

WAF에 적용되는 보안 기술은 아직 발전 단계에 있다. 본 논문에서는 현재 웹 어플리케이션의 위협 요소 중에서 현재의 WAF가 해결하지 못하고 있는 크로스 사이트 스크립팅(Cross-Site Scripting)과 세션 보호의 문제점을 분석하고 이를 해결하기 위한 방안을 제시한다. 또한 앞으로 WAF가 발전하기 위해서 고려해야할 방안을 제시한다.

2. 웹 어플리케이션의 위협 요소

웹의 발전과 더불어 웹 해킹의 기술 역시 발전하였다. 문제는 해킹 기술이 발전함에 따라 오히려 더욱 공격 방법이 일반화 되고 웹을 통하여 손쉽게 해킹 툴과 관련 자료를 얻을 수 있게 되었다.

웹 해킹이 다른 해킹에 비해 다른 점 중 하나는 많은 트래픽, 로그 속에 묻혀 공격자를 찾기가 어렵다는 것이다. 수많은 트래픽 속에서 공격을 막고 로그중 공격자를 찾아내는 것이 WAF의 역할이다.

WAF는 클라이언트와 웹 서버 사이에서 응용계층(L7) 메시지를 분석하여 웹 어플리케이션을

공격자로부터 보호하며, 웹 어플리케이션 전체에 대해서 동일한 보안 수준을 제공 하는 시스템이라고 할 수 있다.

<표1>은 2007년도 OWASP에서 발표한 10대 웹 어플리케이션의 보안 취약점이다[1]. OWASP TOP 10 항목 중에서 A2, A10의 경우 WAF의 도입으로써 충분히 해결이 가능하다.

이 중에서 A1 크로스 사이트 스크립팅(XSS)과 A7 취약한 인증 및 세션 관리는 아직 완벽한 해답이 없는 상태이며, XSS를 통하여 세션이 훔쳐지는 등 두 가지 취약점은 밀접한 관련을 가지고 있다.

본 논문에서 언급하려 하는 문제점은 다음과 같다 XSS의 경우 실제 공격의 악의가 들어나는 시점이 웹 서버가 아니라는 점을 고려해서 보안을 적용해야 한다. 실제적으로 공격의 중심은 클라이언트에 있다.

세션의 경우, 세션의 소유주를 나타내는 것은 오로지 세션 아이디(ID)로 이루어지고 있다. 이로 인하여 세션이 XSS 혹은 세션 하이재킹(session hijacking) 등으로 세션 ID가 넘겨질 경우 원래의 소유주의 ID를 도용할 수 있는 문제점이 있다.

표1. 2007 OWASP TOP 10 리스트[1]

위험도	공격명
A1	크로스사이트 스크립팅(XSS)
A2	인젝션 취약점
A3	악성 파일 실행
A4	불안전한 직접 객체 참조
A5	크로스 사이트 요청 변조(CSRF)
A6	정보 유출 및 부적절한 오류 처리
A7	취약한 인증 및 세션 관리
A8	불안전한 암호화 저장
A9	불안전한 통신
A10	URL 접속 제한 실패

3. 크로스사이트 스크립팅(XSS)

<표2>는 US-CERT에서 보여주는 2007년 9월의 웹 2.0 관련 취약점들이다[2]. 비동기 요청(AJAX)과 XML은 웹 2.0 구현의 필수적인 기술로써 해당 기술 자체가 웹 2.0으로 인식 될 정도로 웹 2.0과는 분리할 수 없는 관계에 있다.

표2. 일반적인 취약점 유형의 분석[2]

Common Vulnerability Exposures (CVE) 2007년 6-8월	
취약점 유형	CVE 개수
XSS	*267*
ActiveX	*84*
JavaScript	*27*
XML	*17*

표3. 주간 XSS 취약점 발견 갯수[3-5]

기간		10-08	10-01	09-24
위험도	높음	13개	0	0
	보통	5	10	19

<표3>을 통해 볼 때 주당 10개 이상의 XSS 취약점이 발견 되는 것을 알 수 있다[3-5].

여기서 문제 되는 것은 XSS을 이용한 해킹의 핵심이 되는 구현 기술, 웹 2.0을 구현하는 기술 역시 비동기 요청이라는 점이다.

웹 2.0과 그것을 기업에 적용하자는 엔터프라이즈 2.0 등에서 플렛폼은 웹이 되며 그 위에 웹 어플리케이션이 동작하게 된다. 이들 웹 어플리케이션들이 기존의 웹 어플리케이션과 다른 점은 비동기 요청과 XML의 AJAX등을 활용하여 좀 더 풍부한 UI를 제공한다는 것이다.

간단한 예로 최근 상당한 발전을 보인 자바스크립트(javascript)와 브라우저 별 다큐멘트 객체 모델(DOM) 기능을 이용한 위지윅(wysiwig) 에디터 컴포넌트와 구글 맵 등을 들 수 있다.

현재로서는 공개용 WAF 운영 하에 웹 2.0 서비스를 제공 하는 것은 오탐 문제가 생길 경우 상당히 힘들지만 일부 상용 WAF에서는 가능할 수 있다.

<그림 1> 에서 XSS의 공격을 도식화 하여 보여주고 있다. 공격의 진행을 보면 공격자는 대상 사이트의 페이지 A 에 공격 코드를 추가한다. 피해자 1은 대상 사이트에 접속 페이지 A를 보게 된

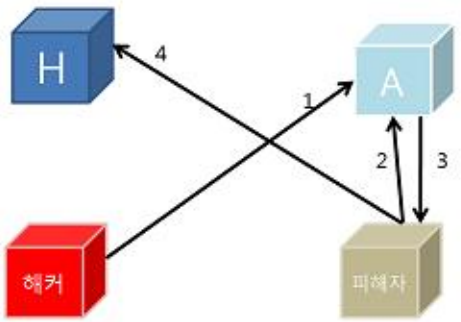

그림 1. XSS 공격

다. 공격 코드는 피해자의 웹 브라우저에 의해 실행된다. 공격 코드는 피해자의 정보를 공격자가 미리 준비해 놓은 서버 H로 전송한다.

이 정보에는 보통 세션 ID등이 포함 되어 있어 공격자가 피해자의 ID를 도용 할 수가 있다. 물론 공격 코드가 사용자의 데이터를 다른 곳으로 전송시키는 것 이외에도 여러 응용이 있을 수 있으며 특히 피해자가 해당 사이트의 관리자일 경우 공격자는 관리자를 도용하여 사이트 전체를 장악할 수도 있다.

이러한 XSS 공격을 막는 가장 확실한 방법은 물론 공격자가 공격 코드를 사이트에 삽입하지 못하도록 하는 것이다. 물론 현재 WAF들은 모두 이러한 기능을 가지고 있다.

대부분의 구현은 패턴 매칭으로 되어 있으며 해당 공격이 들어 왔을 때 공격에 대한 차단을 수행한다. 하지만 만일 공격에 대한 인식과 차단이 이루어 지지 않은 경우 문제가 발생한다.

본 논문에서 제시하는 해법의 기본은 문제 발생 시점과 장소에 기인한다.

웹 해킹의 특성은 공격 의도가 전송부에서는 탐지되지 않고 웹 서버의 웹 어플리케이션에서 일어나는 것이다. XSS 역시 공격의도가 실제 시점이 공격 시점과 다르다.

XSS의 공격 발생 시점은 사용자의 웹 브라우저가 공격 코드가 포함된 페이지를 읽을 때가 된다. 따라서 실제적으로 보안 기능이 들어가야 하는 곳은 웹 브라우저 혹은 클라이언트의 시스템이다.

위의 논리를 기반으로 XSS에 대한 대응책으로 클라이언트 기반의 웹 방화벽이 필요하다. 편의상 WAF와 CWF(Client based Web Firewall)로 나누어 설명 하도록 한다.

CWF는 웹 브라우저 Embed형식과 proxy방식 두 가지로 나뉠 수 있다.

CWF에서 사용 할 수 있는 탐지 방법은 HTML을 분석하여 외부로 데이터를 전송 하는 비동기 요청을 찾고 해당 요청이 대상으로 삼는 서버의 IP 혹은 도메인이 허용 목록 안에 있는지 검사함으로써 공격 코드가 작동하는 것을 막는다.

허용 목록을 관리하는 방법으로는 사용자의 허용자 리스트(white list)와 금지 리스트(black list)를 사용하고 추가적으로 중앙 서버에서 정기적으로 금지 리스트를 내려 받는 식의 구현이 될 것이다.

물론 이 방법을 WAF에서 구현 할 수도 있지만 현재의 요청에 대한 검색에 비해 완성된 페이지 전체에 대하여 검색을 해야 하므로 서비스 효율을 크게 줄이는 원인이 될 수 있기 때문에 바람직하지 않다.

또 다른 방법은 위의 방식에 추가적으로 CWF가 다운로드 받은 웹 페이지의 구조가 일반적인가 검사하는 것이다. 이 기법의 아이디어는 악의적인 코드가 포함된 웹 페이지는 특정 브라우저 전용의 기능 혹은 버그, 비 표준 테그를 이용한다는 점에 있다. 문법 검사를 기본으로 하고 IDS의 프로파일 방식과 같이 페이지의 태그에 사용되지 않는 코드와 있어서는 안 되는 코드 등의 검사를 하게 된다.

XSS 공격에 대한 대응이 완전해 지기 위해서는 무엇보다도 웹 브라우저의 보안성 향상이 필요하다. ActiveX 혹은 기타 추가 기능들로 XSS에 대한 대비책을 마련 한다 해도 브라우저가 직접 보안 기능을 내장 한 것과는 큰 차이가 분명히 존재하기 때문이다.

4. 세션 및 쿠키 보호

XSS의 경우 사용자의 세션 아이디를 해커에게 전송 하는 것이 기본적인 활용이라고 볼 수 있는데 세션 아이디를 해커 이외의 사람이 알 경우 본래의 세션 주인으로 위장할 수 있다.

또한 세션 뿐 아니라 쿠키 역시 아직 해결이 되지 않은 문제이다. 인증, 웹 사이트의 로그인의 경우만 들어도 여전히 쿠키에 사용자가 손대서는 안 되는 정보를 저장하고 있는 웹 사이트들이 많이 존재한다.

우선 쿠키의 문제를 보면 쿠키에는 어떠한 보안 개념도 들어있지 않고 단순히 웹 페이지와 웹 페이지 간의 상태를 저장하기 위한 용도로 고안된 것이기 때문에 별도의 보안 기능이 없다.

하지만 분산 서버 환경에서 세션 만을 가지고 사용자 로그인은 구현하기에는 상황에 따라서 어려울 수 있다. 이 때문에 쿠키값에 사용자 정보를 저장하고 암호화하여 사용하는 경우가 많다.

WAF는 위를 토대로 세션 관리에 보완해야 할 점이 있다. 먼저 세션 ID를 훔쳐 갔어도 원래의 사용자가 아니면 사용할 수 없도록 해야 한다. 이를 위해서 기본적으로 메시지 인증 코드(MAC)가 첨가되는데 여기에 사용자를 확인 할 수 있는 다른 정보들이 들어가야 한다. 또한 세션키(session key)의 라이프 타임을 가능한 짧게 설정해야 한다.

설정에 따라서 쿠키를 보호하기 위해 서버 앞단에서 쿠키의 내용을 모두 암호화 해주는 역할 역시 수행 할 수 있어야 한다.

5. WAF의 발전을 위한 고려사항

5.1 탐지 방법의 변화

기본적으로 WAF에서는 공격 탐지 방법으로 시그니쳐(signature)의 패턴 매칭(pattern matching)을 사용한다. 패턴 매칭 방법은 구현이 용이하고 시그니쳐가 존재할 경우 확실하게 공격에 대응 할 수 있지만 제로데이 공격에 약한 면모를 가지고 있다. 이 문제를 해결하기 위해서 대부분의 상용 WAF에서는 프로파일(profiling) 방식이 같이 사용되고 있으며 오픈 소스 WAF 역시 최신의 버전에서는 프로파일 기능을 추가할 필요가 있다.

5.2 처리 속도

대다수의 방화벽과 마찬가지로 WAF도 역시 처리 속도의 감소를 초래한다. 감소된 속도만큼의 보안성을 확보 할 수 있을 경우 WAF 도입은 타당성을 갖지만 그렇지 않은 경우도 많이 존재한다. 오픈 소스 WAF에서는 비용 문제가 없는 만큼, 처리 속도의 감소폭을 어떻게 줄일 수 있을지가 도입의 문턱을 낮출 수 있을 것이다.

소프트웨어로 구현된 모듈(module) 타입의 WAF들의 경우에는 웹 서버의 각 주요 동작 시점에 보안 관련 기능만을 추가 하는 방식이기 때문에 소프트웨어 방식임에도 불구하고 속도의 저하가 크지 않다. 또한 어플라이언스(appliance)의 경우 하드웨어로 제작되어 빠르지만 보안 장비들을 잘못 구성 하였을 경우 중복과 불필요한 검사로 인하여 속도의 저하를 가져오므로 주의해야 한다.

처리 속도 감소폭을 줄일 수 있는 다른 방법은 WAF처럼 응용 계층의 메시지를 해석해야 하는 보안 소프트웨어 혹은 기기와 병합 하는 것이다.

Unified Treat Management(UTM)이 이 경우에 해당한다. UTM은 방화벽, 안티 바이러스, WAF, IPS 등의 기기를 하나의 기기 혹은 어플리케이션으로 막는다는 개념을 가진 제품이다.

각 보안 기능들을 하나로 통합함으로써 패킷이 전달되는 시간을 줄이며 패킷을 한번만 열어보아도 어플리케이션 레벨의 검사를 모두 할 수 있도록 한다. 더욱이 하위 계층과 상위 계층에서 중복 검사 하는 부분들을 줄일 수 있다. 이것은 보안 기능들을 통합으로 관리의 편의성까지 제공 하고 있다.

5.3 소스코드 분석

WAF의 정의에서 "웹 어플리케이션 전체에 대해 동일한 보안 수준을 제공 할 필요가 없는 경우에는 어떠한가?"라는 질문을 생각 할 수 있다.

사실 웹 어플리케이션 전체에 대해 동일한 보안 수준을 제공할 필요는 현실적으로 없을 것이다. 어느 부분은 단지 뷰어(viewer)의 기능만을 하고 어떤 부분은 순수한 HTML일 수 있다. 또한 어떤 부분은 다른 부분에 비해 좀 더 세심한 보안을 필요로 한다.

대부분의 웹 어플리케이션 작성에 사용되는 언어들이 스크립트 언어이기에 컴파일된 프로그램에 비하여 소스 코드 분석이 용이하다. 소스 코드 분석을 이용하면 보안에 문제가 있는 부분을 발견하고 수정하는 것 이외에 WAF가 어떤 페이지를 보호해야 하는지의 대상 설정(targeting)도 최적화를 할 수 있다. 이러한 대상 설정이 제대로 이루어지면 좀 더 최적화된 감시를 할 수 있고 이것이 속도 증가로 이어 질 수 있다.

예를 들면, 무차별 공격(Bruteforce)의 경우 모르는 URL(unknown URL)에 대한 검색의 경우에도 사용되지만 로그인 등에도 사용 된다. 만약 로

그인 모듈이 여러 곳에 퍼져 있거나 한 서버에서 많은 사이트가 운영 중일 때 소스 코드 분석을 통해서 필요한 보안 기능을 특정 페이지에 제공하기가 수월해 진다.

표2. 호스팅 업체와 SI업체의 WAF 요구사항 비교

	호스팅	SI
복잡도의 관리	X(부분적)	O
코드의 관리	X	O
동일 수준의 보안	X	O
보안기능 삽입 용이	X	O
WAF의 적용	필수적	비효과적

5.4 엔터프라이즈(Enterprise) 환경

기본적인 기능만을 봤을 때 WAF는 결코 엔터프라이즈 환경에 어울리지 않는다. 웹 어플리케이션의 설계가 잘 되어있고 보안 사항을 고려하여 제작되어 보안 기능의 추가가 용이 하다면 WAF의 모든 기능을 적용할 필요는 없다.

표2는 WAF이 필요로 하는 관리적 문제 네 가지를 나열하고 있다. 보안 기능을 나열하지 않은 것은 WAF로 기능을 만족하느냐 웹 어플리케이션 자체에서 구현 하느냐는 문제가 될 뿐 양쪽 모두 해당 기능이 필요하기 때문이다.

이 표에서 일괄적으로 관리할 수 없는 사이트 등의 경우를 호스팅 업체라고 하였다. WAF는 호스팅 업체를 이용하는 사람들 모두에게 일관적인 보안기능을 제공하는 것이 가능하다.

하지만 SI업체의 제품 같은 경우 보안과 확장을 염두에 두고 개발했다면 이미 일관적인 보안 기능이 제공 되고 있을 것이라는 것을 전제로 한다.

따라서 SI제품에서는 웹 서버의 보안 기능을 향상 시키는, 분할 요청, 버퍼 오버플로우(buffer overflow) 방지 등의 웹 어플리케이션에서 막기 힘든 부분에 대한 기능만을 제공 되는 것이 더 효율적일 것이다. 하지만 이 경우 사실상 하나의 어플리케이션이나 장비로써 존재할 가치가 없어진다. 이는 IPS, 방화벽등 기존의 장비에 추가적인 확장을 통해서 충분히 커버 할 수 있다.

물론 호스팅 업체들 같이 웹 어플리케이션의 관리가 어려운 곳일 경우 WAF의 도입은 보안성 향상에 충분한 도움이 된다.

6. 결론

본 논문에서는 향후 WAF가 발전해야 할 방향을 제시하였다. 결국 문제는 안정성 있는 웹 어플리케이션의 개발과 웹 브라우저의 보안성 향상으로 귀착된다. WAF가 앞으로 기술적 성숙을 이어나갈지 혹은 웹 보안의 과도기적 제품으로 그칠지는 좀 더 지켜볼 필요가 있을 것이다.

WAF가 등장한지 약 2년 정도가 지났지만 아직 많은 사람들에게 WAF의 개념은 생소한 편이다. 앞으로 관리적 측면에서 한 서버에서 여러 사이트를 운영하는 사이트에서 WAF의 기능은 필수적인 요구사항이 될 것이다.

[참고문헌]

[1] OWASP Foundation, 10대 가장 심각한 웹 어플리케이션 보안 취약점,
http://www.metasecurity.org/owasp/OWASP_Top_10_2007_Korean.pdf

[2] US-CERT Quarterly Trends and Analysis Report, Vol. 2, Issue 3, September 1, 2007
http://www.us-cert.gov/press_room/trendsanalysisQ307.pdf

[3] US-CERT, Vulnerability Summary for the Week of October 8, 2007
http://www.us-cert.gov/cas/bulletins/SB07-288.html

[4] US-CERT, Vulnerability Summary for the Week of October 1, 2007
http://www.us-cert.gov/cas/bulletins/SB07-281.html

[5] US-CERT, Vulnerability Summary for the Week of September 24, 2007
http://www.us-cert.gov/cas/bulletins/SB07-274.html