웹 기반의 보안 취약점 분석과 대응방안

김선용* · 김기환** · 이훈재***

*동서대학교 정보보안트랙

**동서대학원 유비쿼터스 IT

***동서대학교 컴퓨터공학부

Analysis and response of Petya to Ransomware

Seon-Yong Kim* · Ki-Hwan Kim** · Hoon-Jae Lee***

*Dept. of Information and Communication Engineering, Dongseo University

**Dept. of Ubiquitous IT Graduate School of Dongseo University

***Dept. of Computer Engineering, Dongseo University

E-mail : itkindyong@naver.com, ghksdl90@naver.com, hjlee@dongseo.ac.kr

요 약

전 세계에서 웹이 대중화되면서 서로 간에 쉽게 정보를 제공할 수 있어 개인 간에 혹은 회사 간에 정보 공유가 수월해진 장점이 있지만 개인에게 중요한 정보나, 회사의 보안 문서를 노출당하는 경우가 많이 발생하고 있다. 본 논문에서는 웹에서 어떠한 취약점으로 인해서 해킹이 발생하는지와 취약점을 어떻게 대응할 것인지에 대한 방안을 제시할 것이다.

ABSTRACT

The web is used in various ways such as shopping, news, and searching through a web browser. As the Web becomes more and more common, it is often the case that someone is trying to steal personal information or confidential documents from a company, so security must be paid to ensure security on the web. For this reason, you should be aware of the vulnerabilities that are being exploited maliciously in your web applications and improve security with secure coding. In this paper, we propose a method of detecting hacking and how to deal with vulnerabilities due to some weak points on the web.

키워드

Web, Browser, Internet

Ⅰ. 서 론

많은 일들이 인터넷을 통해 일어나고 있다. 이전보다 더 많은 사람들이 인터넷으로 물건을 사고, 인터넷뱅킹을 이용하며, 여가를 즐긴다. 모바일이 대중화되면서 더욱 더 많은 웹 이용하는 빈도수가 증가했다. 하지만 이러한 이유로 눈에 보이지 않는 보안적인 문제가 더 중요시 되고 있다.

2장에서는 일반적으로 빈번히 일어나는 웹 공격에 어떠한 공격기법이 있는지 알고 어떤 취약점이 발생할 수 있는지에 대해 기술한다.

3장에서는 취약점에 의해 일어난 공격을 어떻게 대처할 것인지 대응방안을 제시한다.

Ⅱ. 웹 애플리케이션 공격기법

2장에서는 웹 공격에 어떠한 공격기법이 있는

지 알고 어떤 취약점이 발생할 수 있는지에 대해 기술한다.

2.1 SQL Injection

SQL Injection은 웹 애플리케이션에서 데이터베이스로 전달되는 SQL 쿼리 값을 변조 및 삽입을 통해 비정상적인 접근 방법으로 데이터베이스에 접근하는 공격기법이다. 이 기법을 통해 인증우회나 시스템 명령어 삽입을 통한 서버해킹, 웹쉘 생성 등의 취약점들을 발생시킨다.

2.2 XSS

외부에서 받는 값 중에서 신뢰할 수 없는 값을 철저한 검증 과정 없이 응답페이지로 전송되어 발생하는 공격으로, 사용자의 중요정보를 가로챌 수 있다. 이 기법은 Reflected XSS와 Stored XSS가 있다. Reflected XSS는 사용자가 입력한 데이터가 웹 애플리케이션의 응답페이지로 전송되어 실행되는 취약점이다. 반면에 Stored XSS는 사용자의 입력데이터가 저장된 데이터베이스에서 해당 값을 실행하는 경우에 취약점이 발생하게 된다.

2.3 File Upload

게시판 등에 웹 서버에서 실행 가능한 확장자를 가진 악의적인 파일인 html, jsp, php 등의 파일을 업로드하여 파일확장자에 대한 검사가 수행되지 않는 경우 취약점이 발생하여 시스템 권한을 장악하는 기법이다.

2.4 File Download

게시판에 첨부된 파일을 사용자에게 제공하는데 다운로드 파일의 위치에 제한 조건을 부여하지 않아 지정된 파일 이외의 위치에 있는 파일들에 접근하거나 다운로드할 수 있는 취약점이 있다. 사용자에게 제공하는 방식으로는 정적과 동적인 방식이 있다. 정적방식은 디렉터리에 파일 링크를 걸어 사용자에게 제공하기 때문에 파라미터 변조가능성이 없는 반면에 동적방식의 경우 파라미터 값을 조작하여 시스템 파일 등의 접근 시도가 가능하다.

2.5 CRLF

사용자가 로그인한 웹 브라우저를 통해, 세션 및 다른 인증정보를 서버로 HTTP 요청하여 정상적인 요청처럼 인식하도록 하는 기법으로, 물품구매나 인터넷뱅킹 등의 악의적인 행동을 하는 취약점을 의미한다.

III. 시큐어 코딩

3.1 SQL Injection 보안 코딩

쿼리 값을 변조 및 삽입하여 비정상적인 방법으로 데이터베이스에 접근한다. 다음 그림1은 그러한 접근이 쉬운 소스코드로 보안에 매우 취약하다.

```
String query = "SELECT account FROM user_data WHERE user_id = "
   + request.getParameter("_Id");

try {
    Statement statement = connection.createStatement( … );
    ResultSet results = statement.executeQuery( query );
}
```

그림 1. Injection 보안에 취약한 소스코드

SQL Injection 공격이 발생하지 않도록 하기 위해 보안 코딩을 한다면 쿼리 값을 점검하여 불필요한 값이 있을 시 필터를 해줘야 한다.

```
String _Id = request.getParameter("Id");
String query = "SELECT account FROM user_data WHERE user_id = "
   + _Id.replace("'", "");

try {
    Statement statement = connection.createStatement( … );
    ResultSet results = statement.executeQuery( query );
}
```

그림 2. Injection 보안 코딩한 소스코드

3.2 XSS 보안 코딩

```
<script>
var pos = document.URL.indexOF("name=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</script>
```

그림 3. XSS 보안에 취약한 소스코드

위 그림3에서 document 오브젝트가 외부입력값을 받아 document.URL.substring을 통해 특정 위치를 지정하여 해당 값을 출력하는 방식이다. 이 구조에서 name이라는 변수에 공격코드인 XSS 문자열을 삽입하면 실행이 된다. 이러한 경우는 서버로 전송되는 값을 읽어서 XSS에 대한 공격코드가 삽입되어 있는지 감지하고 삽입되어 있다면 해당 작업을 수행하지 않는 구조로 코딩한다.

3.3 File Upload와 File Download 보안 코딩

File Upload 취약점은 업로드 하는 파일이 서버에서 실행되는 파일인지 아닌지를 검사하고 실행 파일일 경우 업로드할 수 없도록 업로드 취소를 수행할 수 있도록 취한다.

```
if(fileName.toLowerCase().endsWith(".jpg") ||
   fileName.toLowerCase().endsWith(".jpeg") ||
   fileName.toLowerCase().endsWith(".png") ||
   fileName.toLowerCase().endsWith(".gif") ||
   fileName.toLowerCase().endsWith(".bmp"))
```

그림 4. File Upload 보안에 취약한 소스코드

File Download은 동적인 방식으로 소스 코드를 구현할 경우 파라미터 값을 변조하여 피해가 발생할 수 있으니 파일에 링크를 걸어서 접근하는 정적인 방식을 사용해야 한다.

3.4 CRLF 보안 코딩

요청된 매개변수의 값이 나타나는 각 응답헤더에 대해 애플리케이션이 URL 인코딩된 CR, LF 값인 0x0D와 0x0A를 필터링하지 않아 발생하기 때문에 응답헤더에 있는 값을 필터링해주면 취약점이 발생하지 않는다.

```
public class S113 extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
throws IOException, ServletException {
response.setContentType("text/html");
// 사용자 정보를 읽어온다.
String author = request.getParameter("authorName");
if (author == null || "".equals(author)) return;
// 헤더값이 두개로 나뉘어지는 것을 방지하기 위해 외부에서 입력되는 \n과 \r 등을 제거한다.
String filtered_author = author.replaceAll("\r" , " ").replaceAll("\n" , " ");
Cookie cookie = new Cookie("replidedAuthor", filtered_author);
cookie.setMaxAge(1000);
cookie.setSecure(true);
// 생성된 쿠키를 브라우저에 전송해 저장하도록 한다.
response.addCookie(cookie);
RequestDispatcher frd = request.getRequestDispatcher("cookieTest.jsp");
frd.forward(request, response);
}
}
```

그림 5. CRLF 안전한 소스코드

V. 결 론

본 논문에서는 웹 애플리케이션에서 발생하는 웹 공격을 알아보고 그에 맞는 대응방법을 제시한다. 일상에서 평소와 다르지 않게 웹 브라우저를 통해 인터넷을 사용하고 있는데 이러한 취약점들에 의해 갑작스러운 피해를 겪을 수 있다. 앞으로 더 많은 웹의 취약점이 생겨날 수 있다. 그렇기 때문에 웹 보안에 대해 관심을 가지고 보안에 신경을 써야한다.

감사의 글

이 논문은 2016년도 정부(교육과학기술부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(과제번호: NRF-2016R1D1A1B01011908). 또한 부산광역시에서 지원하는 BB21 과제에서 지원받았음.

참고문헌

[1] 최경철, 김태한 “웹 모의해킹 및 시큐어코딩 진단가이드”

[2] wikipedia, “SQL Injection", "XSS", "CRLF" :http://wikisecurity.net/guide:java_%EA%B0%9C%EB%B0%9C_%EB%B3%B4%EC%95%88_%EA%B0%80%EC%9D%B4%EB%93%9C