Special Report

웹 취약점과 해킹 매커니즘#7

XSS(Cross-Site Scripting) - ① 개념

■ 개요

XSS(Cross-Site Scripting, 크로스사이트 스크립팅)은 공격자가 입력한 악성스크립트가 사용자 측에서 응답하는 취약점으로, 사용자 입력값에 대한 검증이 미흡하거나 출력 시 필터링 되지 않을 경우 발생한다. 쿠키 값 또는 세션 등 사용자의 정보를 탈취하거나 피싱 사이트로의 접근 유도 등 사용자에게 직접적인 피해를 줄 수 있다.

이번 Special Report 에서는 사용자 입력값 검증이 미흡하여 악성스크립트를 삽입할 수 있는 공격인 XSS 에 대해 알아보고, 세 가지 종류의 XSS 공격 동작 과정을 다루며 보안대책을 살펴보고자 한다.

■ 환경 구성

취약점 테스트는 실습을 위해 구축한 웹 서버에서 진행된다. XSS 취약점이 존재하는 해당 웹 사이트는 특정 단어를 검색하면 데이터베이스에서 결과를 불러와 화면에 출력해 주는 검색 기능을 가지고 있다. 또한 글쓰기를 통해 게시물 작성이 가능하며 작성한 게시물은 누구나 열람 가능하다.

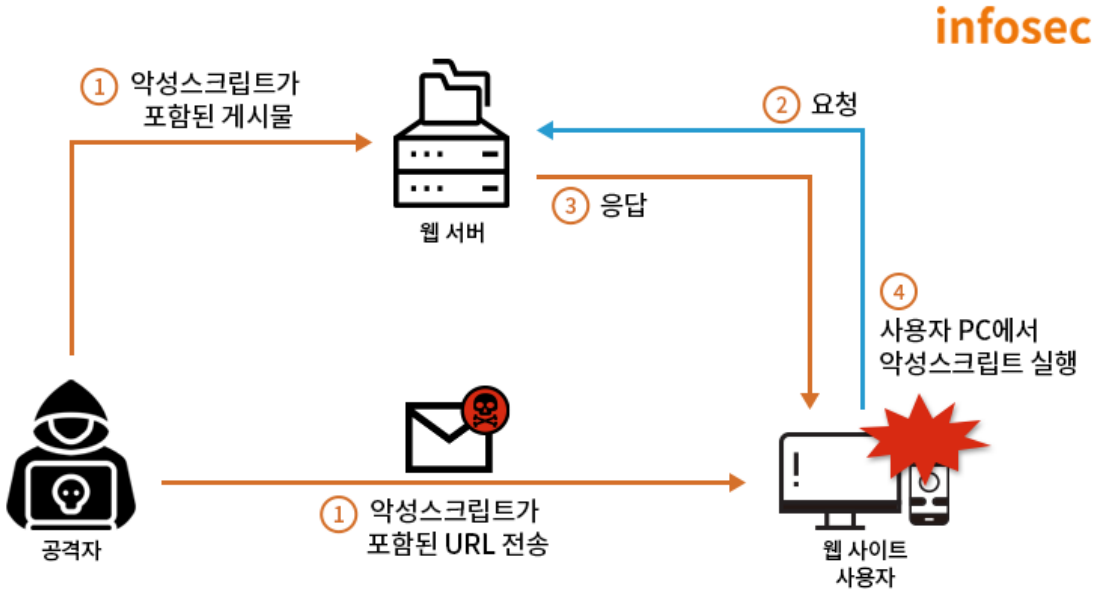
[XSS 취약점이 있는 게시판]

※ 실제 운영 중인 서버에 테스트 또는 공격을 하는 행위는 법적인 책임이 따르므로 개인용 테스트 서버 구축 또는 bWAPP, DVWA, WebGoat 등과 같은 웹 취약점 테스트 환경 구축을 통해 테스트하는 것을 권장한다.

※ 본 Special Report는 JSP와 Oracle Database 11gR2의 환경에서 실습을 진행한다.

■ XSS(Cross-Site Scripting)

XSS 취약점은 서버 측의 입력값 검증이 미흡할 경우 발생할 수 있는 취약점으로, 공격자가 게시물 또는 URL 을 통해 삽입한 악성스크립트가 사용자의 요청에 의해 사용자 측에서 응답하게 되어 발생한다.



[XSS 동작 과정]

사용자의 입력값을 받는 모든 곳에서 발생할 수 있으며, 웹 서버 사용자에게 직접적인 영향을 미칠 수 있는 공격 기법이다. 또한 게시물 또는 URL 에 포함된 악성스크립트가 동작하여 발생하는 취약점이기 때문에 불특정 다수를 대상으로 공격을 시도할 수 있다.

발생할 수 있는 피해는 스크립트 구문에 따라 사용자 쿠키 값 탈취를 통한 권한 도용과 세션 토큰 탈취, keylogger 스크립트를 삽입하여 키보드 입력값 탈취 등이 가능하다. 또한 피싱 사이트와 같은 악성 사이트로의 접근 유도가 가능하며 사용자에게 직접적인 피해를 줄 수 있다.

■ 공격 유형에 따른 분류

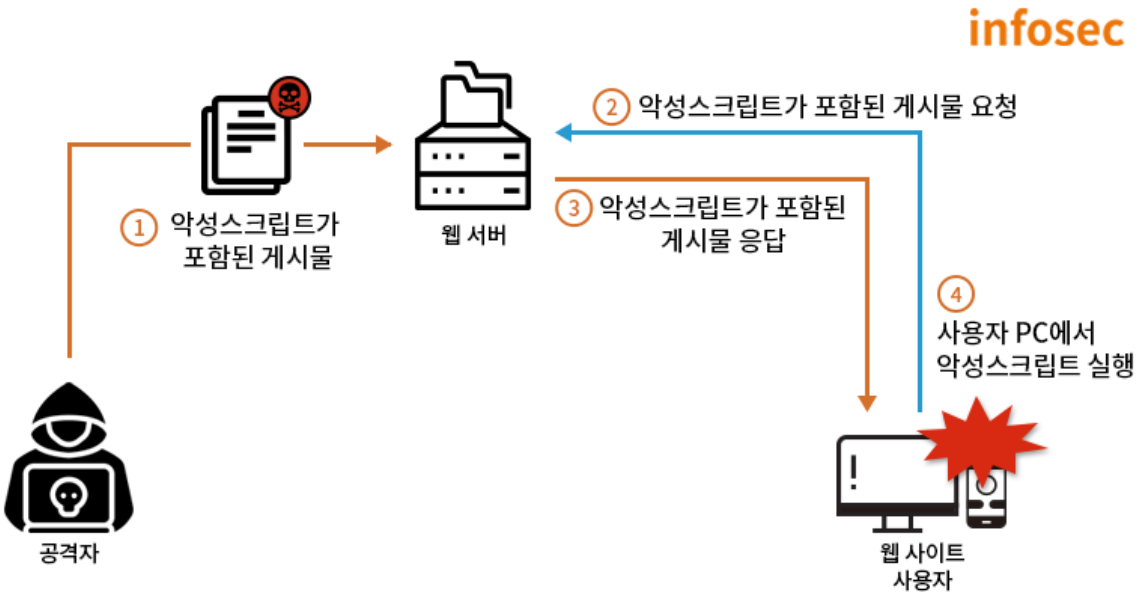
공격자가 삽입한 악성스크립트가 사용자 측에서 어떻게 동작하는지에 따라 크게 세 가지로 분류할 수 있으며 각각의 개념과 동작 과정은 다음과 같다.

Stored XSS (저장형 크로스사이트 스크립팅)

공격자의 악성스크립트가 데이터베이스에 저장되고 이 값을 출력하는 페이지에서 피해가 발생하는 취약점이다.

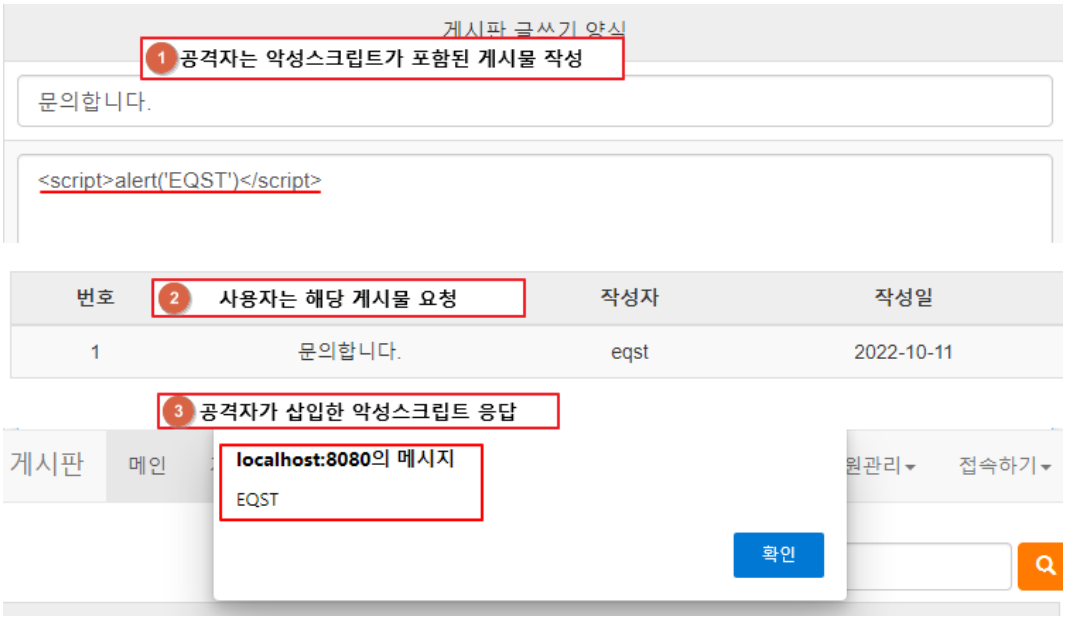
공격자는 악성스크립트가 포함된 게시물을 작성하여 게시판 등 사용자가 접근할 수 있는 페이지에 업로드한다. 이때 사용자가 악성스크립트가 포함된 게시물을 요청하면, 공격자가 삽입한 악성스크립트가 사용자 측에서 동작하게 된다.

공격자의 악성스크립트가 서버에 저장되어 불특정 다수를 대상으로 공격에 이용될 수 있어 Reflected XSS 보다 공격 대상의 범위가 훨씬 크다.



[Stored XSS]

Stored XSS 공격 과정은 다음을 통해 확인할 수 있다.

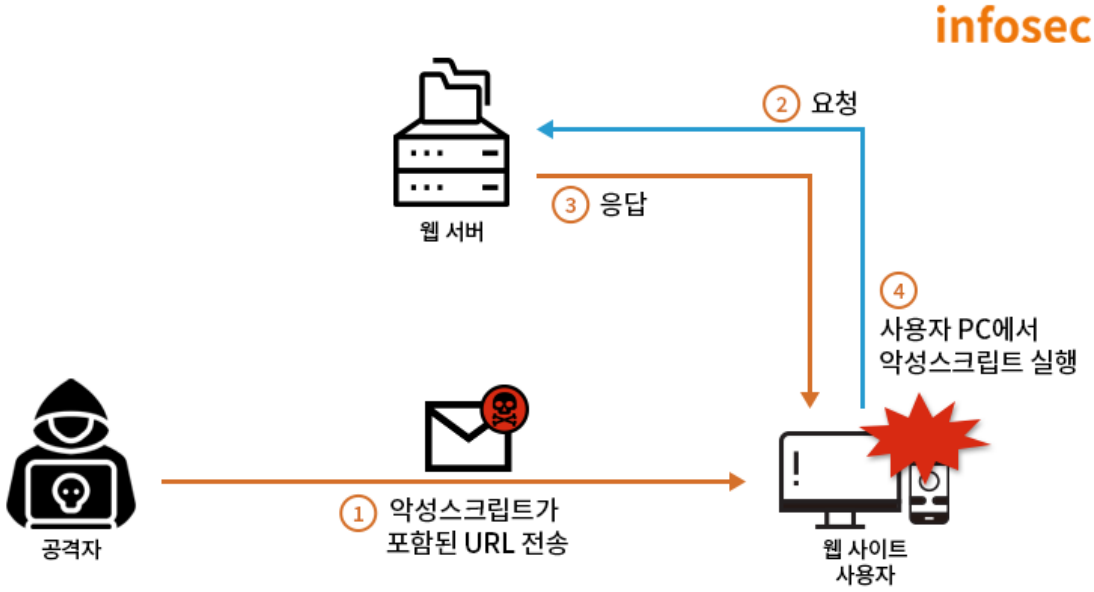


[Stored XSS 공격 과정]

Reflected XSS (반사형 크로스사이트 스크립팅)

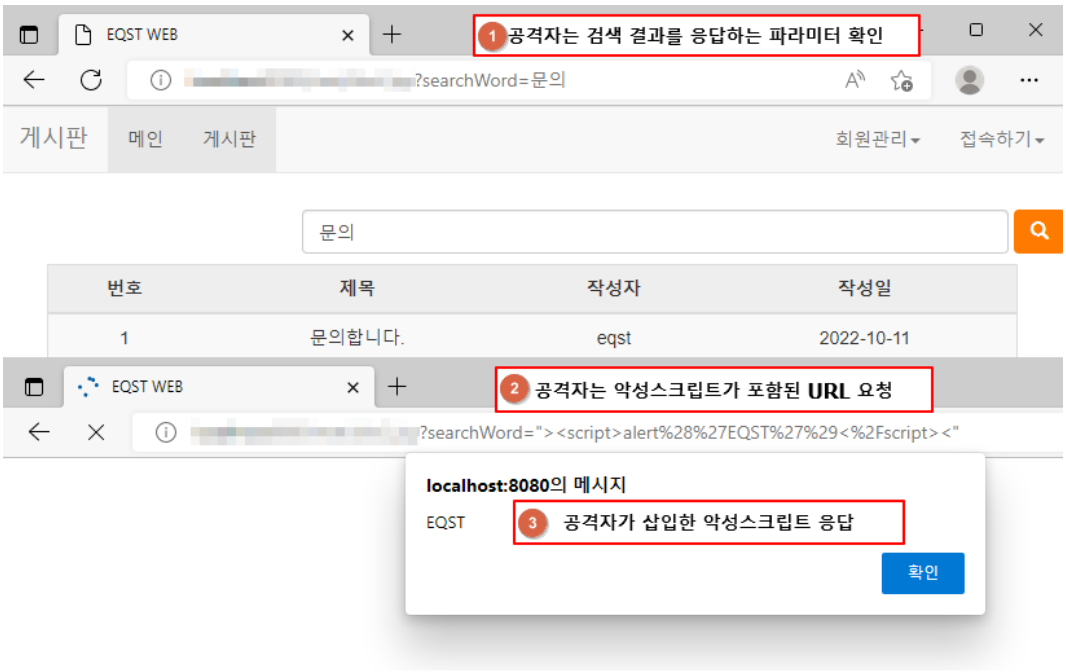
사용자가 요청한 악성스크립트가 사용자 측에서 반사(Reflected)되어 동작하는 취약점으로, 공격자의 악성스크립트가 데이터베이스와 같은 저장소에 별도로 저장되지 않고 사용자의 화면에 즉시 출력되면서 피해가 발생한다.

공격자는 악성스크립트가 포함된 URL 을 이메일, 메신저 등을 통해 사용자가 클릭할 수 있도록 유도한다. 사용자가 악성스크립트가 삽입된 URL 을 클릭하거나 공격자에 의해 악의적으로 조작된 게시물을 클릭했을 때 사용자의 브라우저에서 악성스크립트가 실행된다.



[Reflected XSS]

Reflected XSS 공격 과정은 다음을 통해 확인할 수 있다.

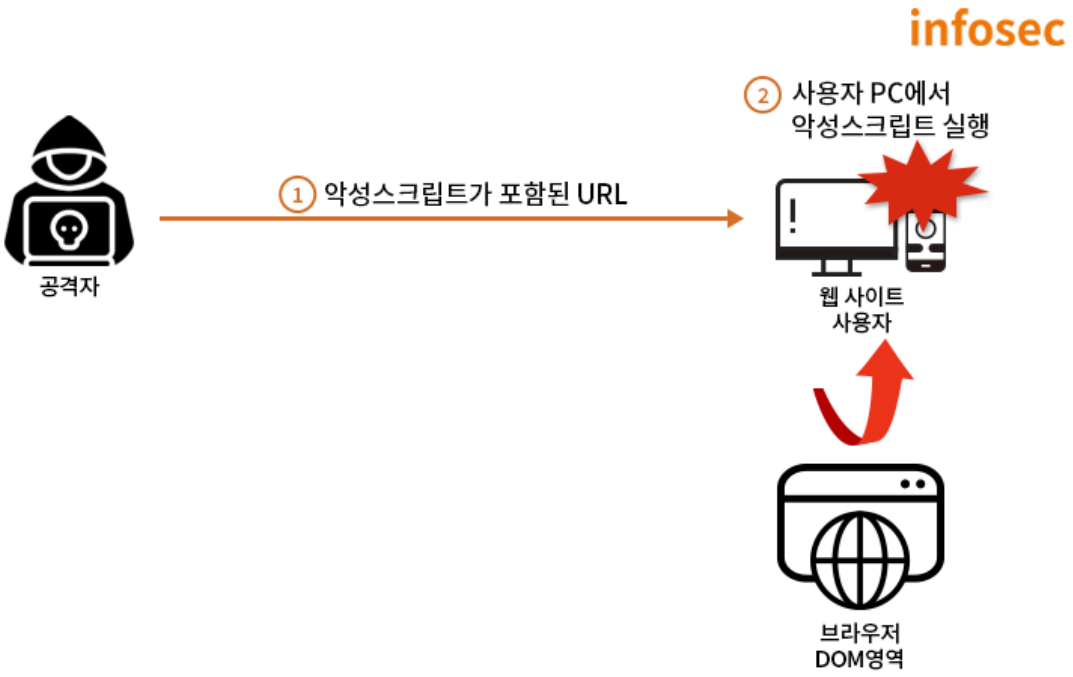


[Reflected XSS 공격 과정]

DOM Based XSS (DOM 기반 크로스사이트 스크립팅)

공격자의 악성스크립트가 DOM 영역에서 실행됨으로써 서버와의 상호작용 없이 브라우저 자체에서 악성스크립트가 실행되는 취약점이다. DOM 영역에 변화가 생기면 브라우저는 서버로 패킷을 보내지 않고 DOM 영역에서 페이지를 변환시킨다. 따라서 DOM 의 일부로 실행되기 때문에 브라우저 자체에서 악성스크립트가 실행된다.

> • DOM(Document Object Model, 문서 객체 모델) 이란?
> 브라우저가 웹 페이지를 렌더링 하는데 사용하는 모델로 HTML 및 XML 문서에 접근하기 위한 인터페이스이다. 브라우저는 HTML 문서를 읽고 해석한 결과를 DOM 형태로 재구성하여 사용자에게 제공한다.



[DOM Based XSS]

DOM Based XSS 공격 과정은 다음을 통해 확인할 수 있다.

[DOM Based XSS 공격 과정]

Stored XSS 과 Reflected XSS 는 서버에서 악성스크립트가 실행되고 공격이 이뤄지는 반면에 DOM Based XSS 는 서버와 상호작용 없이 브라우저에서 악성스크립트가 실행되고 공격이 이뤄진다.

■ 보안대책

XSS 취약점은 공격자가 삽입한 악성스크립트로 인해 발생하기 때문에 입력값 검증을 통해 악성스크립트가 삽입되는 것을 방지해야 하며, 악성스크립트가 입력되어도 동작하지 않도록 출력값을 무효화해야 한다.

입력값 필터링의 경우 데이터베이스에 악성스크립트가 저장되는 것을 원천적으로 차단해야 한다. 또한 악성스크립트가 삽입되어도 동작하지 않도록 출력값을 검증하여 스크립트에 사용되는 특수문자를 HTML Entity 로 치환하여 응답하도록 한다.

아래와 같이 <, >, ', " 등 스크립트에 쓰이는 문자가 입력될 경우 스크립트로 동작하여 XSS 공격이 가능할 수 있으므로 의미가 없는 일반 문자로 치환해야 한다.

문자	<	>	'	"	(	)
Entity	<	>	"	'	(	)

치환 함수인 replaceAll()를 사용하여 외부 입력값으로 받은 스크립트에 쓰이는 문자열을 필터링하여 공격자가 입력한 악성스크립트가 동작하지 않도록 한다.

```
...
String searchWord = request.getParameter("searchWord");

if (searchWord != null) {
	searchWord = searchWord.replaceAll("<","&lt;");
	searchWord = searchWord.replaceAll(">","&gt;");
	searchWord = searchWord.replaceAll("(","&#40;");
	searchWord = searchWord.replaceAll(")","&#41;");
}
...
```

위와 같이 문자열 치환을 적용하면 <script>는 <script> 로 치환되어 일반 문자처리되어 스크립트로 실행되지 않는다.

게시판과 같이 HTML 태그 사용이 필요한 경우에는 WhiteList Filter 를 통해 허용할 태그를 선정하여 해당 태그만 허용하는 방식을 적용해야 한다.

```
searchWord = searchWord.replaceAll("&lt;p&gt;","<p>");
searchWord = searchWord.replaceAll("&lt;br&gt;","<br>");
searchWord = searchWord.replaceAll("&lt;P&gt;","<P>");
searchWord = searchWord.replaceAll("&lt;BR&gt;","<BR>");
```

허용할 태그 목록 선정이 어려울 경우에는 javascript, document, alert 등과 같이 공격에 사용될 가능성이 있는 모든 문자열을 차단하는 BlackList Filter 방법을 사용할 수 있다. 하지만 모든 태그를 차단하는 것은 현실적으로 어려움이 있고 위험이 존재한다. 부득이하게 사용해야 하는 경우에는 태그마다 적용 가능한 이벤트 핸들러가 다르다는 것을 고려하여 적용하면 된다.

추가적으로 알려진 XSS 필터 관련 외부 라이브러리를 활용하는 방법도 있다.

- Lucy-XSS-Filter: WhiteList 방식으로 XSS 공격을 방어하는 Java 기반의 오픈 소스 라이브러리
  https://github.com/naver/lucy-xss-filter
- OWASP ESAPI: OWASP에서 제공하는 무료 오픈 소스 라이브러리로 JAVA, PHP 등 다양한 언어 지원
  https://github.com/ESAPI/esapi-java-legacy

■ 맺음말

지금까지 사용자 입력값 검증 미흡으로 인해 발생하는 취약점인 XSS(Cross-Site Scripting)에 대한 기본 개념을 살펴보았다. 공격자가 삽입한 악성스크립트가 사용자 측에서 동작하여 계정 탈취 등의 피해가 발생할 수 있으므로 보안대책을 통해 방지해야 한다.

이어지는 11 월호에서는 실무에서의 진단 방법, 필터링 우회 방법 등 실무에서 적용되는 내용을 다룬 XSS(Cross-Site Scripting) - ② 심화 편이 진행된다.