**보안 취약점 문제 해결**

**XSS 공격**

소프트웨어는 다른 사용자에게 제공되는 웹 페이지로 사용되는 출력에 배치되기 전에 사용자가 제어 할 수있는 입력을 무력 화하거나 잘못 무력화하지 않습니다. XSS (교차 사이트 스크립팅) 취약성은 다음과 같은 경우에 발생합니다.

1. 신뢰할 수없는 데이터는 일반적으로 웹 요청에서 웹 응용 프로그램에 입력됩니다.

2. 웹 응용 프로그램은이 신뢰할 수없는 데이터를 포함하는 웹 페이지를 동적으로 생성합니다.

3. 페이지 생성 중에 응용 프로그램은 JavaScript, HTML 태그, HTML 속성, 마우스 이벤트, Flash, ActiveX 등과 같이 웹 브라 우저에서 실행할 수있는 내용을 데이터에 포함하지 못하도록합니다.

4. 피해자는 신뢰할 수없는 데이터를 이용하여 삽입 된 악성 스크립트가 포함 된 웹 브라우저를 통해 생성 된 웹 페이지를 방문합니다.

5. 스크립트는 웹 서버에서 보낸 웹 페이지에서 온 것이므로 피해자의 웹 브라우저는 웹 서버의 도메인 컨텍스트에서 악성 스크립트를 실행합니다.

6. 이것은 한 도메인의 스크립트가 다른 도메인의 리소스에 액세스하거나 코드를 실행할 수 없어야한다는 웹 브라우저의 동일 출처 정책의 의도를 효과적으로 위반합니다. XSS에는 세 가지 주요 종류가 있습니다.

 유형 1 : 반영된 XSS (또는 비 영구)-서버가 HTTP 요청에서 직접 데이터를 읽고이를 HTTP 응답에 다시 반영합니다. 반사 된 XSS 익스플로잇은 공격자가 피해자가 취약한 웹 애플리케이션에 위험한 콘텐츠를 제공하도록 유도 한 다음 피해자에게 다시 반영되어 웹 브라우저에 의해 실행될 때 발생합니다. 악성 콘텐츠를 전달하는 가장 일반적인 메커니즘은 공개적으로 게 시되거나 피해자에게 직접 이메일로 전송되는 URL에 매개 변수로 포함하는 것입니다. 이러한 방식으로 구성된 URL은 많은 피싱 체계의 핵심을 구성하며, 공격자는 피해자가 취약한 사이트를 참조하는 URL을 방문하도록 유도합니다. 사이트가 공격 자의 콘텐츠를 피해자에게 다시 반영한 후 해당 콘텐츠는 피해자의 브라우저에 의해 실행됩니다.

 유형 2 : 저장된 XSS (또는 영구)-애플리케이션이 데이터베이스, 메시지 포럼, 방문자 로그 또는 기타 신뢰할 수있는 데이 터 저장소에 위험한 데이터를 저장합니다. 나중에 위험한 데이터가 애플리케이션으로 다시 읽어 들여 동적 콘텐츠에 포함됩 니다. 공격자의 관점에서 악성 콘텐츠를 삽입하기위한 최적의 위치는 많은 사용자 또는 특히 흥미로운 사용자에게 표시되는 영역입니다. 관심있는 사용자는 일반적으로 응용 프로그램에서 높은 권한을 갖거나 공격자에게 중요한 중요한 데이터와 상 호 작용합니다. 이러한 사용자 중 한 명이 악성 콘텐츠를 실행하는 경우 공격자는 사용자를 대신하여 권한있는 작업을 수행하 거나 사용자 소유의 민감한 데이터에 액세스 할 수 있습니다. 예를 들어 공격자는 XSS를 로그 메시지에 삽입 할 수 있으며, 관 리자가 로그를 볼 때 제대로 처리되지 않을 수 있습니다.

 유형 0 : DOM 기반 XSS-DOM 기반 XSS에서 클라이언트는 페이지에 XSS 삽입을 수행합니다. 다른 유형에서는 서버가 주 입을 수행합니다. DOM 기반 XSS에는 일반적으로 사용자가 양식을 제출하기 전에 양식에 대한 온 전성 검사를 수행하는 Javascript와 같이 클라이언트로 전송되는 서버 제어의 신뢰할 수있는 스크립트가 포함됩니다. 서버 제공 스크립트가 사용자 제공 데이터를 처리 한 다음이를 웹 페이지에 다시 삽입하면 (예 : 동적 HTML 사용) DOM 기반 XSS가 가능합니다. 악성 스크립트가 삽입되면 공격자는 다양한 악성 활동을 수행 할 수 있습니다. 공격자는 세션 정보를 포함 할 수있는 쿠키와 같은 개인 정보를 피해자의 컴퓨터에서 공격자에게 전송할 수 있습니다. 공격자는 피해자를 대신하여 웹 사이트에 악의적 인 요청을 보낼 수 있으며, 피해자가 해당 사이트를 관리 할 수있는 관리자 권한이있는 경우 사이트에 특히 위험 할 수 있습니다. 피싱 공격은 신뢰할 수있는 웹 사이트를 모방하고 피해자가 암호를 입력하도록 속여 공격자가 해당 웹 사이트에서 피해자의 계정을 손상시킬 수 있습니다. 마지막으로 스크립트는 "드라이브 바이 해킹"이라고도하는 피해자의 컴퓨터를 장악 할 수있는 웹 브라우저 자체의 취약점을 악용 할 수 있습니다. 대부분의 경우 공격은 피해자가 알지 못하는 사이에 시작될 수 있습니다. 신중한 사용자라도 공격자는 URL 인코딩 또는 유니 코드와 같은 공격의 악의적 인 부분을 인코딩하기 위해 다양한 방법을 자주 사용하므로 요청이 의심스럽지 않게 보입니다.

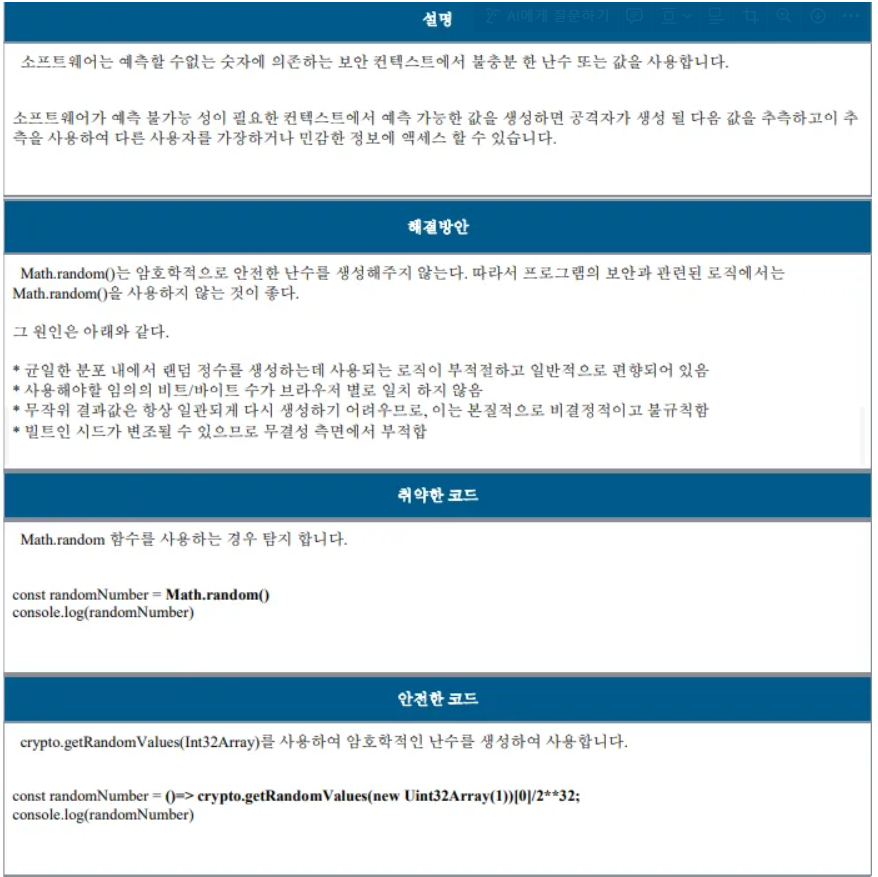
예시)

var element = document.createElement('test'); element.innerHTML = '<div id="test">test</div>';

해결방안)

var element = document.getElementById('test'); element.insertAdjacentHTML('afterend', '<div id="test">test</div>')

**암호(Math.random)**



**eval**



**자식 노드 한번에 지우기**

document.getElementById('prog\_num').replaceChildren();

**특문처리**

String originalString = "{}$%^&\*()";

// 특수문자를 HTML 엔티티로 변환 String encodedString = StringEscapeUtils.escapeHtml4(originalString);

// HTML엔티티를 원래 특수문자로 변환

StringEscapeUtils.unescapeHtml4(encodedString);