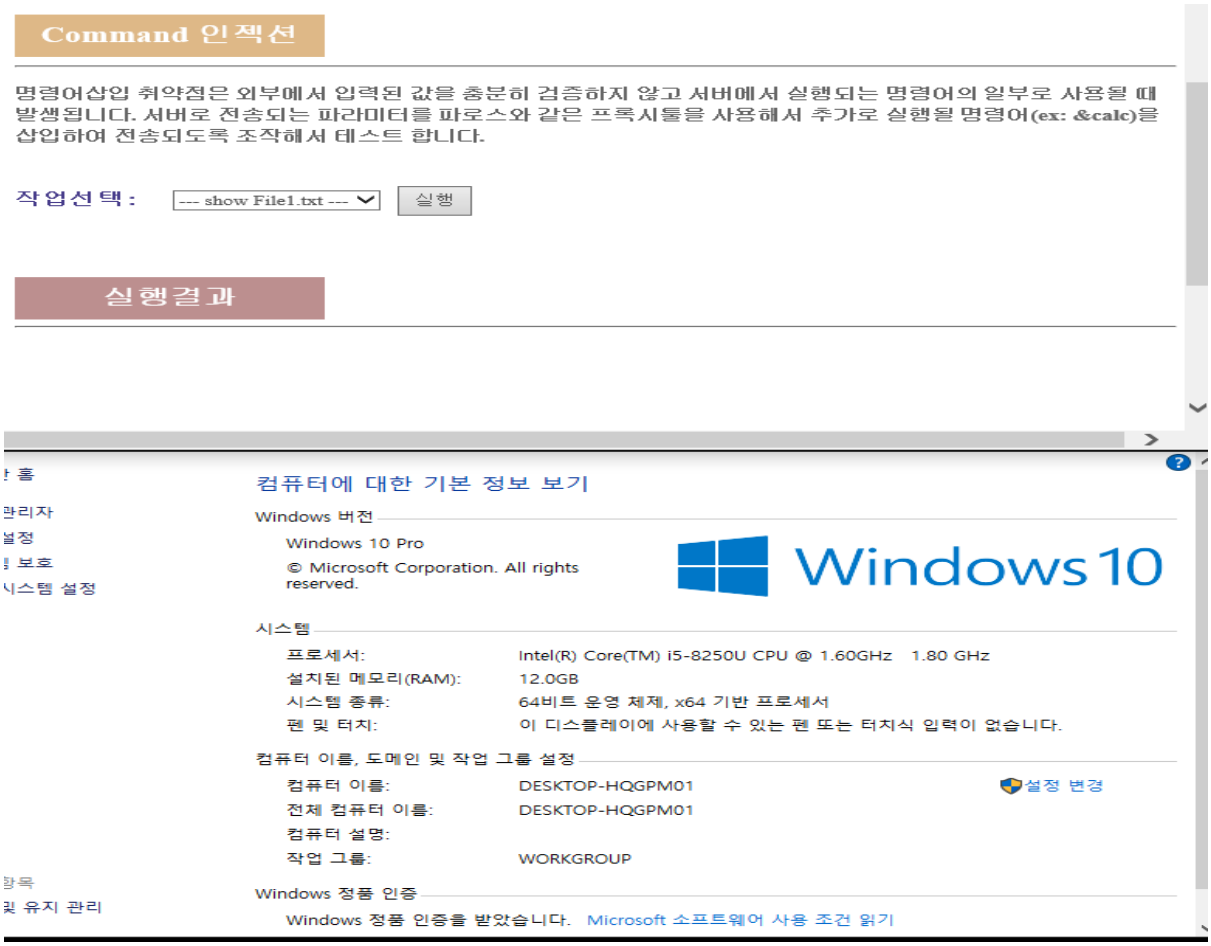
과목명	시큐어코딩
담당교수	우사무엘 교수님
과제명	12주차 실습 보고서
학과	소프트웨어학과
학번	32151671
이름	박민혁
제출일자	2021-05-23

1. 운영체제 명령어 삽입

Original Code

/Command 인젝션
@RequestMapping(value="/test/command_test.do", method = RequestMethod.POST)
@ResponseBody
public String testCommandInjection(HttpServletRequest request, HttpSession session){
StringBuffer buffer=new StringBuffer();
String data=request.getParameter("data");
 	
if (data != null && data.equals("type")) {
data=data+" "+
request.getSession().getServletContext().getRealPath("/")+
"file1.txt";
System.out.println(data);
}



data=type

Command 인젝션

명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)을 삽입하여 전송되도록 조작해서 테스트 합니다.

작업선택 : --- show File1.txt --- 실행

실행결과

실행결과:
Hello Kim !! http://openeg.co.kr

Command 인젝션

명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)를 삽입하여 전송되도록 조작해서 테스트 합니다.

작업선택 : --- show Dir --- 실행

실행결과

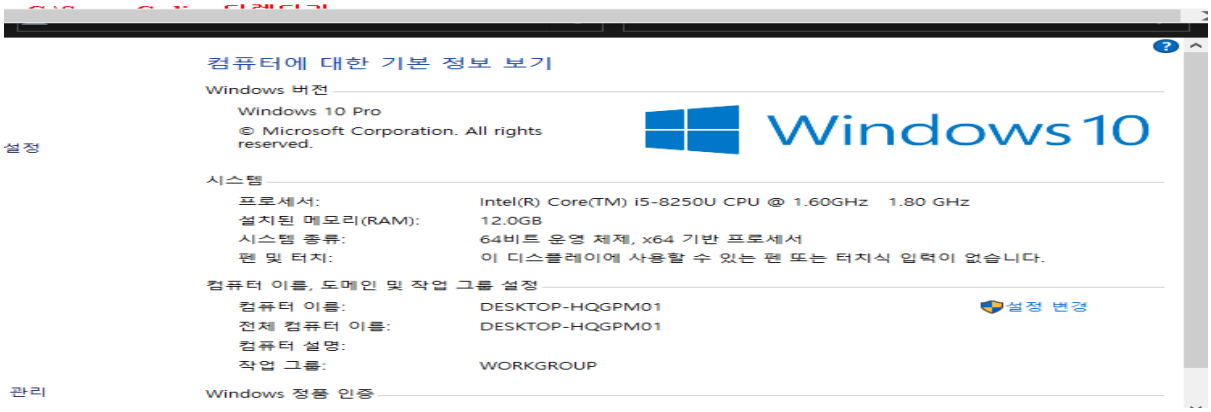
Command 인젝션

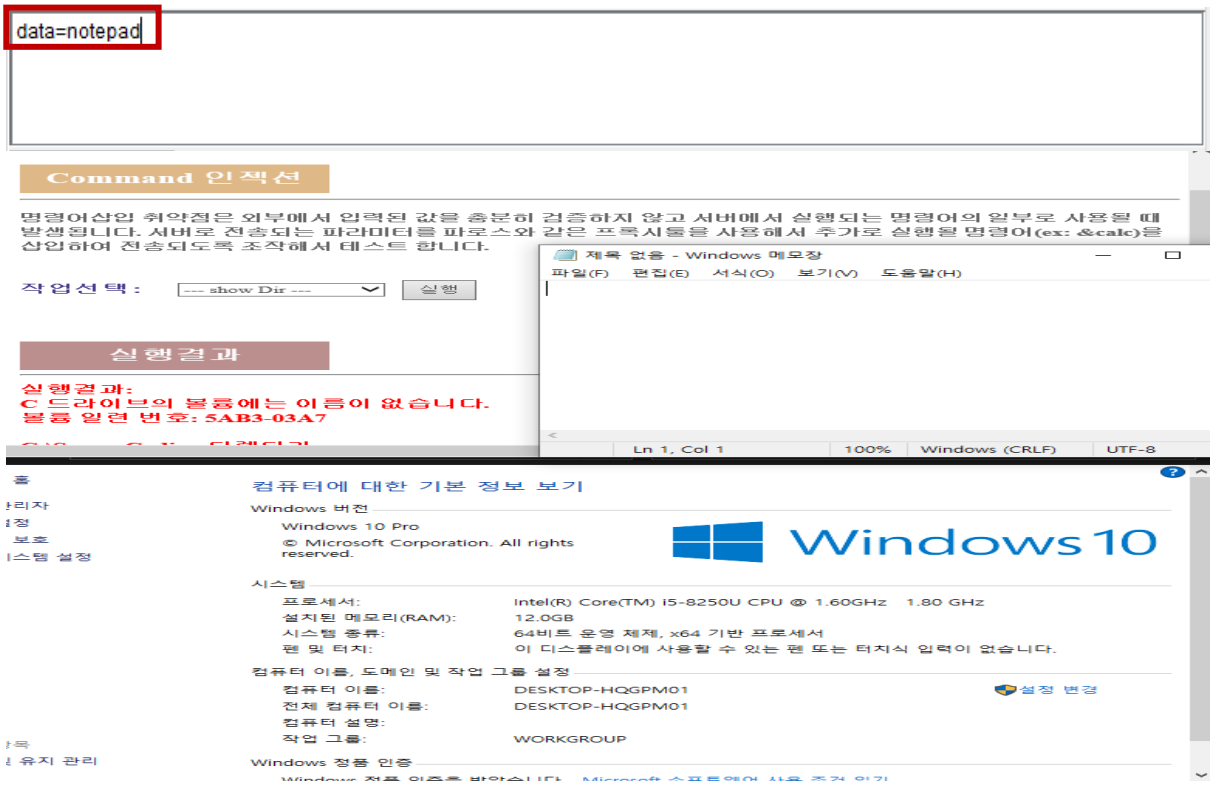
명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)를 삽입하여 전송되도록 조작해서 테스트 합니다.

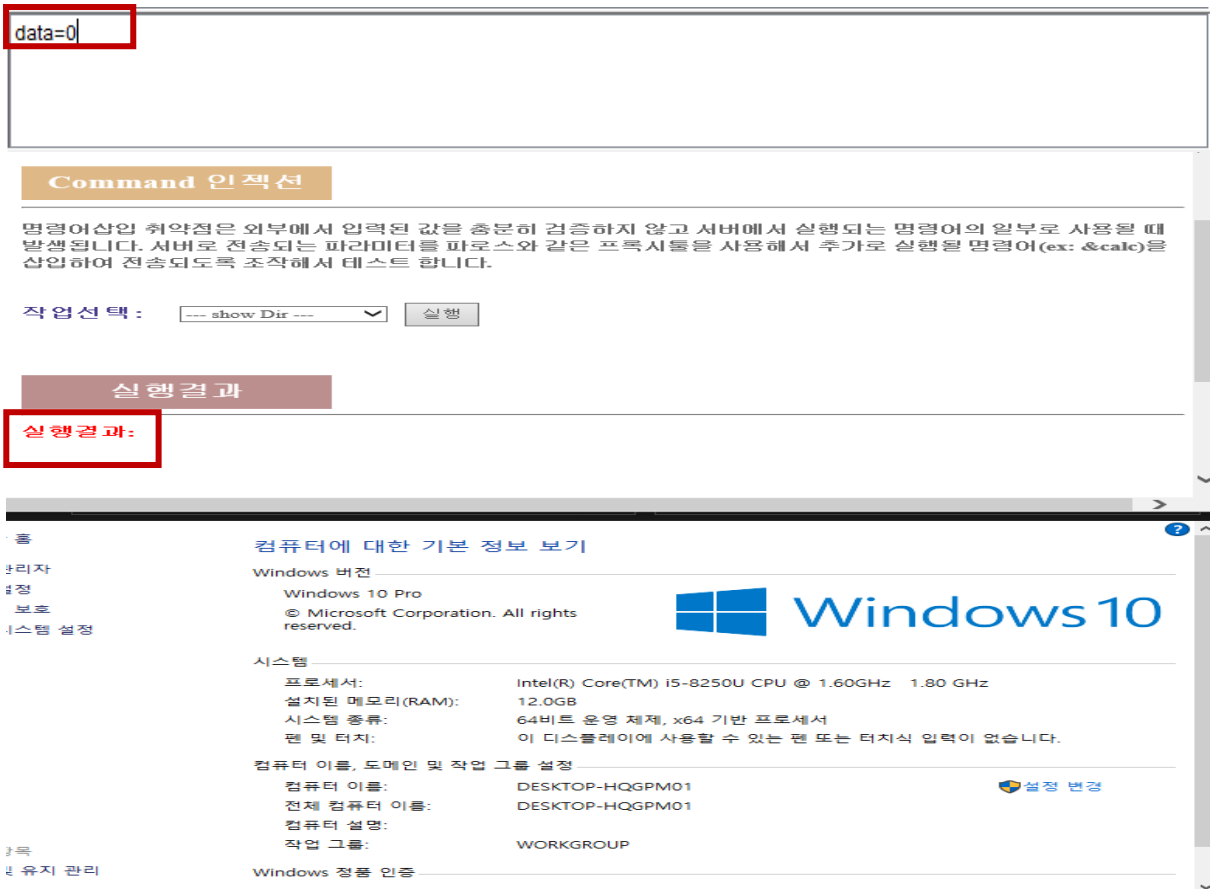
작업선택 : --- show Dir --- 실행

실행결과

실행결과:
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 5AB3-03A7

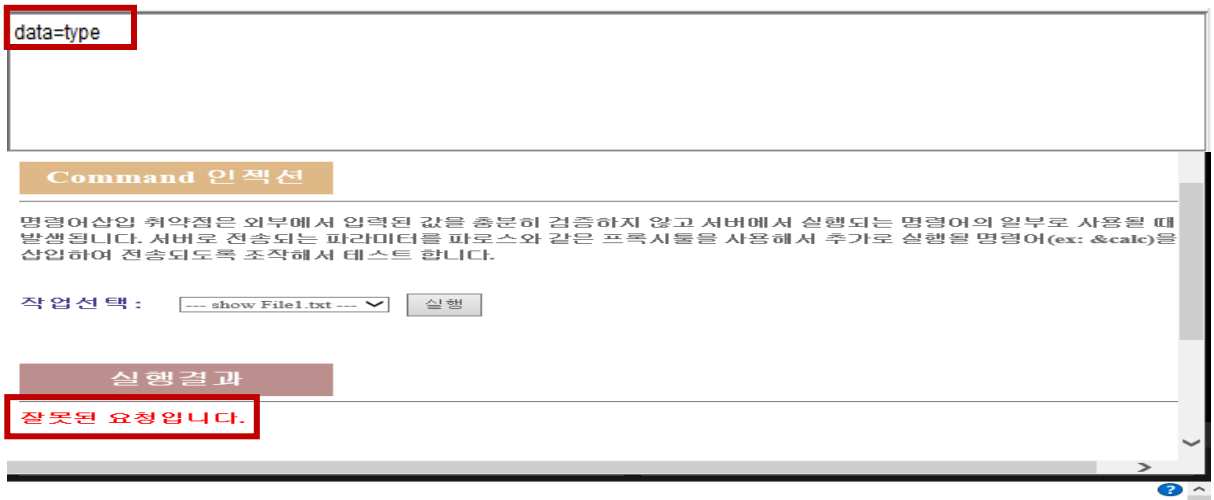


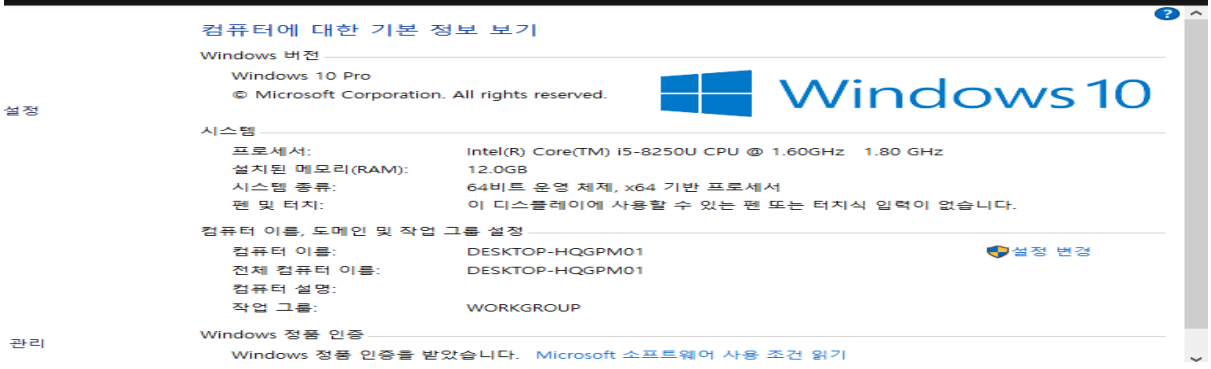

data=notepad
Command 인젝션
명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)를 삽입하여 전송되도록 조작해서 테스트 합니다.
작업선택 :
--- show Dir ---
실행
실행결과
실행결과:
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 5AB3-03A7
제목 없음 - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8
컴퓨터에 대한 기본 정보 보기
Windows 버전
Windows 10 Pro
© Microsoft Corporation. All rights reserved.
Windows 10
시스템
프로세서: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
설치된 메모리(RAM): 12.0GB
시스템 종류: 64비트 운영 체제, x64 기반 프로세서
펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.
컴퓨터 이름, 도메인 및 작업 그룹 설정
컴퓨터 이름: DESKTOP-HQGPM01
전체 컴퓨터 이름: DESKTOP-HQGPM01
컴퓨터 설명:
작업 그룹: WORKGROUP
설정 변경
Windows 정품 인증

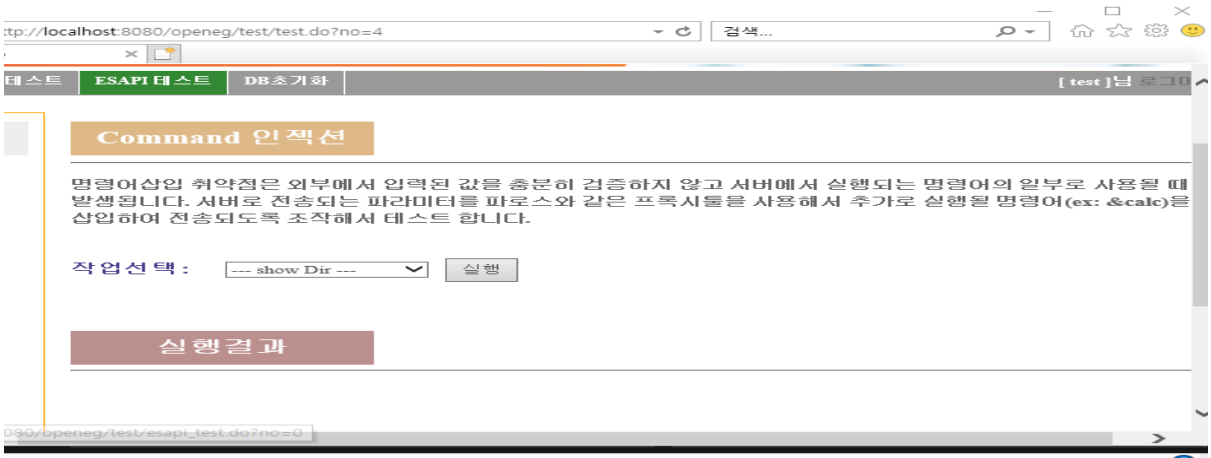

data=0
Command 인젝션
명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)를 삽입하여 전송되도록 조작해서 테스트 합니다.
작업선택 :
--- show Dir ---
실행
실행결과
실행결과:
컴퓨터에 대한 기본 정보 보기
Windows 버전
Windows 10 Pro
© Microsoft Corporation. All rights reserved.
Windows 10
시스템
프로세서: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
설치된 메모리(RAM): 12.0GB
시스템 종류: 64비트 운영 체제, x64 기반 프로세서
펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.
컴퓨터 이름, 도메인 및 작업 그룹 설정
컴퓨터 이름: DESKTOP-HQGPM01
전체 컴퓨터 이름: DESKTOP-HQGPM01
컴퓨터 설명:
작업 그룹: WORKGROUP
설정 변경
Windows 정품 인증

Change Code

```
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);
         }
```


data=type
Command 인젝션
명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)를 삽입하여 전송되도록 조작해서 테스트 합니다.
작업선택 :
--- show File1.txt ---
실행
실행결과
잘못된 요청입니다.


컴퓨터에 대한 기본 정보 보기
Windows 버전
Windows 10 Pro
© Microsoft Corporation. All rights reserved.
Windows 10
시스템
프로세서: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
설치된 메모리(RAM): 12.0GB
시스템 종류: 64비트 운영 체제, x64 기반 프로세서
펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.
컴퓨터 이름, 도메인 및 작업 그룹 설정
컴퓨터 이름: DESKTOP-HQGPM01
설정 변경
전체 컴퓨터 이름: DESKTOP-HQGPM01
컴퓨터 설명:
작업 그룹: WORKGROUP
Windows 정품 인증
Windows 정품 인증을 받았습니다. Microsoft 소프트웨어 사용 조건 읽기


http://localhost:8080/openeg/test/test.do?no=4
검색...
ESAPI 테스트
DB초기화
[test]님
Command 인젝션
명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)를 삽입하여 전송되도록 조작해서 테스트 합니다.
작업선택 :
--- show Dir ---
실행
실행결과


컴퓨터에 대한 기본 정보 보기
Windows 버전
Windows 10 Pro
© Microsoft Corporation. All rights reserved.
Windows 10
시스템
프로세서: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
설치된 메모리(RAM): 12.0GB
시스템 종류: 64비트 운영 체제, x64 기반 프로세서
펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.
컴퓨터 이름, 도메인 및 작업 그룹 설정
컴퓨터 이름: DESKTOP-HQGPM01
설정 변경
전체 컴퓨터 이름: DESKTOP-HQGPM01
컴퓨터 설명:
작업 그룹: WORKGROUP
Windows 정품 인증
Windows 정품 인증을 받았습니다. Microsoft 소프트웨어 사용 조건 읽기

data=dir

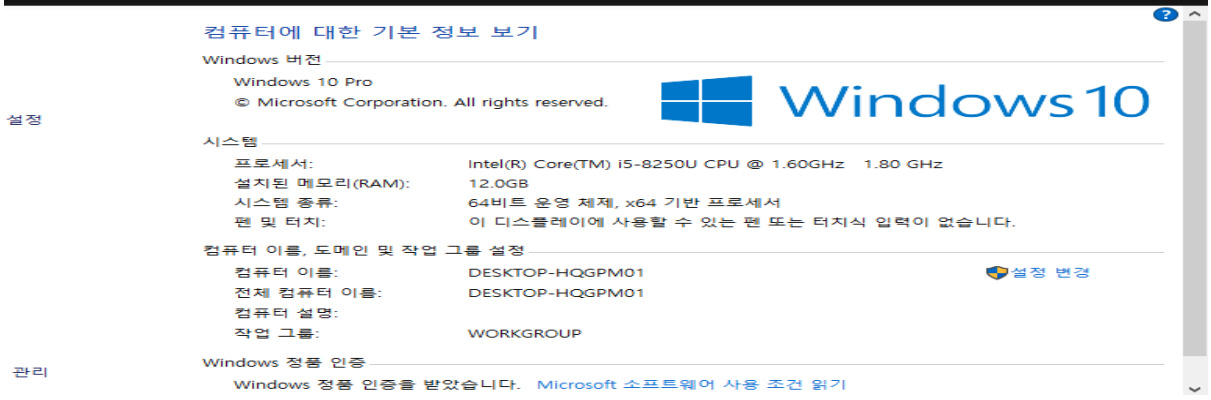
Command 인젝션

명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)를 삽입하여 전송되도록 조작해서 테스트 합니다.

작업선택 : --- show Dir --- 실행

실행결과

잘못된 요청입니다.



data=notepad

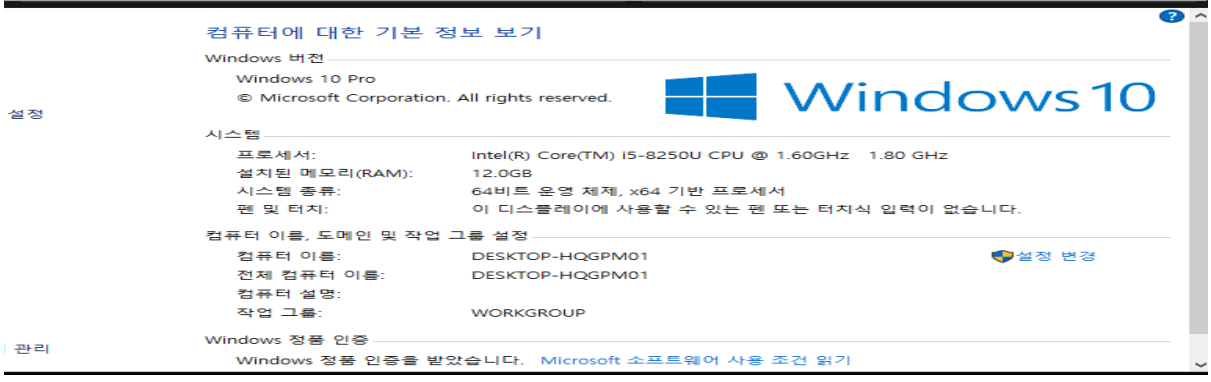
Command 인젝션

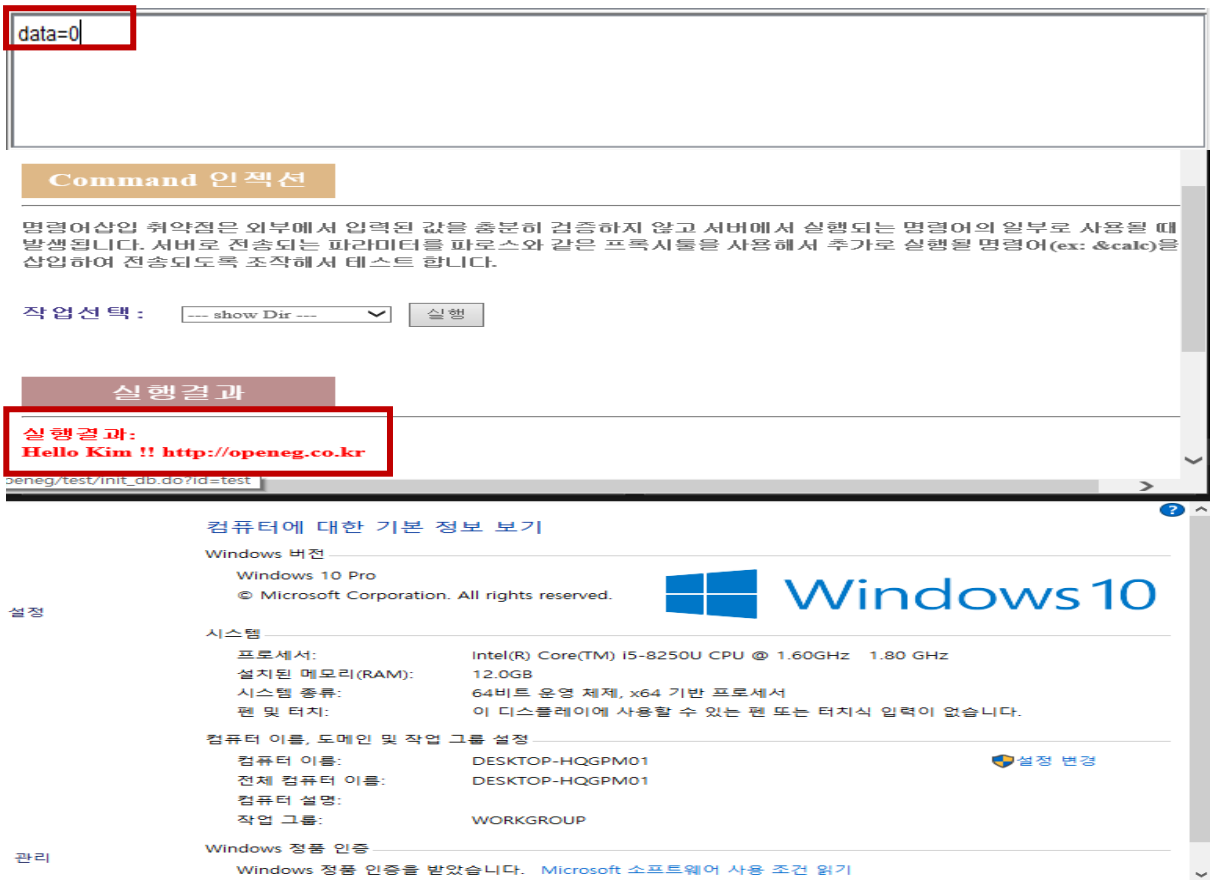
명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)를 삽입하여 전송되도록 조작해서 테스트 합니다.

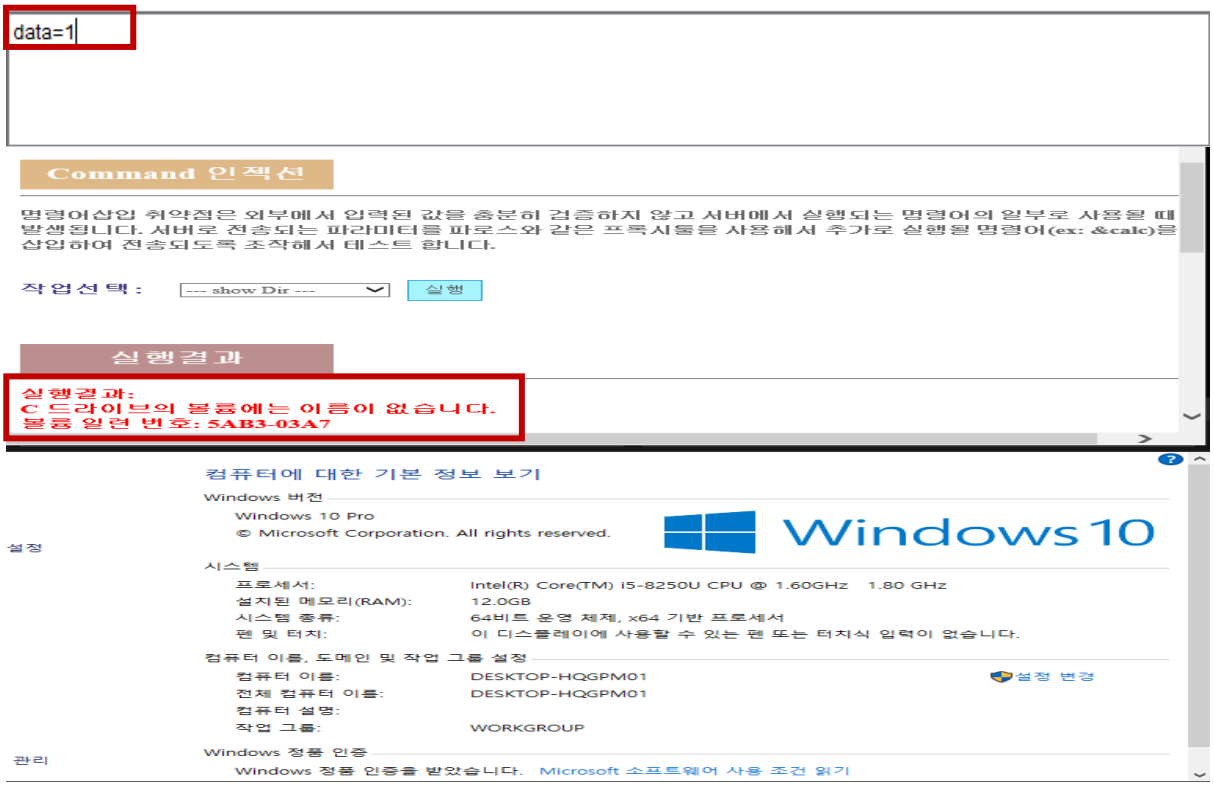
작업선택 : --- show Dir --- 실행

실행결과

잘못된 요청입니다.

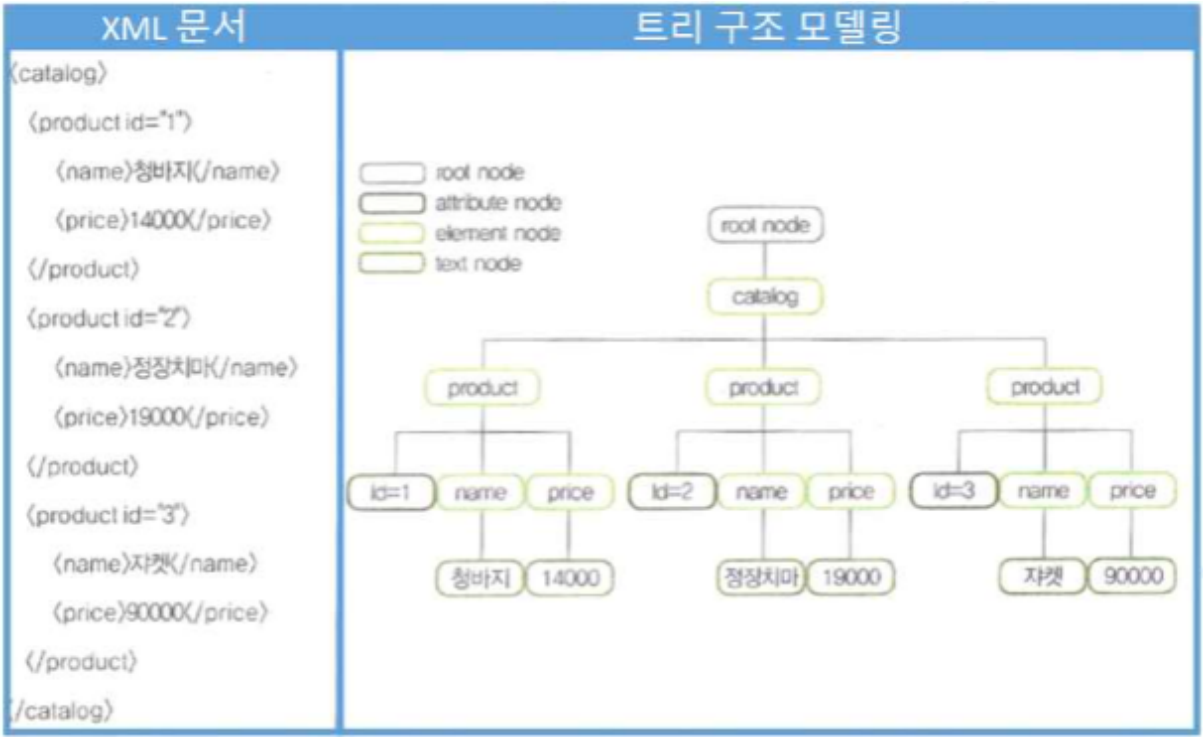



data=0
Command 인젝션
명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)을 삽입하여 전송되도록 조작해서 테스트 합니다.
작업선택 :
--- show Dir ---
실행
실행결과
실행결과:
Hello Kim !! http://openeg.co.kr
컴퓨터에 대한 기본 정보 보기
Windows 버전
Windows 10 Pro
© Microsoft Corporation. All rights reserved.
Windows 10
시스템
프로세서: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
설치된 메모리(RAM): 12.0GB
시스템 종류: 64비트 운영 체제, x64 기반 프로세서
펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.
컴퓨터 이름, 도메인 및 작업 그룹 설정
컴퓨터 이름: DESKTOP-HQGPM01
설정 변경
전체 컴퓨터 이름: DESKTOP-HQGPM01
컴퓨터 설명:
작업 그룹: WORKGROUP
Windows 정품 인증
Windows 정품 인증을 받았습니다. Microsoft 소프트웨어 사용 조건 읽기
설정
관리


data=1
Command 인젝션
명령어삽입 취약점은 외부에서 입력된 값을 충분히 검증하지 않고 서버에서 실행되는 명령어의 일부로 사용될 때 발생됩니다. 서버로 전송되는 파라미터를 파로스와 같은 프록시툴을 사용해서 추가로 실행될 명령어(ex: &calc)을 삽입하여 전송되도록 조작해서 테스트 합니다.
작업선택 :
--- show Dir ---
실행
실행결과
실행결과:
C 드라이브의 볼륨에는 이름이 없습니다.
볼륨 일련 번호: 5AB3-03A7
컴퓨터에 대한 기본 정보 보기
Windows 버전
Windows 10 Pro
© Microsoft Corporation. All rights reserved.
Windows 10
시스템
프로세서: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
설치된 메모리(RAM): 12.0GB
시스템 종류: 64비트 운영 체제, x64 기반 프로세서
펜 및 터치: 이 디스플레이에 사용할 수 있는 펜 또는 터치식 입력이 없습니다.
컴퓨터 이름, 도메인 및 작업 그룹 설정
컴퓨터 이름: DESKTOP-HQGPM01
설정 변경
전체 컴퓨터 이름: DESKTOP-HQGPM01
컴퓨터 설명:
작업 그룹: WORKGROUP
Windows 정품 인증
Windows 정품 인증을 받았습니다. Microsoft 소프트웨어 사용 조건 읽기
설정
관리

2. 입력 값 검증 부재로 인한 삽입 취약점 실습

- XPath : XML 문서에서 특정 요소나 속성까지 도달하기 위한 경로를 요소의 계층을 이용해 표현하는 것이다.

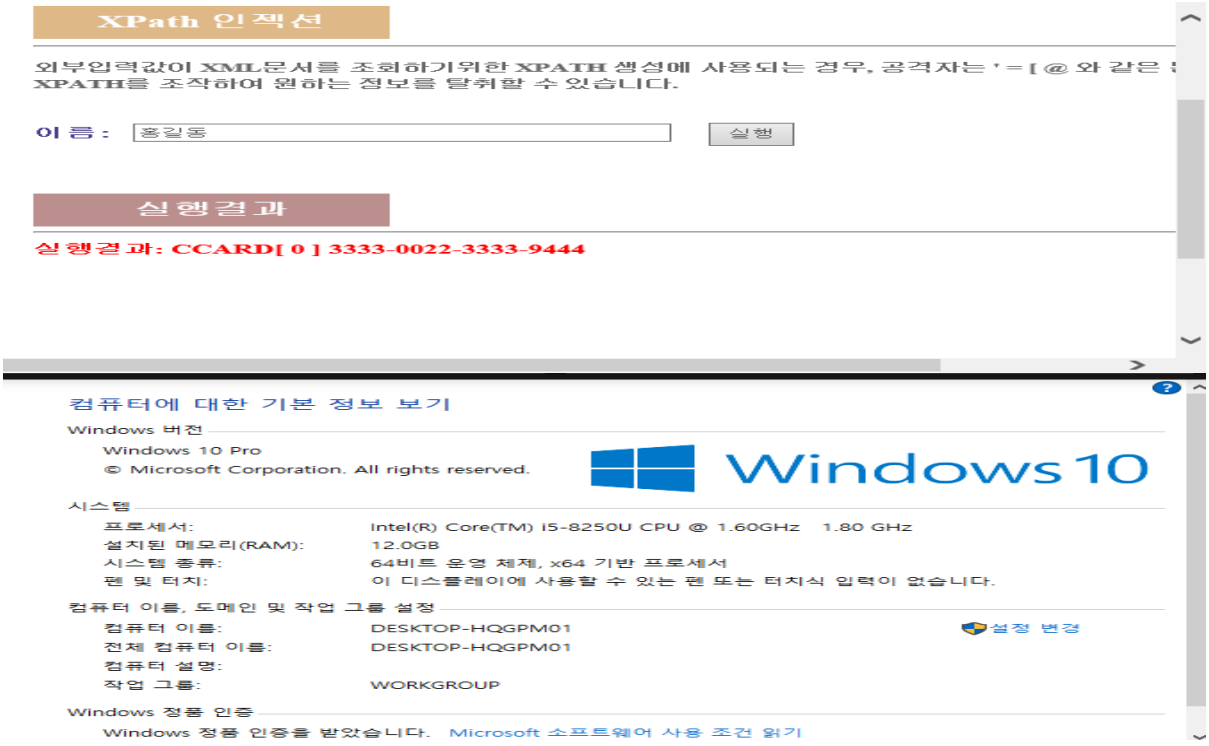


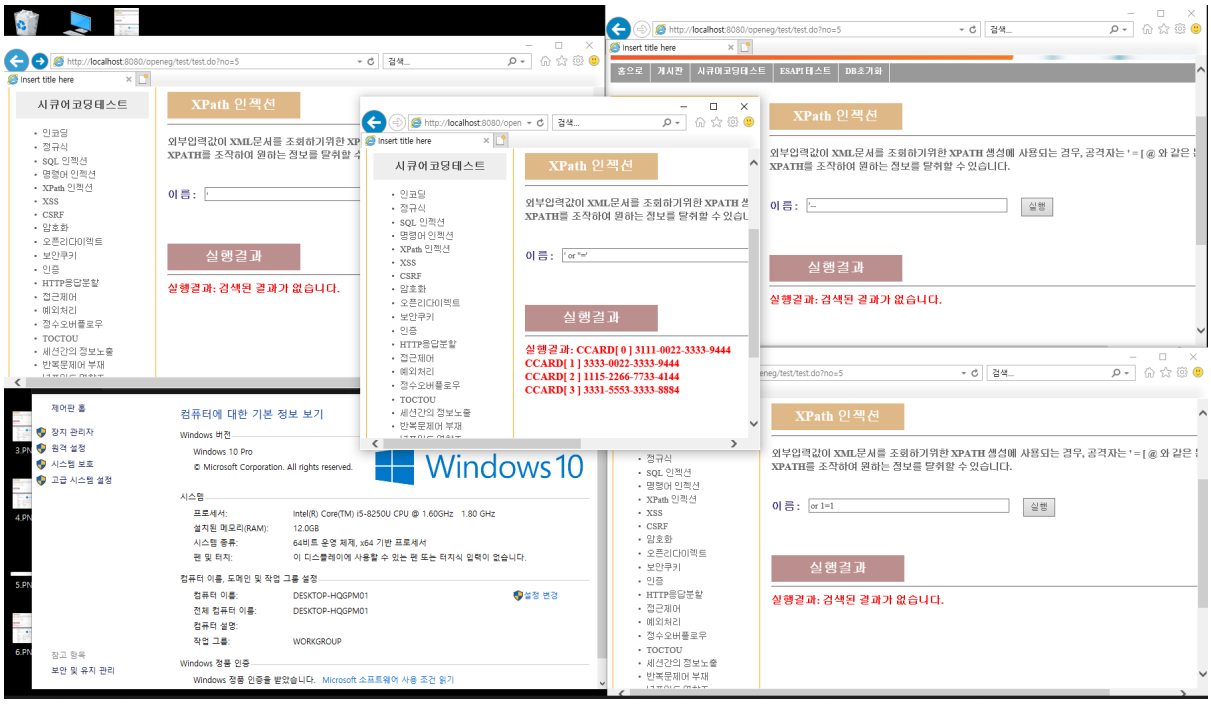
- XPath 삽입 : XML 문서에 저장된 데이터를 애플리케이션에서 검색하거나 읽기 귀해 사용하는 표현 방식이다.

- 취약점 발생 원인 : XPath 쿼리문을 생성할 때, 입력 값에 대한 검증 작업을 수행하지 않고 동적으로 생성되는 XPath문에 삽입해 사용하는 경우 취약점이 발생한다.

(1). XPath 삽입 공격

입력 값이 XML 쿼리의 조회 키로 사용되는지 먼저 체크한다.

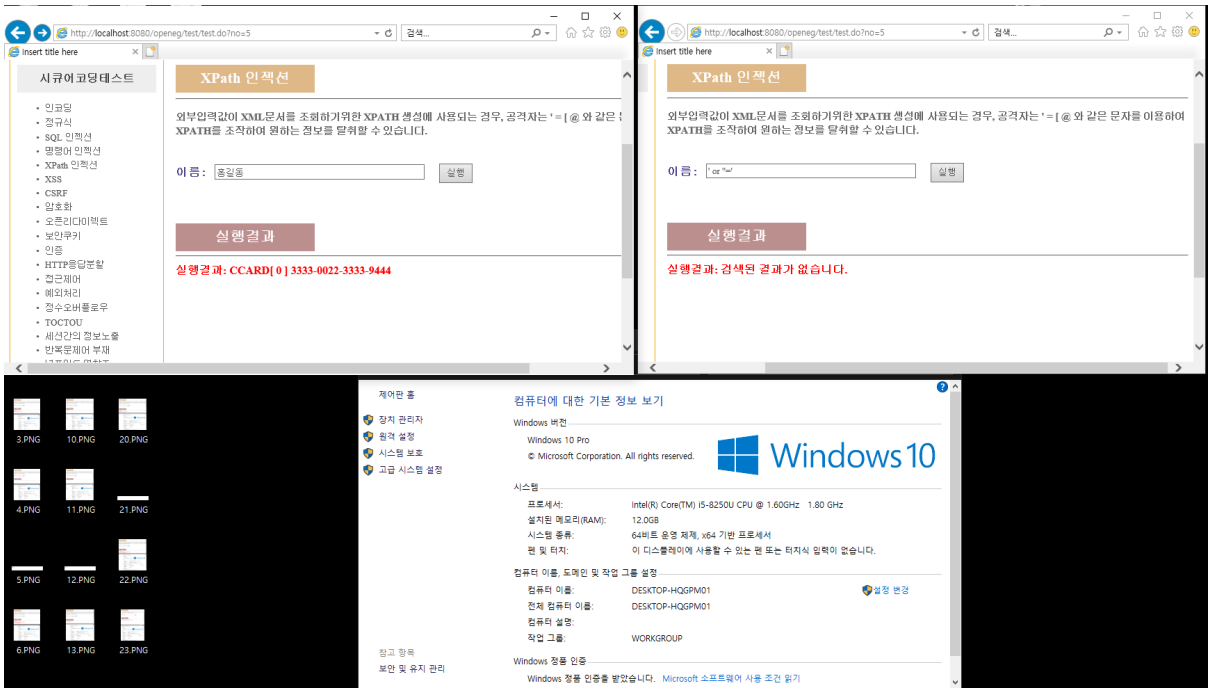




(2). XPath 삽입 방어

XPath에서 사용되는 외부 입력 값에 대해 안전한 값으로 필터링해 사용한다.

```
public String XPathFilter(String input) {
  return input.replaceAll("[ ', \\[]", "");
}
//String expression = "/addresses/address[@name='"+name+"']/ccard";
String expression = "/addresses/address[@name='"+XPathFilter(name)+"']/ccard";
```



- XPathFilter() : XPath 삽입을 발생시킬 수 있는 문자 필터링이다. 쿼리문의 의미를 바꿀 수 있는 특수 문자를 공백으로 변경한다. 따라서 **' or ''='** 구문을 이용하여 허가 되지 않은 파일의 정보를 조회하려 할 때, XPathFilter() 함수에 의해 입력 값이 필터링된다. 결과적으로 안전한 값만 프로그램에서 사용하도록 한다.

3. 크로스 사이트 스크립팅 취약점 실습

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

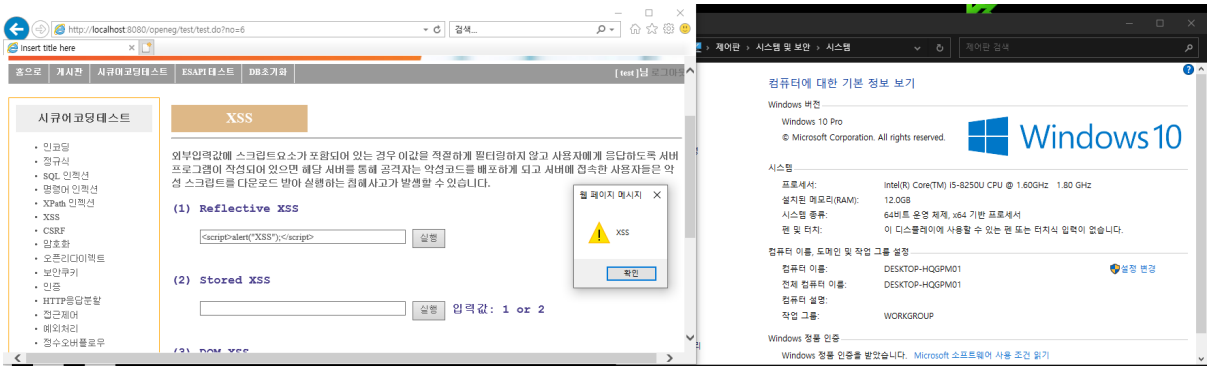
- Reflective XSS : 공격자가 악성 스크립트를 입력 값으로 전달하도록 만들어진 URL을 사용자가 클릭하도록 유도한다.

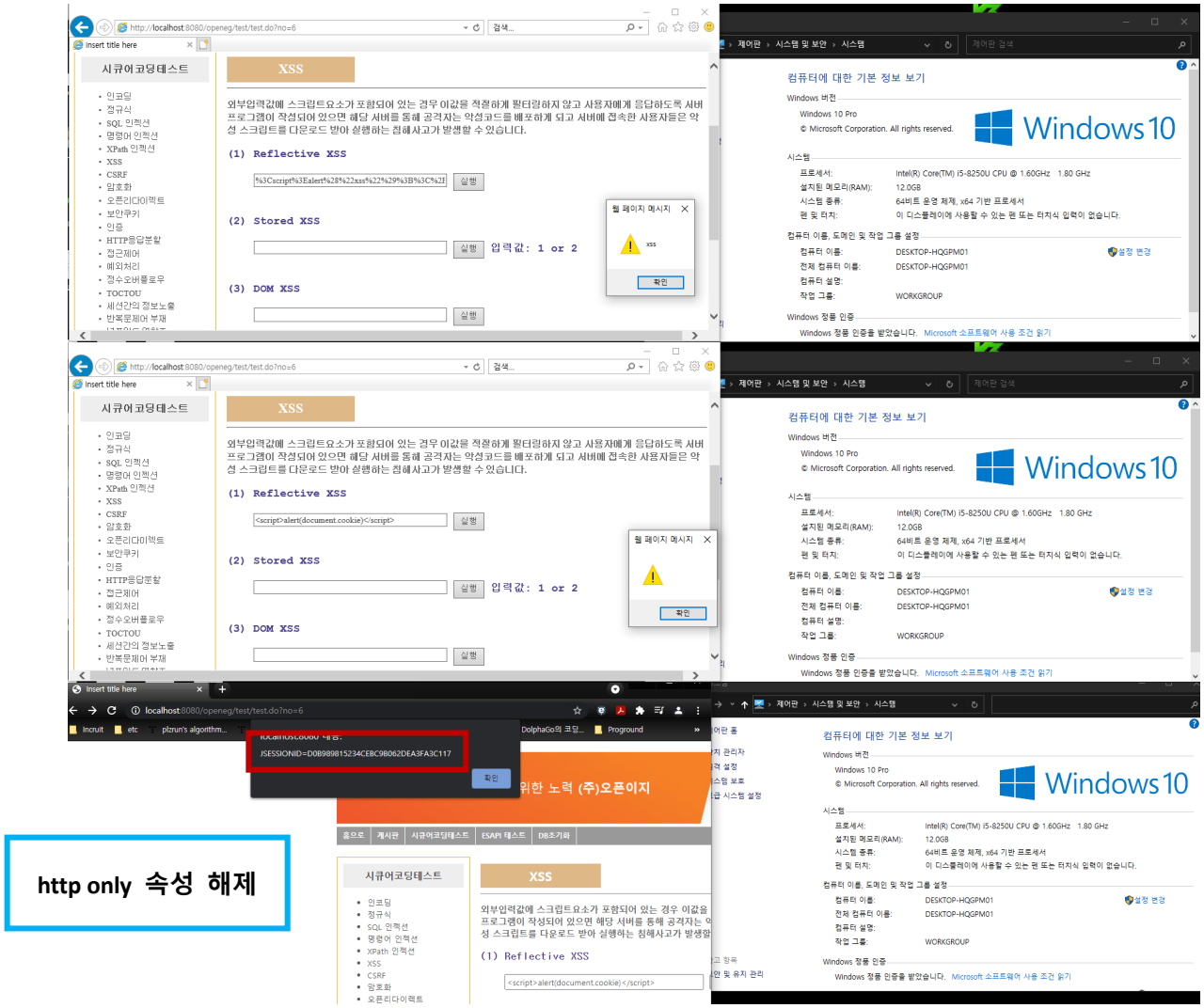
- Stored XSS : 악성 스크립트를 DB에 저장하여 시스템을 사용하는 모든 사용자들이 해당 스크립트를 실행하게 한다. 사용자 쿠키 정보 탈취 및 악성 사이트로 이동시킨다.

- DOM XSS : AJAX 프로그램에서 사용되는 자바스크립트를 이용해 브라우저에 수신된 데이터를 다시 잘라 Write 작업을 수행하는 경우 XSS 공격이 가능하게 한다.

- 취약점 발생 원인 : 외부 사용자의 입력 값이나 DB에서 검색한 결과 값을 검증하지 않고 사용한 경우

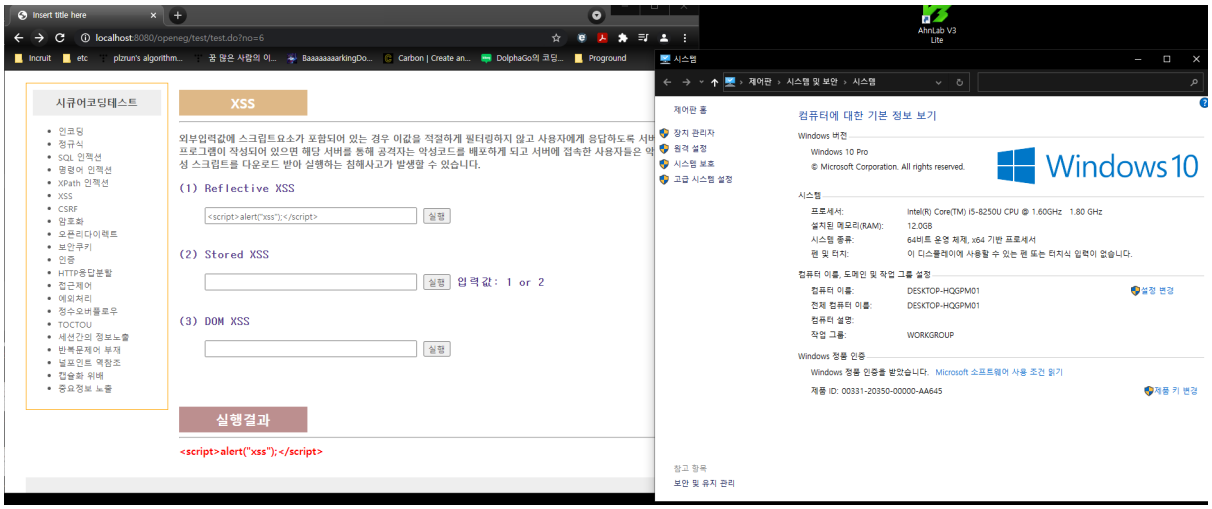
(1). Reflective XSS 공격

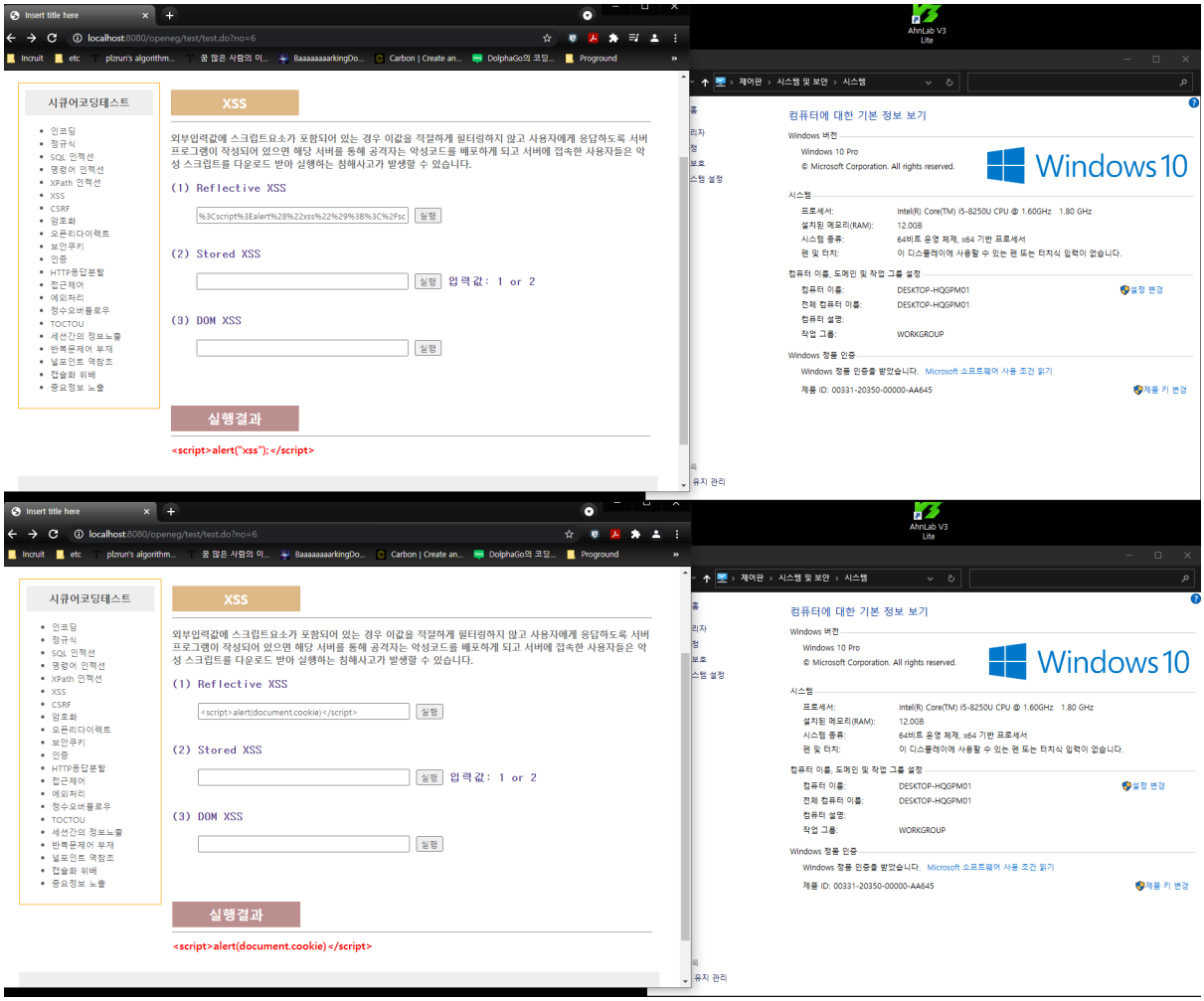




(2) Reflective XSS 방어

오픈소스 라이브러리를 활용하여 입력 값에 대해 XSSFilter 적용






```
// 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");

    try {
        data = URLDecoder.decode(data, "UTF-8");
        System.out.println("Data : " + data);

    }catch(UnsupportedEncodingException e) {
        System.out.println(e);
    }
    XssFilter filter = XssFilter.getInstance("lucy-xss-superset.xml");
    buffer.append(filter.doFilter(data));

    return buffer.toString();
}
```

- 각종 방식으로 인코딩 되어 전달되는 데이터를 필터링 하기 위해서는 인코딩 규칙까지도 감안하여 만들어야 한다. XSS Filter를 만드는 것보다 이미 잘 검증된 필터를 활용하는 것을 권장한다.

- **data = URLDecoder.decode(data, “UTF-8”);**
인코딩 되어 입력되는 값은 디코딩 후 필터를 적용한다.

- **XSSFilter filter = XssFilter.getInstance(“lucy-xss-superset.xml”);**
필터 객체 생성 후 출력 값에 대해 Anti-Xss 필터를 적용한다.

따라서 사용자가 요청을 한 입력 값에 대해 XSS Filter를 적용하여 안전한 값만 전달한다. 그 후 interceptor가 입력 값을 검증하여 안전하지 않은 입력 값에 대해서는 요청을 차단하고, 안전한 입력 값은 넘긴다.