Special Report

웹 취약점과 해킹 매커니즘#8

XSS(Cross-Site Scripting) - ② 심화

■ 개요

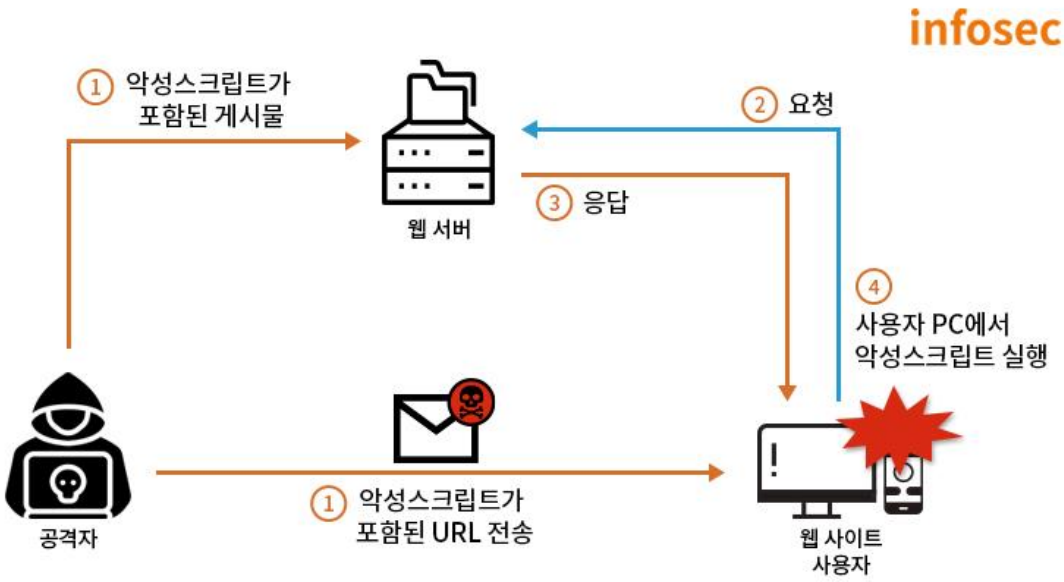
XSS(Cross-Site Scripting, 크로스사이트 스크립팅)은 공격자가 입력한 악성스크립트가 사용자 측에서 응답하는 취약점으로, 사용자 입력값에 대한 검증이 미흡하거나 출력 시 필터링 되지 않을 경우 발생한다. 쿠키 값 또는 세션 등 사용자의 정보를 탈취하거나 피싱 사이트로의 접근 유도 등 사용자에게 직접적인 피해를 줄 수 있으며, 다양한 공격 기법과 연계하여 큰 피해를 입힐 수 있다.

이번 Special Report 에서는 지난달 XSS - ①기본 편에 이어 XSS 의 다양한 공격 포인트와 XSS 를 이용한 연계 공격, 필터링 외의 대응 방안에 대해 다루고자 한다.

■ XSS 동작원리

상세 내용에 들어가기 앞서 XSS 의 동작원리를 살펴보자.

XSS 는 공격자가 웹 페이지에 악성스크립트를 삽입했을 때 사용자 측에서 동작하여 사용자에게 영향을 미치는 공격이다.



■ XSS 공격 포인트

대부분 alert 함수를 통해 XSS 취약점 존재 여부를 판단하는데, 이외에도 confirm, prompt 등 javascript 함수를 사용할 수 있다. 하지만 이처럼 자주 쓰이는 태그들에 대해선 필터링이 적용된 경우가 대부분이기 때문에 알려지지 않은 태그와 속성들을 사용하여 BlackList Filter 를 우회할 수 있다.

또한 이벤트, 태그, 브라우저 별로 적용할 수 있는 구문이 다르며, 웹 사이트 구성에 따라 사용할 수 있는 구문들이 다양하다. 따라서 다양한 공격 구문을 삽입해 브라우저의 스크립트 오류를 유도함으로써 XSS 발생 가능성을 확인해야 한다.

진단 시 참고할 수 있는 사이트는 다음과 같다.

- https://portswigger.net/web-security/cross-site-scripting/cheat-sheet
- https://github.com/RenwaX23/XSS-Payloads/blob/master/Payloads.txt
- https://cheatsheetseries.owasp.org/cheatsheets/XSS_Filter_Evasion_Cheat_Sheet.html

■ XSS 연계 공격

보통 XSS 취약점을 통해 alert 와 같은 함수를 사용하여 알림 창을 띄우는 행위 또는 사용자의 쿠키 및 세션을 탈취하여 권한을 획득하는 것이 일반적이다. 하지만 실제 공격은 악성스크립트 삽입을 통해 개발자가 의도하지 않은 행위를 하게 만드는 것이다. 또한, 웹 취약점을 이용하거나 공격 프레임워크와 연계가 가능하기에 이를 통해서 피해를 가중시킬 수 있다.

※ 이번 Special Report 를 통해 진행되는 실습은 가상 환경에 구축한 웹 서버 및 bWAPP 환경에서 진행한다. bWAPP 은 웹 취약점 실습이 가능한 오픈소스이다. 실제 운영 중인 서버에 테스트 또는 공격을 하는 것은 불법이며 법적인 책임이 따르므로 주의해야 한다.

웹 취약점과의 연계

Case 1) 파라미터 변조 + XSS

파라미터 변조 취약점은 공격자가 웹 브라우저에서 서버로 패킷이 전송되는 과정에서, 해당 패킷을 가로채 공격자의 의도대로 파라미터를 변조하는 공격이다. 공격자의 요청 값에 대한 검증이 미흡할 경우 발생하며 이 과정에서 XSS 공격을 연계한다면 피해자에게 영향을 미칠 수 있다.

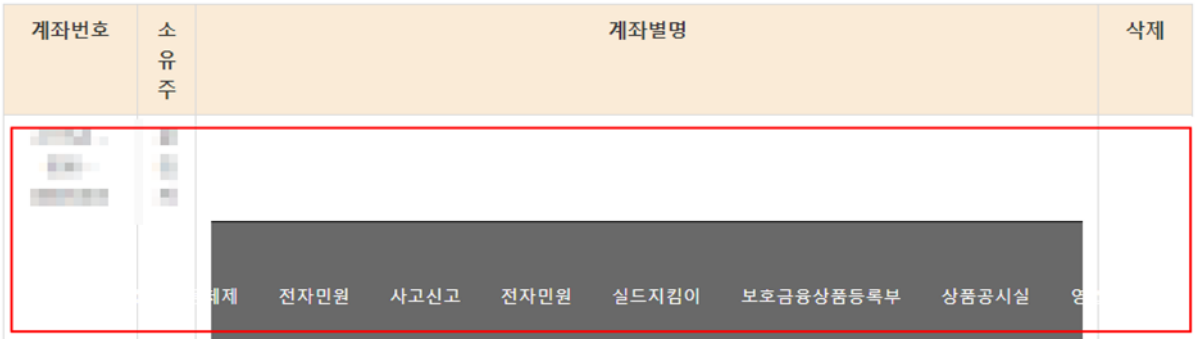
다음은 파라미터 변조를 통해 타 사용자의 계좌 정보에 접근한 후 악성스크립트를 삽입하여 사용자의 계좌 조회 페이지에 영향을 미치는 예시이다.

step 1) 공격자는 파라미터 변조를 통해 타 사용자의 계좌 정보 접근

step 2) 공격자는 계좌별명에 JavaScript 를 주석 처리하는 스크립트 삽입

step 3) Stored XSS 취약점으로 인해 악성스크립트가 서버에 저장되어 동작

계좌별명에 삽입된 악성스크립트 구문인 주석 처리가 동작하여 삭제 버튼이 나타나지 않으며, 해당 스크립트가 종료된 이후의 화면이 나타나는 것을 볼 수 있다.



Case 2) 파일 업로드 + XSS

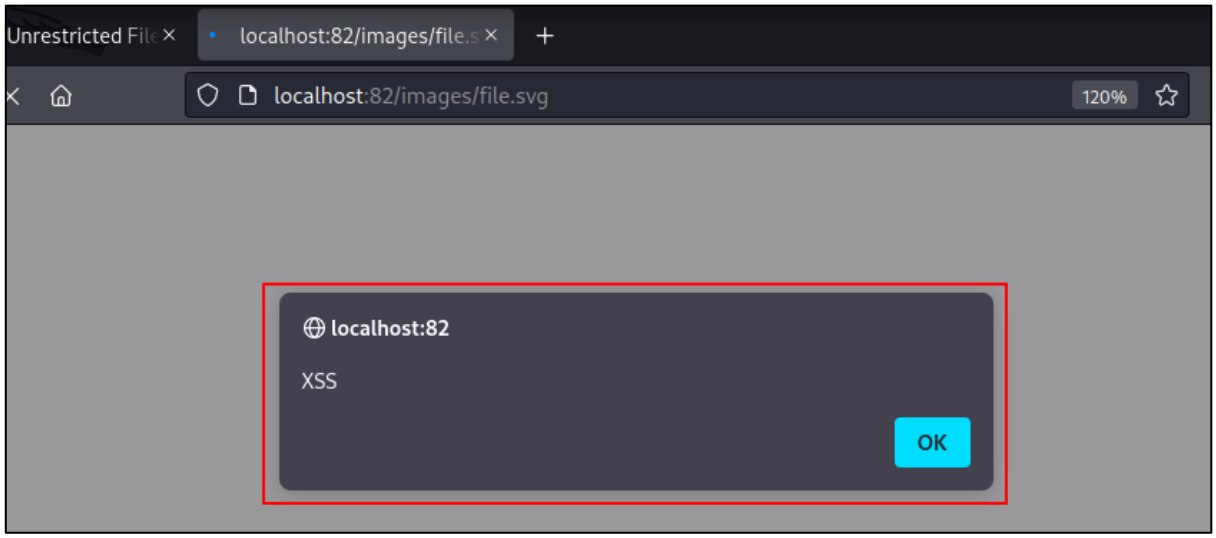
파일 업로드 취약점은 웹 사이트의 파일 업로드 기능을 이용하여 인가받지 않은 파일을 서버에 임의로 업로드하는 공격이다. 웹쉘 형태로 파일 업로드에 성공할 경우 서버의 자원을 장악할 수 있고 페이지를 변조하여 클라이언트에게 영향을 미칠 수 있다. 파일 확장자 검증이 미흡하고 업로드한 파일에 접근하여 실행할 수 있을 때 악성스크립트가 포함된 파일에 접근하여 실행하면 해당 스크립트가 동작한다.

다음은 파일 업로드 기능이 있는 페이지에 alert 창을 띄우는 악성스크립트가 작성된 파일을 업로드하여 실행한 예시이다.

step 1) 악성스크립트가 포함된 svg 파일을 서버에 업로드

```
1 <svg xmlns="http://www.w3.org/2000/svg">
2 <script>alert('XSS');</script>
3 <rect x="0" height="100" width="100" style="fill: #cccccc"/>
4 <line xl="20" yl="80" x2="80" y2="80" style="stroke: #ff0000; stroke-width: 5;"/>
5 </svg>
```

step 2) 업로드한 파일에 접근하여 실행 시 악성스크립트 동작 확인

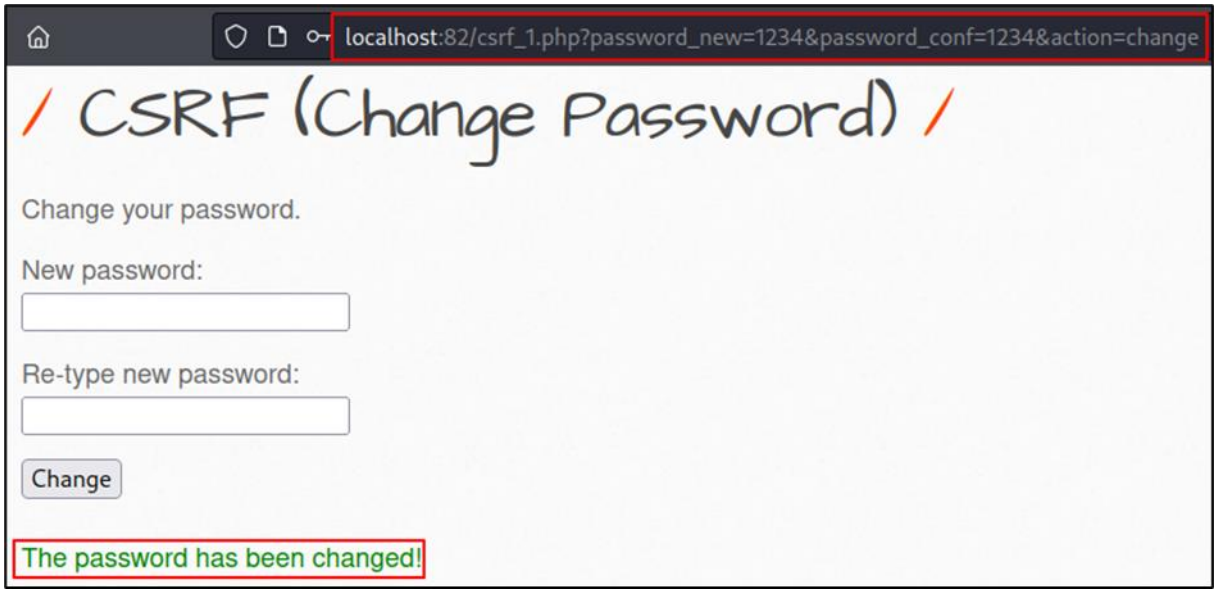


Case 3) CSRF(Cross Site Request Forgery)

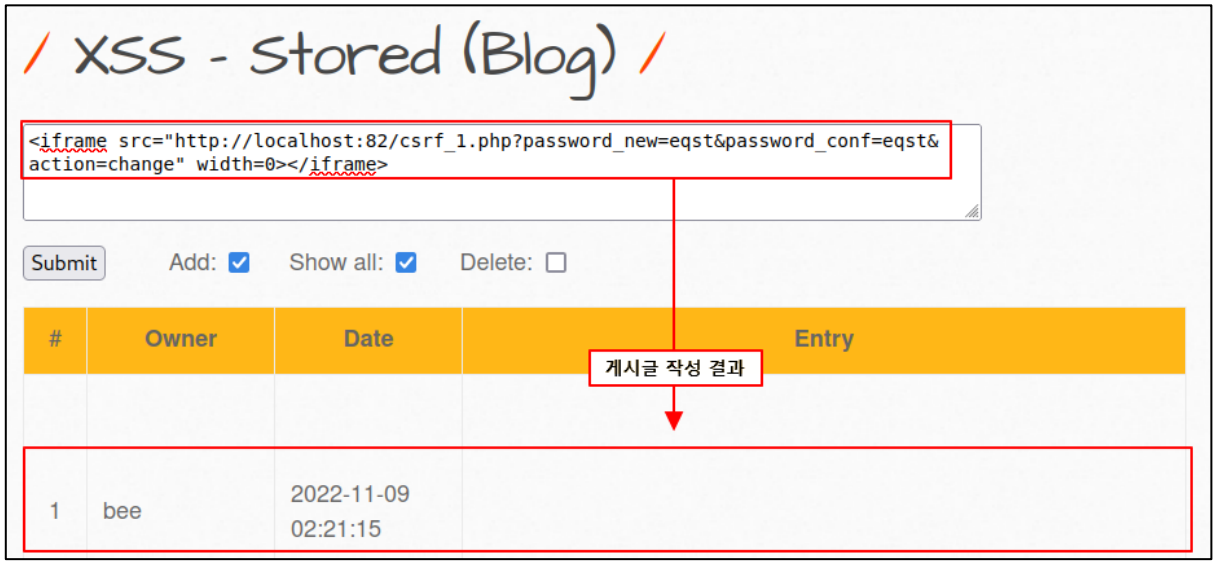
CSRF 는 사용자가 의도하지 않은 작업을 수행하도록 유도하는 공격으로, 사용자의 권한을 통해 공격자가 의도한 행위를 웹 사이트에 요청하도록 만드는 공격이다. XSS 와 연계한다면 공격 범위를 확장할 수 있다.

다음은 Stored XSS 취약점이 있는 게시판에 패스워드를 변경하는 악성스크립트를 삽입하여 일반 사용자가 해당 게시글을 클릭하면 공격자의 의도대로 패스워드가 변경되는 예시이다.

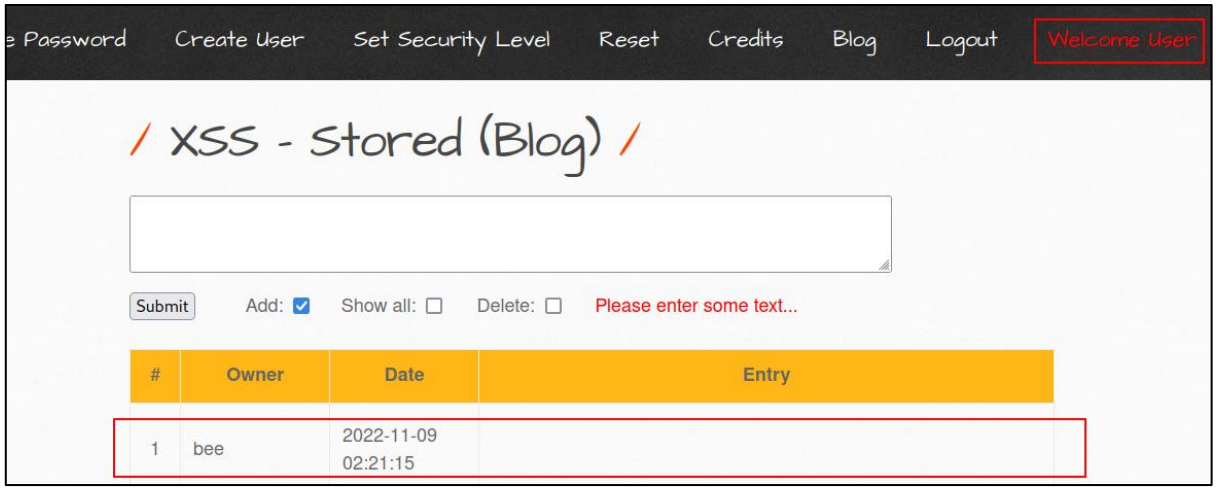
step 1) 공격자는 패스워드 변경 시 사용되는 파라미터 획득



step 2) 공격자는 Stored XSS 취약점이 존재하는 게시판에 공격자가 지정한 패스워드(eqst)로 변경을 유도하는 스크립트 구문 삽입



step 3) 공격자가 작성한 게시글을 일반 사용자(user)로 로그인한 사용자가 조회



step 4) 패스워드 변경 스크립트가 동작하여 일반 사용자는 기존의 패스워드로 로그인 불가

위의 예시와 같이 XSS 취약점을 이용하여 공격자가 원하는 동작을 하는 스크립트 구문을 삽입해 CSRF 공격과 연계할 수 있다. CSRF 개념과 원리 등 자세한 내용은 웹 취약점과 매커니즘 #9 에서 이어질 예정이다.

공격 프레임워크 – BeEF

BeEF(The Browser Exploitation Framework)는 웹 브라우저에 중점을 둔 침투 테스트 도구이며, XSS 와 연계하여 다양한 공격을 실습해 볼 수 있는 프레임워크이다. 칼리리눅스를 통해 간단히 설치할 수 있으며, 후킹을 담당하는 JavaScript 파일(hook.js)을 가지고 있다. 따라서 공격자는 BeEF 를 실행하고 있는 자신의 IP 주소, Port 번호와 hook.js 파일로 구성된 스크립트를 피해자 서버에 삽입하고, XSS 공격에 성공하면 BeEF 를 통해 후킹이 가능해져 다양한 공격을 할 수 있다.

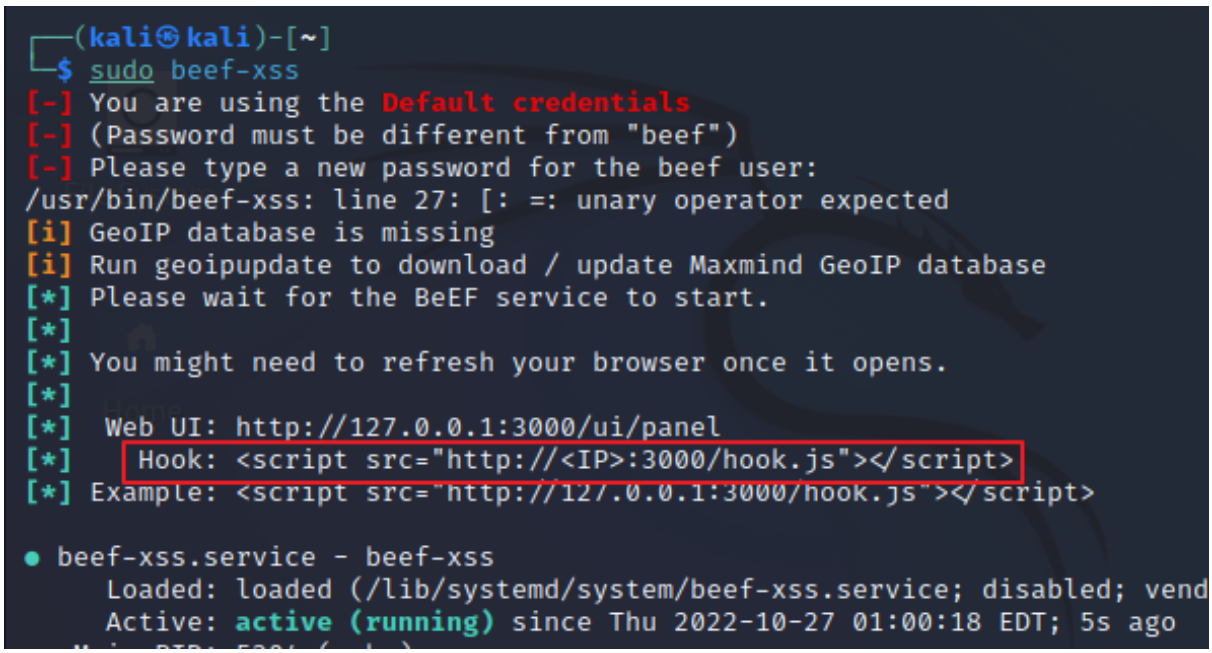
설치 방법은 아래의 URL 또는 칼리리눅스에서 명령어를 입력하여 설치할 수 있다.

- URL : https://beefproject.com/
- 명령어 : $ apt-get install beef-xss

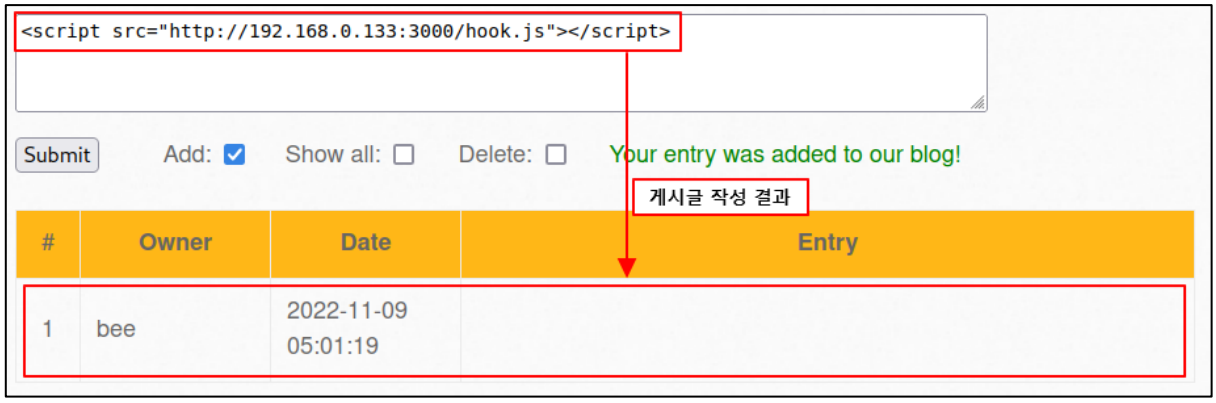
다음은 BeEF 를 통해 피해자 PC 에 구글 로그인 화면을 띄우는 공격의 실습 예시이다.

step 1) BeEF 실행

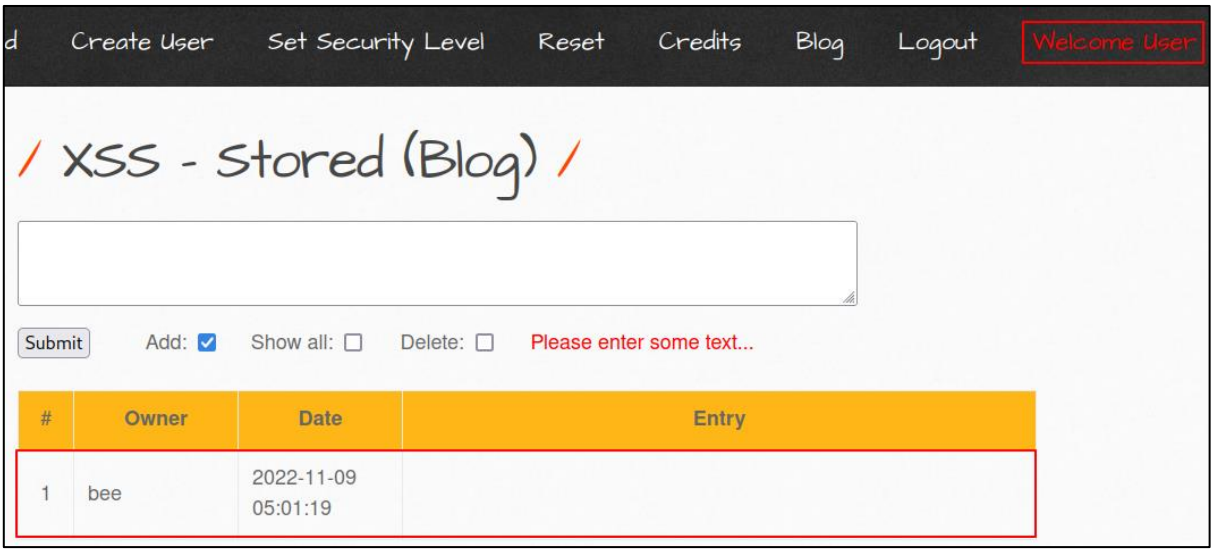
후킹을 할 수 있는 스크립트 구문을 확인할 수 있다.



step 2) 공격자는 후킹을 위한 스크립트를 게시판에 작성

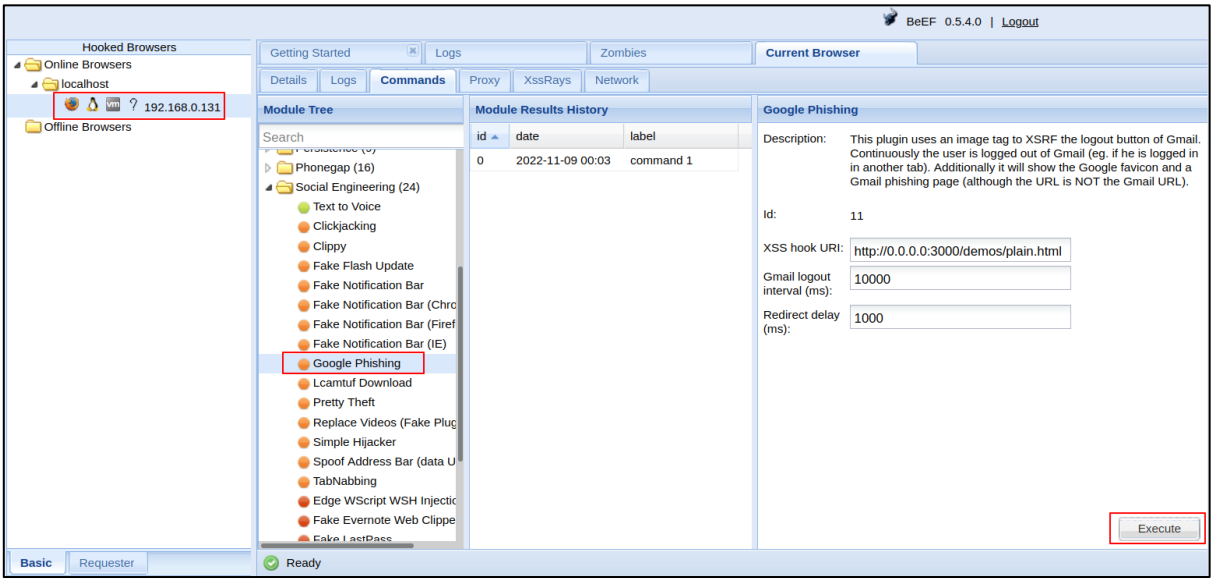


step 3) 일반 사용자(user)는 공격자가 작성한 게시글에 접근

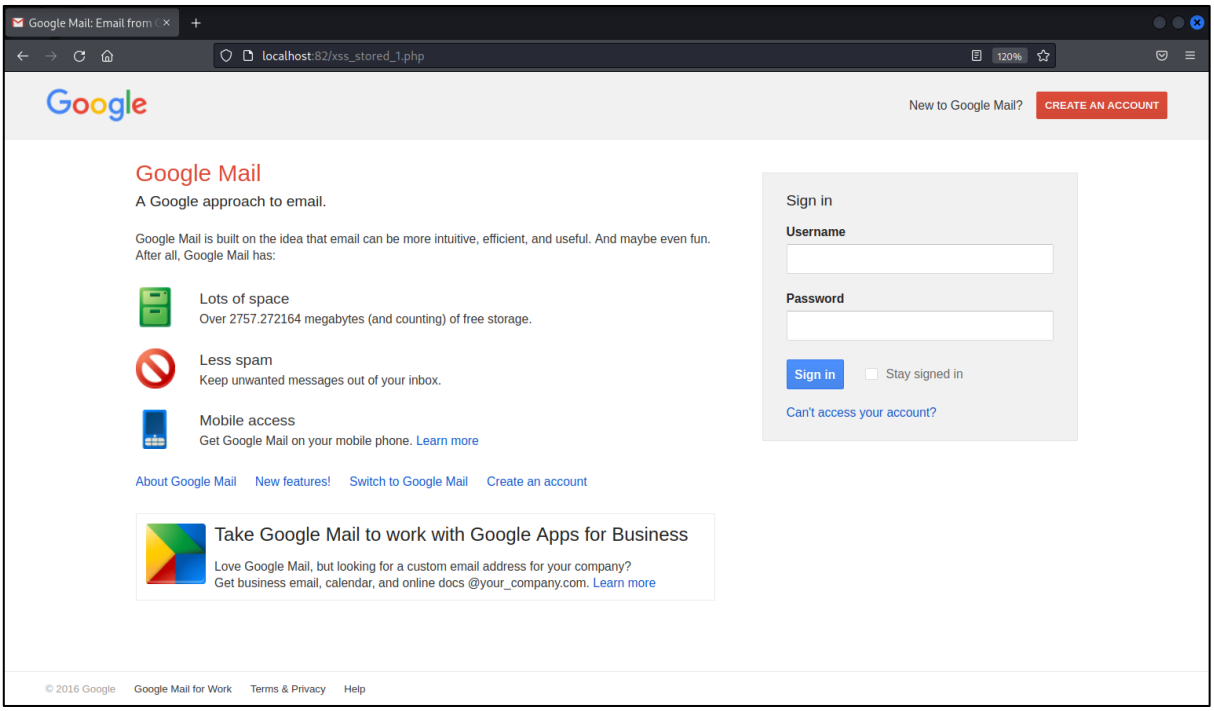


step 4) 일반 사용자 PC 가 공격자 PC 의 BeEF 에 연결되어 다양한 공격 가능

예시에서는 피해자 PC 에 구글 로그인 화면을 띄우는 피싱을 실행하였다.



step 5) 일반 사용자의 PC 화면이 구글 메일 로그인 화면으로 변경



■ jQuery 환경에서의 XSS

jQuery 란?

jQuery는 JavaScript를 쉽게 활용할 수 있도록 도와주는 오픈소스 기반의 JavaScript 라이브러리로, 하나의 JavaScript 파일(.js)로 존재하며 해당 파일을 선언하여 사용한다. 다양한 브라우저 간 호환성을 지원하며 JavaScript 코드를 간결하게 표현할 수 있다. DOM(Document Object Model) 조작, 이벤트 처리 등의 작업을 간편하게 할 수 있다는 장점이 있다.

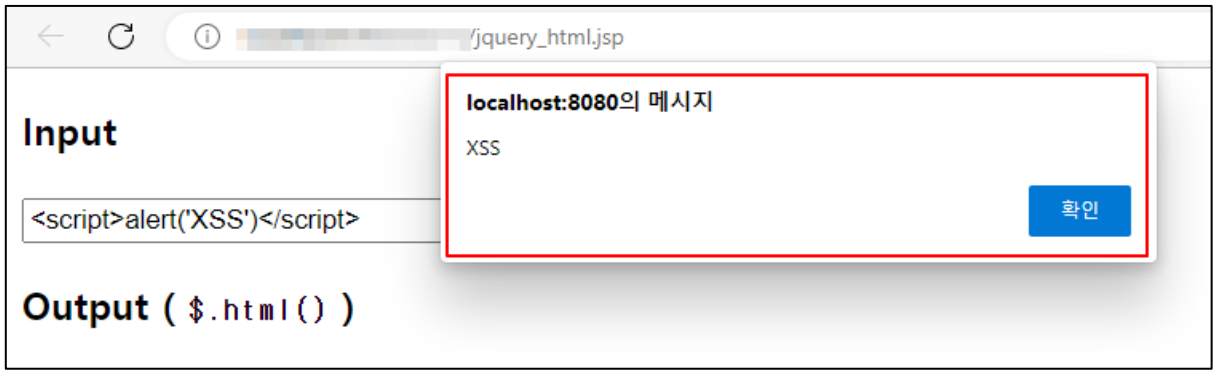
* jQuery를 사용하는 방법은 여러 가지가 있지만, CDN(Contents Delivery Network)를 통해 별도의 설치 없이 HTML의 〈head〉 태그에 추가하여 사용할 수 있다. (https://releases.jquery.com/)

jQuery 사용 시 메소드 중 .html(), .append() 등을 사용할 경우 XSS에 주의해야 한다. HTML 태그 입력 시 이를 태그 자체로 인식하여 악성스크립트가 삽입될 경우 동작할 수 있기 때문이다.

jQuery 환경에서의 XSS에 대한 내용은 아래의 링크를 통해 자세히 확인할 수 있다.

- https://portswigger.net/web-security/cross-site-scripting/dom-based

다음 예시를 통해 .html() 메소드 사용 시 HTML 태그가 인식되어 alert 창이 뜨는 것을 확인할 수 있다.




```
<script>
    $('form').on('submit', function(e) {
      e.preventDefault();
      var ov = $('#test').val();
    $('#output-html').html(ov);
    });
</script>
```

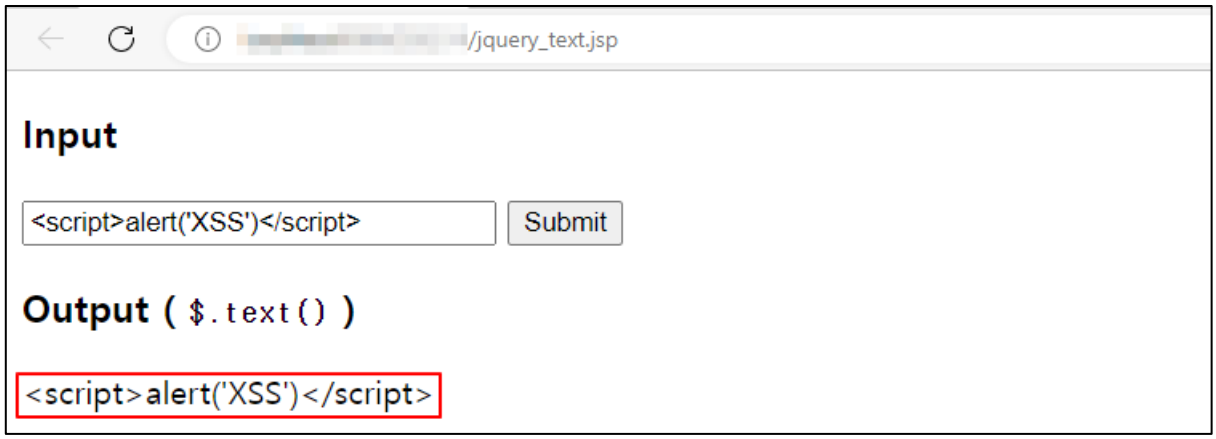
jQuery 의 안전한 메소드 사용

JavaScript 라이브러리인 jQuery 를 사용하여 환경을 구성한 경우, XSS 에 안전한 메소드를 사용하여 공격을 예방해야 한다. .html() 메소드를 사용하여 HTML 태그가 직접적으로 삽입되는 방식 대신 사용자 HTML 입력값을 이스케이프 처리하여 텍스트로 삽입되도록 아래의 메소드를 사용해야 한다. 이는 공격자가 악성스크립트를 삽입해도 일반 문자열로 인식해 사용자 측에서 동작하지 않도록 할 수 있다.

XSS에 안전한 jQuery 메소드			
.text()	.attr()	.prop()	.val()

※ .attr()의 경우 href 속성을 이용하면 XSS 로부터 안전하지 않으니 주의해야 한다.

.text() 메소드 사용 시 HTML 태그가 문자 그대로 인식되어 출력되는 것을 확인할 수 있다.




```
<script>
    $('form').on('submit', function(e) {
      e.preventDefault();
      var ov = $('#test').val();
    $('#output-text').text(ov);
    });
</script>
```

jQuery Ajax 환경에서의 XSS

jQuery 는 $.ajax 함수를 이용하여 자체적으로 Ajax(Asynchronous JavaScript And XML)를 지원한다. Ajax 은 JavaScript 와 XML 을 이용하여 클라이언트와 서버 간의 데이터를 비동기 통신으로 주고받는 언어이다. 서버로부터 데이터를 가져올 때 전체 페이지를 새로고침하지 않고 필요한 데이터만 불러올 수 있다. 웹 서버와 비동기식 통신을 하기 위한 객체인 브라우저의 XMLHttpRequest 객체를 이용해 전체 페이지를 새로고침 없이 페이지의 일부만 로드하는 방법을 사용하며, 최근에는 Fetch API 를 사용한다.

비동기 방식으로 웹 페이지를 불러오는 Ajax 는 XSS 공격에 취약하다. 비동기 방식으로 웹 페이지를 불러오는 과정에서 악성스크립트가 삽입된다면 사용자 모르게 공격 코드가 사용자 측에서 동작한다. 따라서 이를 이용하여 XSS 공격을 하면 사용자가 게시글을 읽는 것만으로도 새로운 게시글 작성, 회원 탈퇴 등의 악의적인 행위를 할 수 있다.

■ XSS 대응방안

사용자 입력값과 출력값을 필터링하거나 치환하는 방법을 통해 XSS 를 예방할 수 있지만, HTML 태그를 사용하는 게시판이 존재하거나 외부 자바스크립트 API 사용 등이 불가피한 경우가 있어 XSS 공격에 영향을 미치는 태그나 문자열을 모두 필터링하는 것은 한계가 있다.

따라서 XSS – ①기본 편에서 이야기한 보안대책 외에도 악성스크립트가 사용자 측에서 동작하지 않도록 관련 라이브러리, 프레임워크 등을 활용하는 방안도 고려해야 한다.

예를 들어 Spring 프레임워크 사용 시 Spring Security 를 사용하거나, JSP 환경에서는 JSTL 라이브러리를 통해 XSS 공격에 대응할 수 있다. 이번 Special Report 에서는 JSTL 을 통해 출력값을 처리하는 예시를 다룬다. 적용 가능한 알려진 보안 라이브러리는 다음과 같다.

JSTL 에서의 출력값 처리

JSTL 은 자바 표준 태그 라이브러리(JavaServer Pages Standard Tag Library)로 JSP 에서 자주 사용되는 태그를 커스텀 하여 라이브러리로 정의한 코드 집합이다. 사용법이 간단하고 코드를 간결하게 표현할 수 있어 가독성이 좋고 협업, 유지 보수 등에 유리하다.

JSTL 에는 Core, XML, Database 등 다양한 태그가 존재하지만, 이번 Special Report 에서는 JSP 환경에서 XSS 공격에 대응할 수 있는 Core 라이브러리의 c:out 태그에 대한 내용을 다룬다. 또한 escapeXml 함수를 통해 특수문자를 문자 그대로 출력하여 사용자 입력값에 의해 악성스크립트가 삽입되어도 동작하지 않도록 출력값을 처리할 수 있다.

설치 및 선언 방법은 다음과 같으며, c:out 태그 사용 시 유의할 점은 적용할 파라미터마다 태그를 명시해야 한다.

step 1) JSTL 라이브러리 설치 후 프로젝트의 /WEB-INF/lib 경로에 추가한다.

설치 URL	https://mvnrepository.com/artifact/javax.servlet/jstl

step 2) 태그를 적용할 jsp 페이지 상단에 Core 라이브러리와 함수 사용을 선언한다.

Core 라이브러리 선언	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
함수 사용 선언	<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

step 3) 적용할 파라미터에 c:out 태그를 명시한다.

c:out 형식	<c:out value="[출력값]" escapeXml="[true|flase]"/>

※ escapeXml 속성이 true일 경우가 기본 값이며, 특수문자가 문자 그대로 출력된다.

알려진 보안 라이브러리 사용

이미 만들어진 외부 라이브러리인 OWASP ESAPI, NAVER Lucy XSS Filter 등을 사용하는 것이 유용하다. 하지만 항상 최신 버전 사용을 사용해야 하며 누락된 문자열 등이 있을 수 있으니 사용에 주의해야 한다. 또한 Lucy-XSS-Filter 의 경우, JSON 에 대한 요청은 처리해 주지 않기 때문에 이러한 점에 유의하여 라이브러리를 적용해야 한다.

- OWASP ESAPI : https://github.com/ESAPI/esapi-java-legacy

- Lucy-XSS-Filter : https://github.com/naver/lucy-xss-servlet-filter

주의할 점은, 이를 사용한다고 해서 반드시 안전한 것은 아니므로 항상 최신 버전을 유지하고 라이브러리의 경우 누락된 문자열이 있는지에 대해 점검하며 지속적으로 업데이트해야 한다.

■ 맺음말

지금까지 사용자 입력값에 대한 검증과 출력값 처리가 미흡할 경우, 공격자가 입력한 악성스크립트가 사용자 측에서 동작하는 취약점인 XSS(Cross-Site Scripting)에 대해 알아보았다.

공격자가 삽입한 악성스크립트가 사용자 측에서 동작하여 계정 탈취 등의 피해가 발생할 수 있고, 파라미터 변조 등과 같은 웹 취약점 또는 공격 프레임워크를 이용한 다양한 연계 공격이 가능하다. 또한 웹 사이트의 구성과 표현에 따라 사용할 수 있는 구문이 다양하고 특수문자 필터링 여부 등에 따라 새로운 구문과 패턴을 통해 우회할 가능성이 존재한다.

따라서 클라이언트 측에서 공격자의 악성스크립트가 삽입되어도 동작하지 않도록 서버 측에서의 보안 설정이 반드시 필요하며 상시 점검을 통해 발생 가능성을 지속적으로 확인해야 한다.