높은 이식성을 제공하는 향상된 웹 서비스 보안 시스템 설계 및 구현

문창현, 박지수, 박종혁*
서울과학기술대학교 컴퓨터공학과
e-mail:{ckdgus2482, jisoo08, jhpark1}@seoultech.ac.kr

Design and Implementation of Advanced Web Services Security System With High Portability

Chang-Hyun Moon, Ji Soo Park, Jong Hyuk Park
Seoul National University of Technology

요 약

현대사회에서 웹 서비스의 대중화로 인한 정보누출의 위험성이 증가하고 있다. 웹 서비스에 대한 보안 사고를 줄이기 위해서는 웹 서비스 제공자에게 취약점을 파악하고 대처하는 능력이 요구된다. 하지만 공개된 취약점에도 보안이 되지 않은 웹 서비스들이 다수 존재하고 새로운 해킹기법들의 등장에 따른 새로운 보안솔루션 연구가 요구된다. 본 논문에서는 사용자의 입력값을 검증하여 무분별하게 입력되는 Script 공격을 방어하는 높은 이식성을 제공하는 향상된 웹 서비스 보안 시스템을 설계 및 프로토타입을 구현한다.

1. 서론

최근 정보보안에 대한 관심이 높아지면서 웹서비스 제공자는 안전한 서비스를 제공할 의무를 갖게 되었다. 그러나 웹서비스 제공자가 보안 기술을 적용하기에는 시간적 기술적 금전적 어려움이 크다. 본 논문에서는 웹서비스 개발 후 서비스 시작시 발생할 수 있는 보안 이슈들에 대해 살펴본다. 또한 이러한 웹 서비스의 개발 및 서비스에서 발생하는 어려움과 보안 취약점을 해결하기 위한 모의해킹이 가능한 테스트베드를 구축한다. 마지막으로, 웹서비스 제작 시 발생할 수 있는 보안 취약점에 대한 보고서를 작성하여 이러한 취약점을 해결할 수 있는 보안 모듈의 프로토타입을 설계 및 구현한다.

2. 관련 연구

2.1 웹 서비스 보안 이슈

The Open Web Application Security Project(OWASP)에서는 웹 서비스에서 제공되는 웹 어플리케이션의 주요 보안이슈를 10가지로 정리하고 공격 시나리오와 대응방안을 구성하여 발표하고 있다. OWASP은 2010년도에 Top 10으로 Injection, Cross Site Script(XSS), 취약한 인증과 세션관리, 안전하지 않은 직접 객체 참조, Cross Site Request Forgery(CSRF), 보안상 잘못된 구성, 안전하지 않은 암호저장, URL 접근 제한 실패, 불충분한 전송 계층 보호, 검증되지 않은 리다이렉트와 포워드를 발표했다 [1].

이중 1, 2위를 차지하고 있는 Injection을 포함한 몇몇 공격 기법들을 분석, 비교하였다.

- **Sql Injection** : 로그인 폼 같은 데이터베이스에 전송할 수 있는 곳에 악의적인 sql구문을 삽입하여 sql질의가 실행되게 하는 기법이다. 이는 사용자 입력 폼에서 받은 문자열에 sql구문이 있는지 검사하여 부적절한 입력값을 걸러냄으로써 막을 수 있다. 또한 sql오류메시지를 감추는 것도 필요하다 [2,3,4].
- **XSS(Cross Site Script)** - 공격자가 입력 폼 또는 url에 client side script를 삽입함으로써 해당 페이지의 파라미터값이나 쿠키값 등을 조회 할 수 있고 js파일 등의 파일 업로드와 연계하면 웹 사이트의 변조, 악의적 콘텐츠 삽입, 피싱 등으로 이어지게 된다. 이 또한 입력 폼에서 받아온 문자열에 html태그나 javascript이 있는지 검사하여 걸러내면 막을 수 있다 [2,3,4].
- **File Upload공격** - 공격자가 대상 웹 서버에 악의적인 웹 스크립트 파일을 업로드하고 정상적인 해당 게시판에서 이미지파일 따위의 경로를 보고 웹 스크립트 파일의 경로를 추측하여 접근함으로써 웹서버자원에 마음대로 접근하거나 경우에 따라서 시스템해킹으로 이어질 수 있다. 이는 업로드시 업로드파일의 확장자를 검사함으로써 어느 정도 막을 수 있으며 업로드된 파일의 경로를 추측하지 못하도록 웹페이지에 로드되는 파일의 경로를 감춰야 한다 [2,3,4].

2.2 웹 취약점 방어기술 연구

* 교신저자 : 박종혁 교수(서울과학기술대학교)

jsp/servlet기반 웹 어플리케이션은 웹 컴포넌트끼리 요청객체를 넘겨주면서 동작한다. 웹 브라우저에서 입력받은 데이터도 결국은 요청객체에 저장되어 데이터처리를 담당하는 웹 컴포넌트에게 전달된다.

javax.servlet.Filter 인터페이스를 구현하면 각각의 웹 컴포넌트끼리 요청이 있을 때마다 그 사이에서 동작하는 Filter를 만들 수 있다. Filter는 요청객체에 저장된 파라미터값을 변조할 수는 없으므로 추가적으로 Wrapper도 필요하다. Wrapper에서 HttpServletRequest객체를 파싱할 때 쓰이는 getParameter, getParameterValues, getParameterMap 메소드를 오버라이딩 함으로써 이 메소드에서 문자열 검증을 담당하는 함수를 호출하여 입력값 검증을 하도록 하였다. Wrapper객체는 Filter에서 다음 필터체인으로 전달되며 결국 변형된 데이터를 포함한 request 객체가 다른 웹 컴포넌트로 도달한다 [5,6].

3. 향상된 웹 서비스 보안 시스템

Sql Injection, XSS를 비롯하여 url점핑 파일 업로드 등의 공격이 모두 클라이언트로부터 입력된 파라미터 값에 대한 검증의 부재로 인한 취약점을 공략하고 있다. 따라서 모든 웹 컴포넌트 사이에서 오고가는 파라미터 값에 대해 필터링을 하는 보안 모듈을 설계한다.

이식성을 고려하여 쉽게 설치할 수 있는 Filter 및 Wrapper클래스를 활용한다.

3.1 구현환경 및 구성

(1) 운영체제 : Windows2008/XP/7
(2) 웹 서버 : Apache Tomcat 7.0
(3) DBMA : My-SQL 5.5
(4) 구현언어 : JAVA, JSP, XML

보안 모듈의 구성도는 다음과 같다. 데이터를 처리해주는 Business Logic과 Client의 웹브라우저 사이에 Parameter Filter가 존재하여 입력 값의 적절성을 검증해준다.

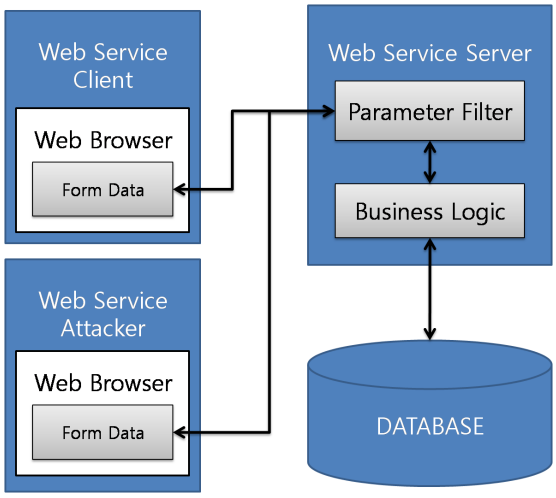

그림 1. Service Model Architecture

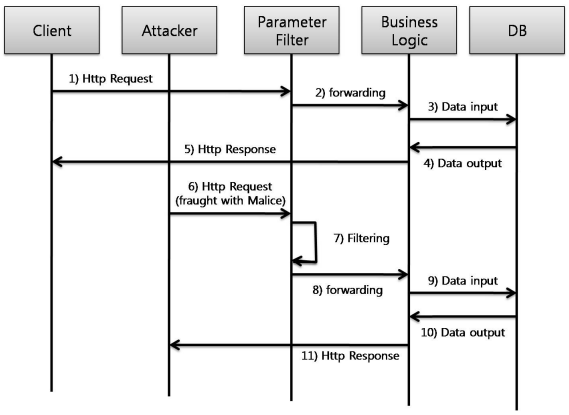

그림 2. Service Model FlowChart

그림 2는 서비스 모델에 대한 흐름도를 나타내고 있다. Parameter Filter는 Http Request를 통해 전달된 Parameter에 대해 내장된 패턴에 따라 적절한지 판단한 후 BusinessLogic에게 forwaording을 한다.

3.2 기능

검증 로직은 특수문자에 대한 검증로직과 문자열에 대한 검증로직이 다른 방식으로 동작하는데 전자의 예를 들자면 '<'같은 문자가 replaceAll메소드를 이용하여 "<"와 같이 바뀌도록 하였다. 이와 같이 하면 '<'가 태그를 여는 기능을 하는 게 아니라 단순한 문자로 인식되어 XSS의 공격을 막을 수 있다. 후자는 대·소문자에 상관없이 필터링 기능을 제공하나 복잡성이 증가하였다. 먼저 문자열에 대해 toUpperCase메소드를 호출하고 반환된 문자열에 대해 indexOf("[걸러내고자 하는 문자열(대문자)]") 메소드를 호출하여 반환된 값을 int형 자료형에 넣어 그게 -1이 아닌지를 검사한다. 다음은 필터링할 문자열을 value라고하고 걸러낼 문자열을 "script"~"SCRIPT"라고 했을 경우의 예시이다.

```
int cnt;
StringBuffer sbf = new StringBuffer(value);

while((cnt=((value.toString()).toUpperCase()).index
Of("SCRIPT"))!=-1)
{
   sbf = sbf.delete(cnt,cnt+6);
   value = sbf.toString();
}
```

현재 필터링 패턴은 ', \\(, \\), <, >, alter, script 등이며, alter, script와 같이 정상적인 입력일 가능성이 있는 것은 걸러내지 않도록 개선중이다.

3.4 프로토타입 구현

필터링 적용 전후의 차이를 회원가입 폼을 통해 테스트

한 비교 결과는 다음과 같다.

- **필터 적용 전**

아이디에 <ScripT>aa를 입력하면 스크립트가 그대로 전달되어 데이터베이스에 그대로 입력되는 것을 확인할 수 있다.

그림 3. 필터 적용전 입력데이터

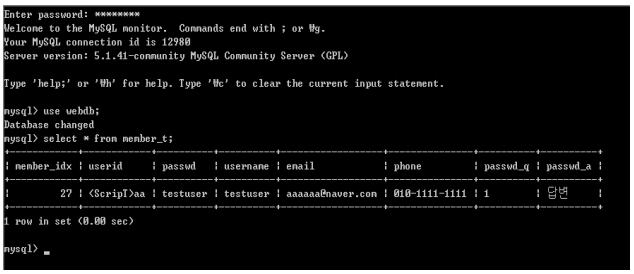

그림 4. 필터 적용 전 입력결과

- **필터 적용 후**

다음으로 필터를 적용한 후 유사한 경우에 대해서 테스트 하였다. 아이디에 <ScriptT>bb를 입력하고 데이터베이스를 확인하면 <는 <로 >는 >로 바뀌었고 ScripT라는 문자열은 사라져 스크립트가 제거된 것을 확인 할 수 있다.

그림 5. 필터 적용후 입력데이터

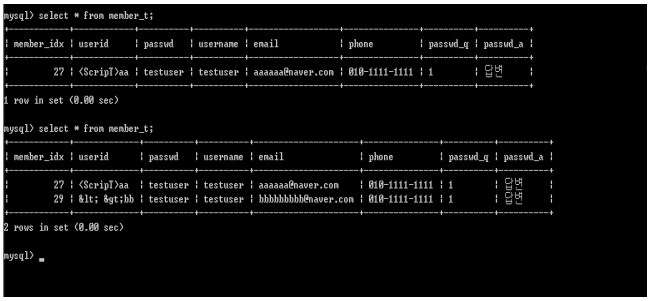

그림 6. 필터 적용 후 입력결과

<는 <로 >는 >로 바뀌었고 ScripT라는 문자열은 사라진다.

4. 결론 및 향후 계획

본 논문에서는 웹 서비스에 대한 보안위협이 증가하는 현황에 따라 여러 웹 서비스에 쉽게 적용할 수 있는 보안 모듈을 제안하고 그 프로토 타입을 설계 및 구현 하였다. 그러나 경우에 따라 Http Request객체가 아닌 MultipartRequest 등의 객체가 파라미터를 전달하는 경우도 있어 이에 대한 필터링은 동작하지 않는 한계를 갖는다. 향후 모의 해킹 및 기술 연구를 통해 본 제안시스템을 향상시킬 예정이다.

MultipartRequest객체로 파라미터를 전달하는 경우에 있어서도 필터링이 작동하도록 관련모듈을 추가할 것이며 지속적으로 필터링 패턴을 추가하여 보다 발전된 시스템을 구현할 예정이다.

참고문헌

[1] The Open Web Application Security Project (OWASP), https://www.owasp.org/

[2] 이인용, 조재익 , 조규형, 문종섭, "SQL 질의 애트리뷰트 값 제거 방법을 이용한 효과적인 SQL Injection 공격 탐지 방법 연구", 정보보호학회논문지 제18권 제5호, pp.135-147, 2008.10.

[3] 고훈, "최근 해킹기법에 대한 분석 및 대응 방법", 인터넷정보학회지 제5권 제1호, pp.37-42, 2004.3.

[4] 송진영, 박대우, "취약점 분석을 통한 Web Site 해킹 연구", 한국정보통신학회, 2010.

[5] 신명훈, "ASP 웹 어플리케이션 보안 취약점 분석 및 대응책 연구", 건국대학교(석사학위논문), 2010.

[6] 김윤명, "뇌를 자극하는 JSP&Servlet", 한빛미디어, 2010.