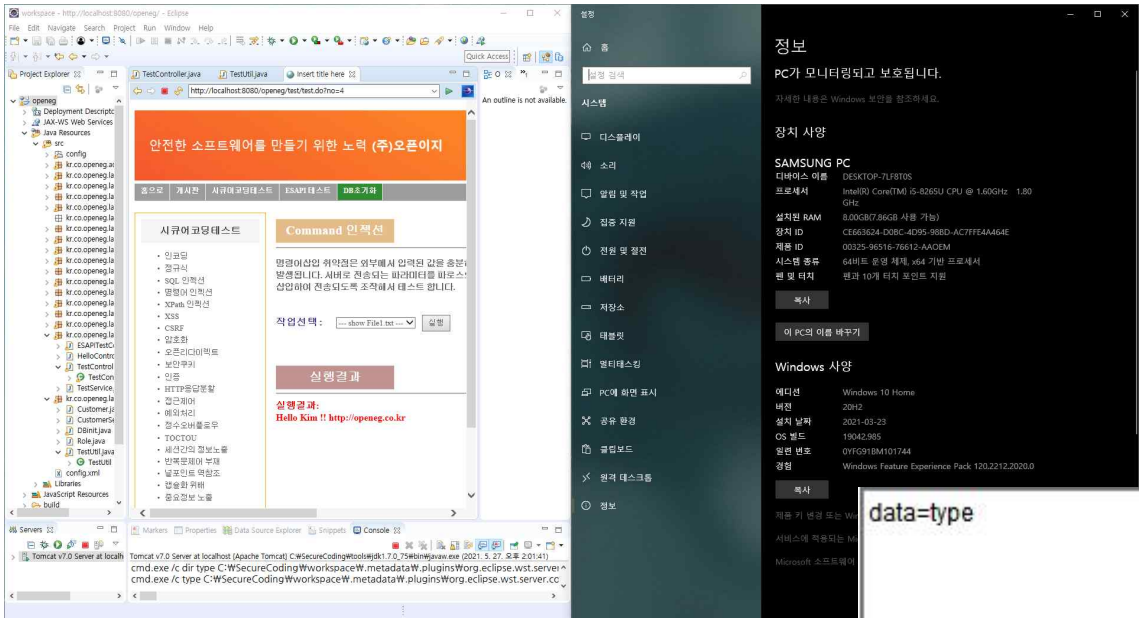
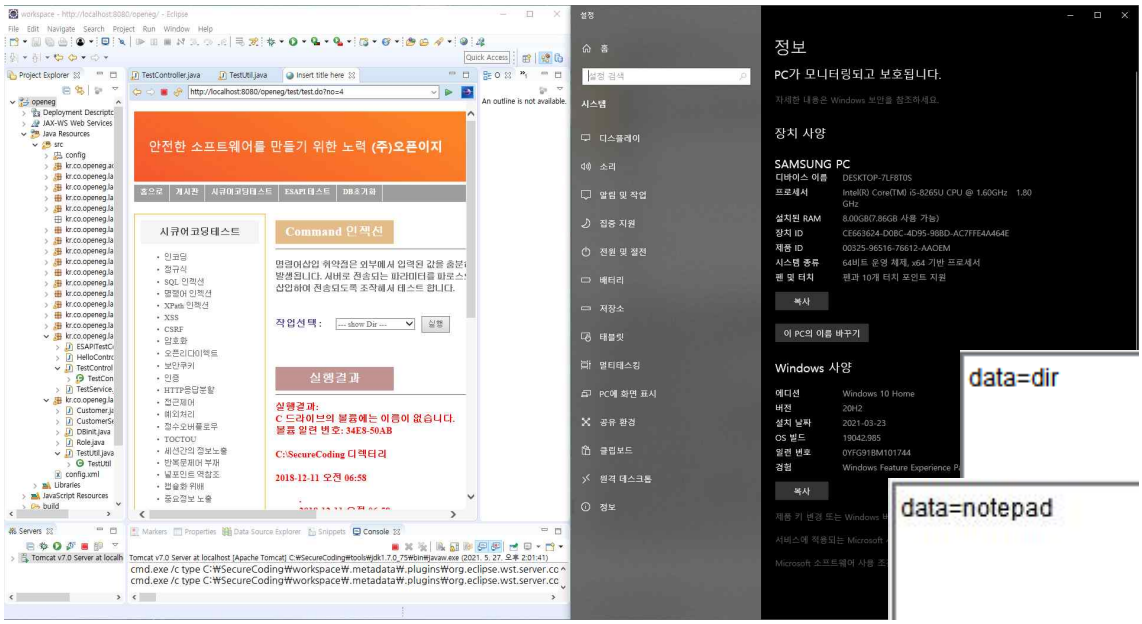
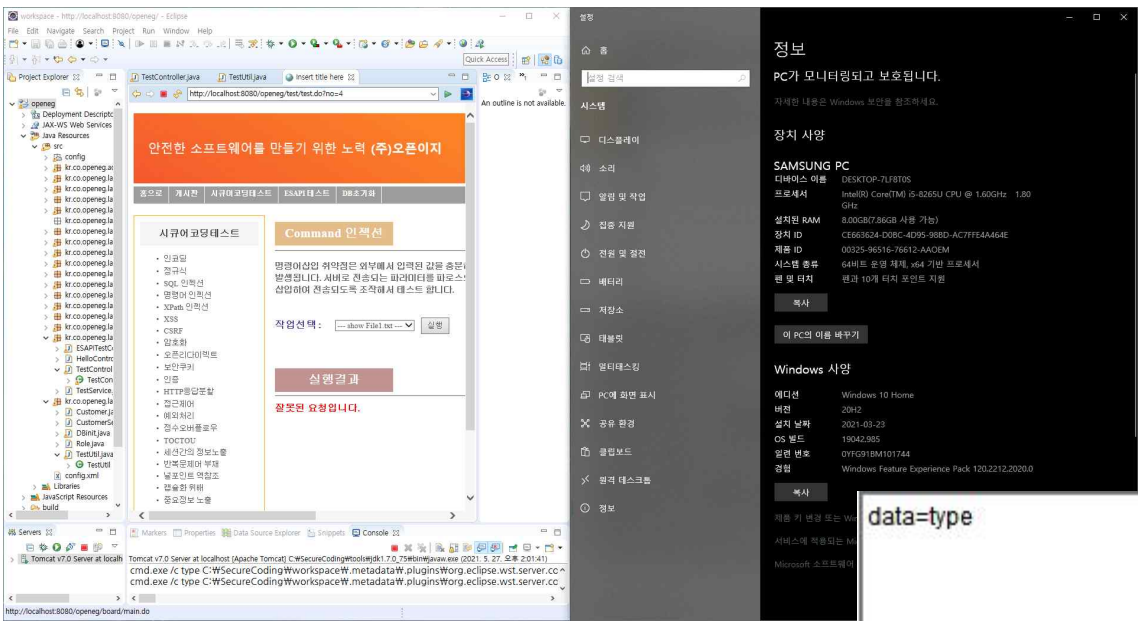
1. 운영체제 명령어 삽입

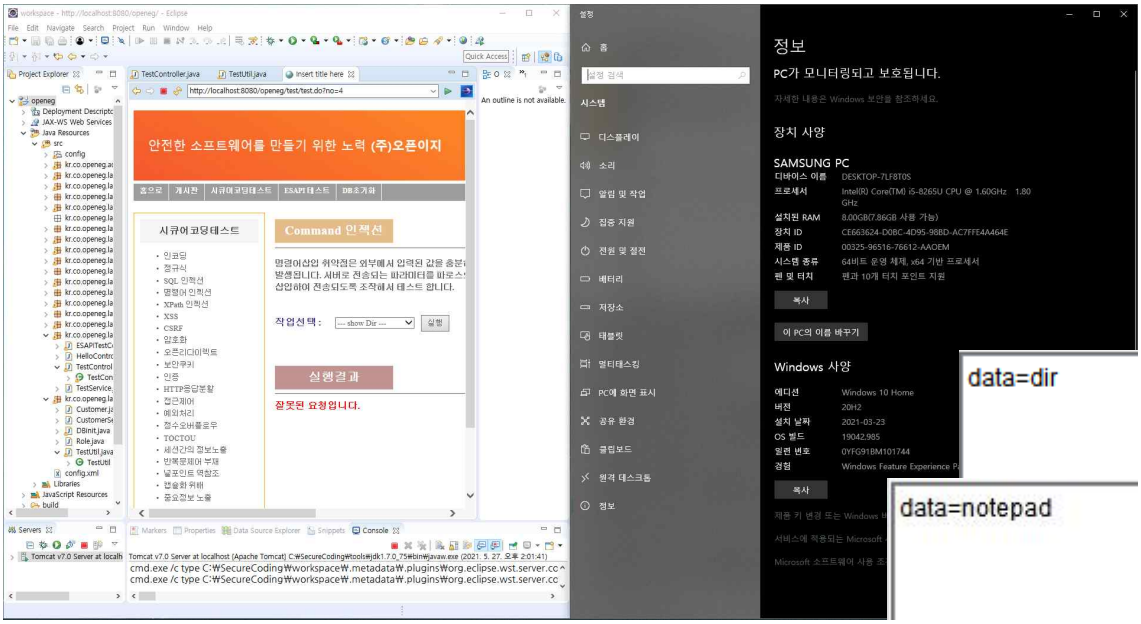

안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지
시큐어코딩테스트
Command 인젝션
실행결과
실행결과:
Hello Kim !! http://openeg.co.kr
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양
data=type

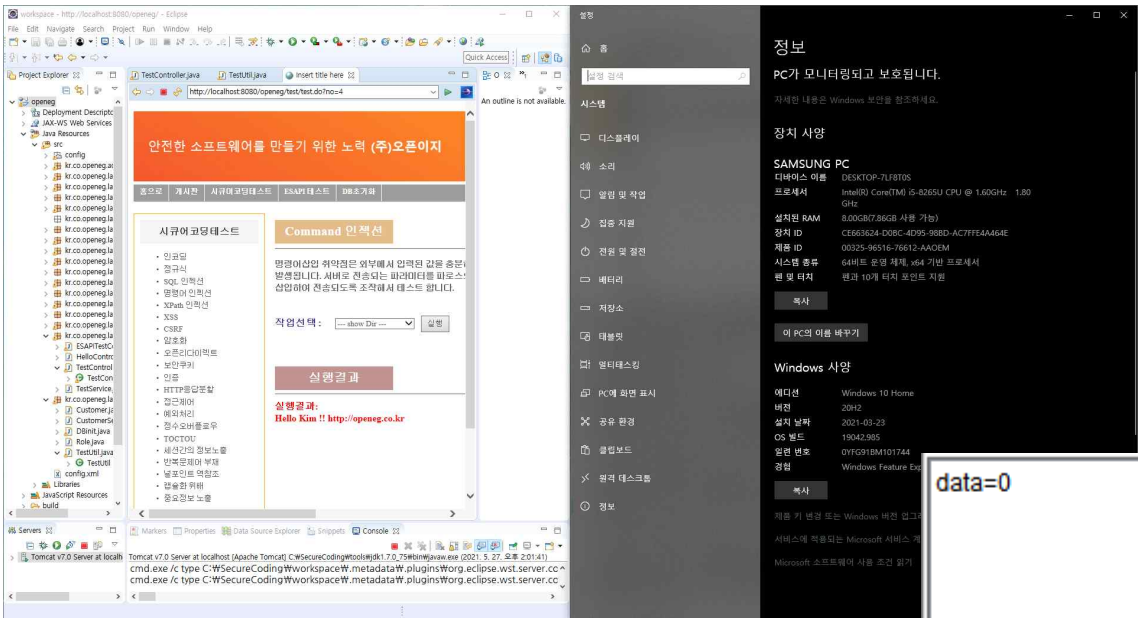

안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지
시큐어코딩테스트
Command 인젝션
실행결과
실행결과:
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 34E8-50AB
C:\SecureCoding 디렉터리
2018-12-11 오전 06:58
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양
data=dir
data=notepad

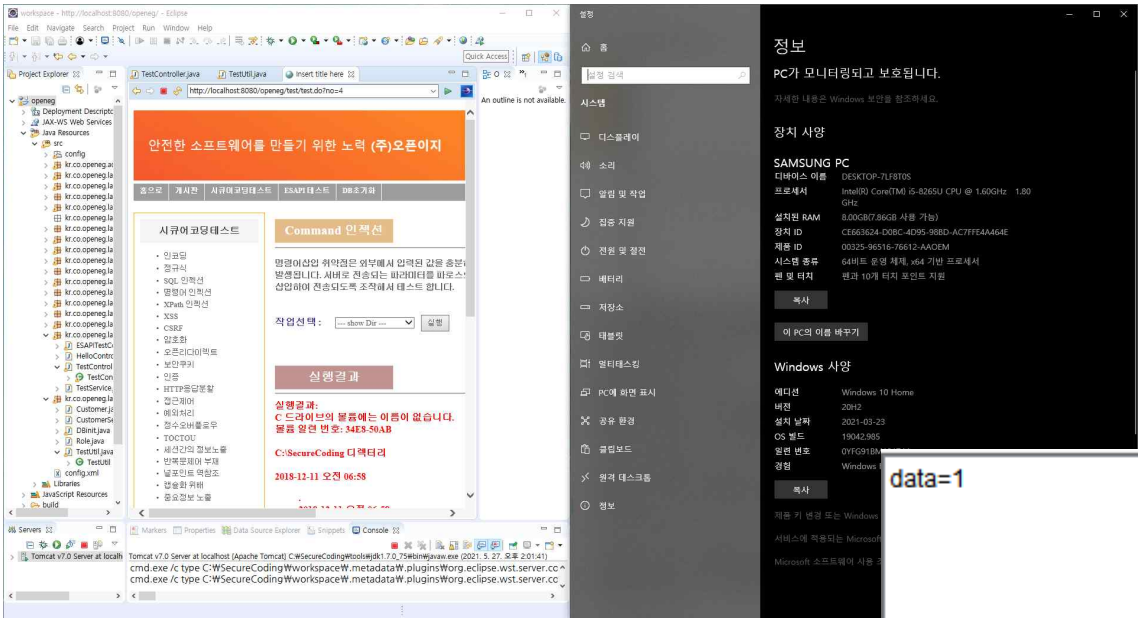
(코드수정)

```
String[] allowCommand={"type", "dir"};

int index=TestUtil.getInt(data);

if(index < 0 || index > 1) {
   buffer.append("잘못된 요청입니다.");
   return buffer.toString();
}
else {
   data=allowCommand[index];
}

if ( data != null  && data.equals("type")) {
      data=data+" "+
            request.getSession().getServletContext().getRealPath("/")+
            "file1.txt";
      System.out.println(data);
}
```



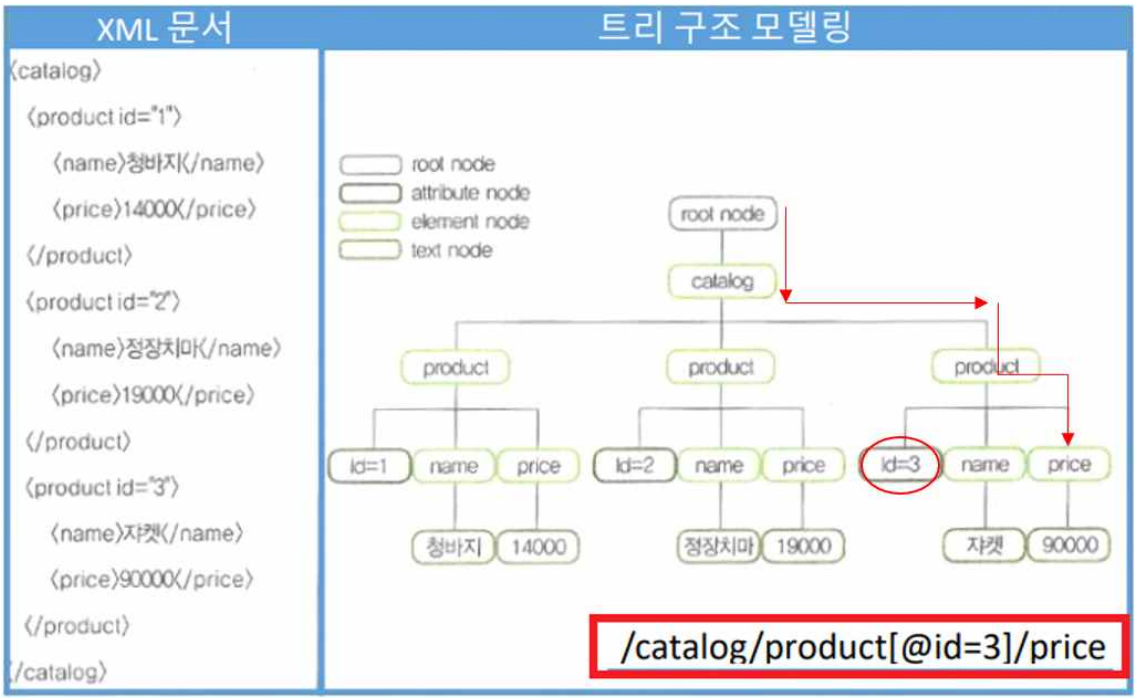

안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지
시큐어코딩테스트
Command 인젝션
실행결과
잘못된 요청입니다.
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양
data=dir
data=notepad


안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지
시큐어코딩테스트
Command 인젝션
실행결과
실행결과:
Hello Kim !! http://openeg.co.kr
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양
data=0


안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지
시큐어코딩테스트
Command 인젝션
실행결과
C:\SecureCoding 디렉터리
2018-12-11 오전 06:58
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양
data=1

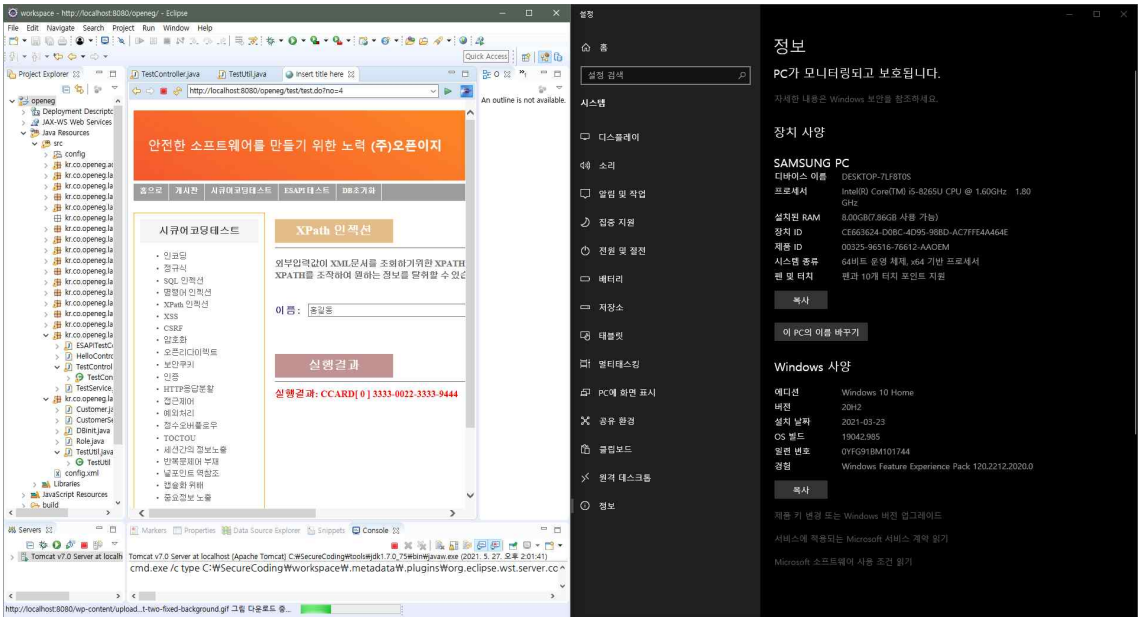
2. XPath 삽입

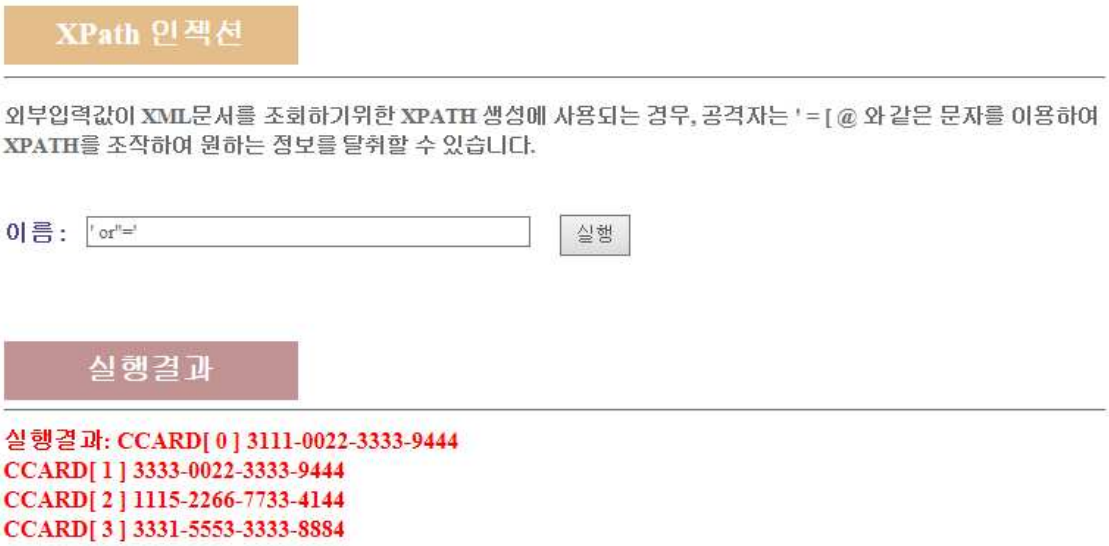
– XPath(XML Path Language): XML문서에 저장된 데이터를 애플리케이션에서 검색하거나 읽기 위해 사용하는 표현 방식



– XPath 삽입 취약점 발생 원인: 사용자 입력을 받아, 입력 값에 대한 검증 없이 동적으로 XPath 쿼리를 생성하면 공격자가 해당 쿼리문의 의미를 수정할 수 있음

(1) XPath 삽입 공격


안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지
시큐어코딩테스트
XPath 인젝션
실행결과
실행결과: CCARD[0] 3333-0022-3333-9444
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양

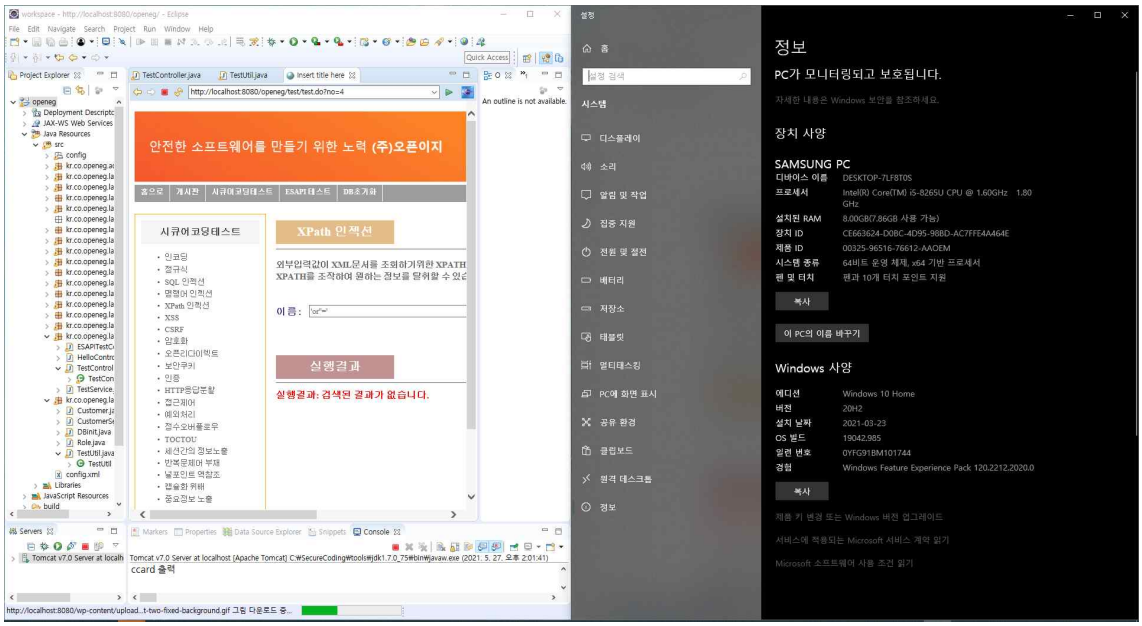

XPath 인젝션
외부입력값이 XML문서를 조회하기위한 XPATH 생성에 사용되는 경우, 공격자는 ' = [@ 와 같은 문자를 이용하여 XPATH를 조작하여 원하는 정보를 탈취할 수 있습니다.
이름 :
' or''='
실행
실행결과
실행결과: CCARD[0] 3111-0022-3333-9444
CCARD[1] 3333-0022-3333-9444
CCARD[2] 1115-2266-7733-4144
CCARD[3] 3331-5553-3333-8884

(2) XPath 삽입 방어

(코드 수정)

```
public String XPathFilter(String input)
{
  return input.replaceAll("['₩₩[]","");
}
```

```
//    String expression = "/addresses/address[@name='"+name+"']/ccard";
    String expression = "/addresses/address[@name='"+XPathFilter(name)+"']/ccard";
```



– XPathFilter(): 입력된 값에 대해서 replaceAll이라는 함수를 사용하여 쿼리문에서 사용되어지면 안되는 특수문자들을 공백으로 바꾸어 주는 필터링 작업을 수행한다.

3. 크로스 사이트 스크립팅(XSS) 공격 실습

– 크로스 사이트 스크립팅(XSS) 취약점: 외부 입력 값이 충분한 검증 없이 동적으로 생성되는 응답 페이지에 사용되는 경우

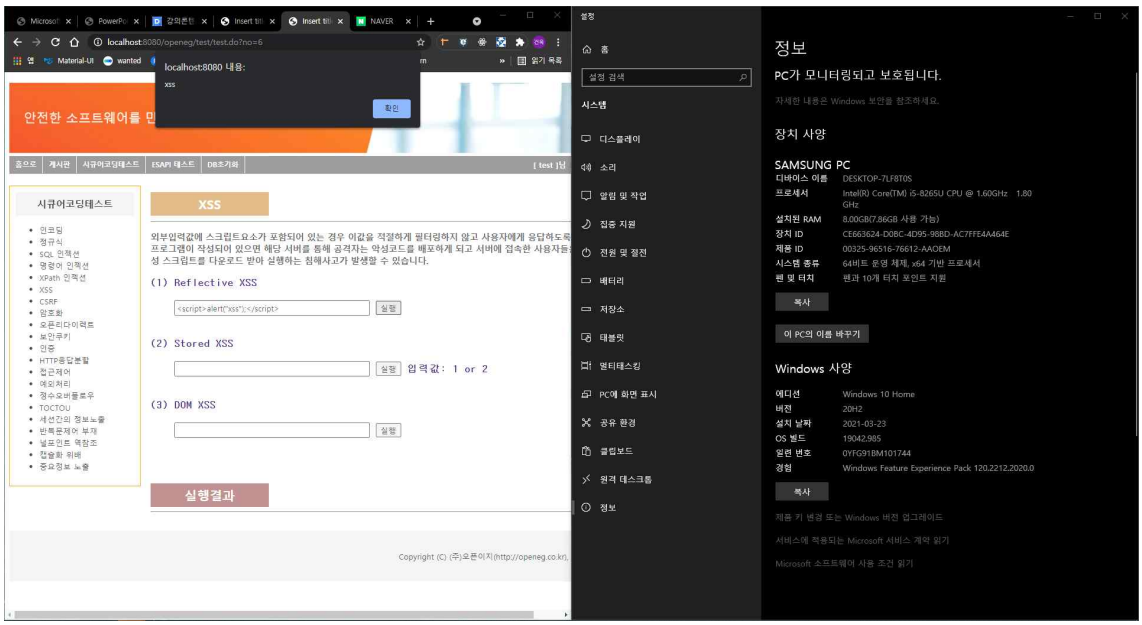
– Reflective XSS(반사공격): 공격자가 악성 스크립트가 포함된 URL을 클라이언트에 노출한 후 클릭을 유도하고 악성 행위를 수행한다.

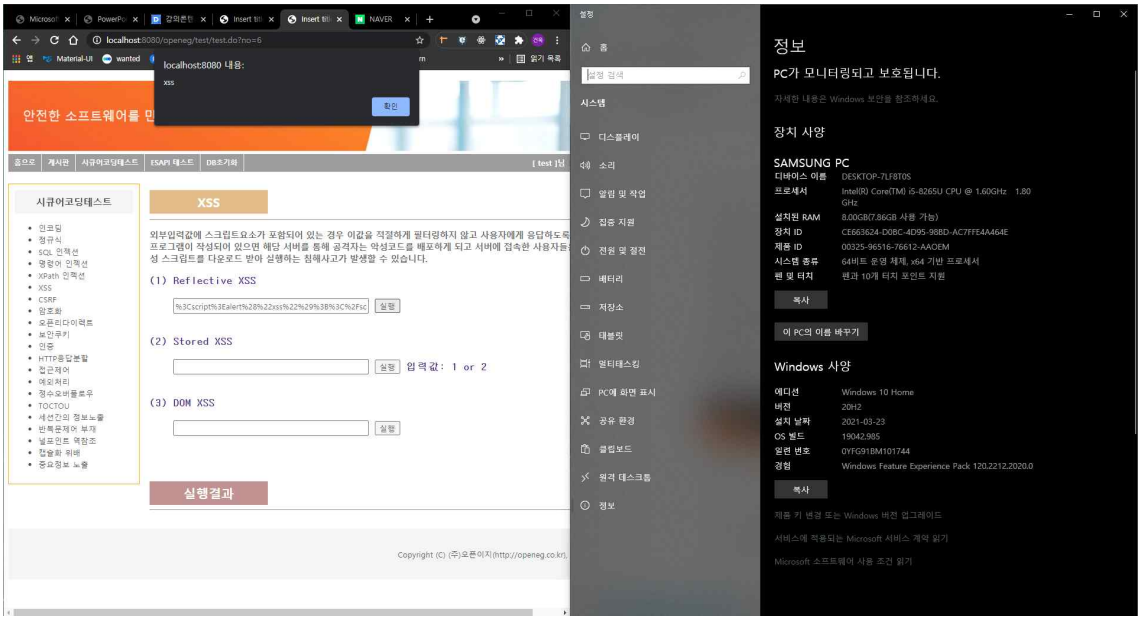
– Stored XSS: 악성 스크립트를 데이터베이스에 저장하여 모든 사용자들이 해당 스크립트를 실행하게 함으로써 악성 행위를 수행한다.

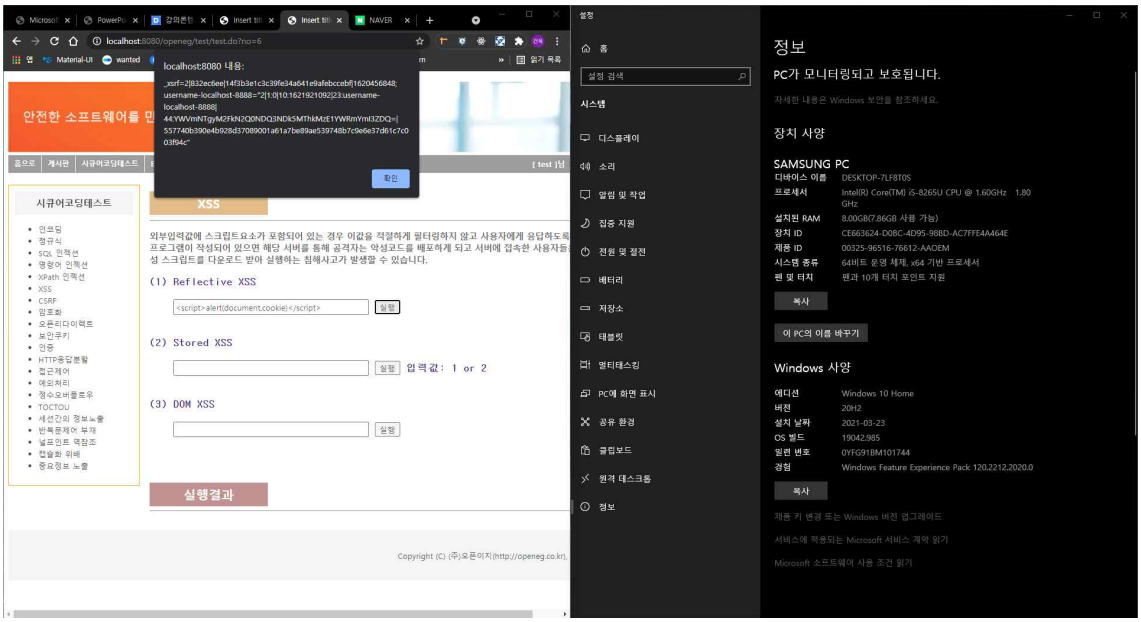
– XSS 취약점의 발생 원인: 사용자의 입력 값이나 데이터베이스에서 검색한 결과값을 검증하지 않고 응답의 일부로 사용하기 때문에 발생한다.

(XSS 공격 차단)
1. 입출력 값에 필터링 적용
2. 오픈소스 라이브러리 활용 XSS Filter(lucy xss filter)

(1) Reflective XSS 공격




localhost:8080 내용:
xss
확인
안전한 소프트웨어를 만
XSS
(1) Reflective XSS
(2) Stored XSS
(3) DOM XSS
실행결과
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양


localhost:8080 내용:
확인
안전한 소프트웨어를 만
XSS
(1) Reflective XSS
(2) Stored XSS
(3) DOM XSS
실행결과
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양

(2) Reflective XSS 방어

(코드수정)

```
// Reflective XSS 테스트
@RequestMapping(value="/test/xss_test.do", method = RequestMethod.POST
@ResponseBody
public String testXss(HttpServletRequest request) {
  StringBuffer buffer=new StringBuffer();
  String data=request.getParameter("data");

  //added
  try {
    data = URLDecoder.decode(data,"UTF-8");
    System.out.println("data:" +data);
  }
  catch(IOException e)
  {
    System.out.println(e);
  }
  XssFilter filter = XssFilter.getInstance("lucy-xss-superset.xml");
  buffer.append(filter.doFilter(data));
    return buffer.toString();

  //buffer.append(data);
    //return buffer.toString();
}
```