# **크로스 사이트 스크립팅 대응방안**

## 개요

웹 페이지에 악의적인 스크립트를 포함시켜 불특정 다수의 사용자들에게서 동작 시키는 공격인 크로스 사이트 스크립팅은 비교적 쉬운 테스트 코드로 인하여 쉽게 대응할 수 있을 것이라 생각할 수 있지만 상당한 복잡성을 가지고 있다. 이에 크로스 사이트 스크립팅 필터를 제작하여 대응방안을 자세히 알아보고자 한다.

## 보안대책

외부입력 값에 삽입된 스크립트 구문 작성 시 사용되는 특수문자를 HTML 인코딩 문자로 치환하여 스크립트 구문의 동작을 막는다.

### 별도의 제약사항 없이 모든 태그 무력화

#### 코드 예제

|  |  |
| --- | --- |
| package webfilter;  import java.util.Scanner;  import java.util.regex.Pattern;  import java.util.regex.Matcher;  import org.apache.commons.lang3.\*;  public class main {    public static void main(String[] args) {  String message;  String escapeMessage;  String unescapeMessage;  Scanner scan = new Scanner(System.in);    while(true){  System.out.println("input message : ");  message = scan.nextLine(); // 테스트 값 입력    System.out.println("Original : " + message);  **escapeMessage = escape(message);**  System.out.println("Escape : " + escapeMessage);  **unescapeMessage = unescape(escapeMessage);**  System.out.println("Unescape : " + unescapeMessage + "\n");  }  }    public static String **escape**(String value) {  Pattern escapePattern = Pattern.compile("'");    if (value == null) {  return null;  }  String trustValue = StringEscapeUtils.escapeHtml4(value); // &lt;, &gt;, &quot; 문자 치환    Matcher matcher = escapePattern.matcher(trustValue);  if (matcher.find()) {  return matcher.replaceAll("&#39"); // &#39 치환  }    return trustValue;  }  public static String **unescape**(String value) {  Pattern unescapePattern = Pattern.compile("&#39;");    if (value == null) {  return null;  }    String originalValue = StringEscapeUtils.unescapeHtml4(value); // <, >, " 치환    Matcher matcher = unescapePattern.matcher(originalValue);  if (matcher.find()) {  return matcher.replaceAll("'"); // 싱글쿼터로 치환  }  return originalValue;  }  } | |
| **실행 결과** |  |

제약사항이 없으므로 구현이 간단하다. 대상 문자열에 존재하는 스크립트 관련 특수문자 모두를 org.apache.commons.lang3.StringEscapeUtils 라이브러리를 사용하여 치환 및 복구한다.

#### 문제점

새롭게 추가되는 인자 각각에 반복적으로 필터 메소드를 적용해야 하는 번거로움이 있으며, 예외적으로 태그의 사용이 필요한 게시판과 같은 어플리케이션 존재 시 소스코드가 복잡해진다.