〈 파일 업로드/다운로드 취약점 〉

(파일 업로드)

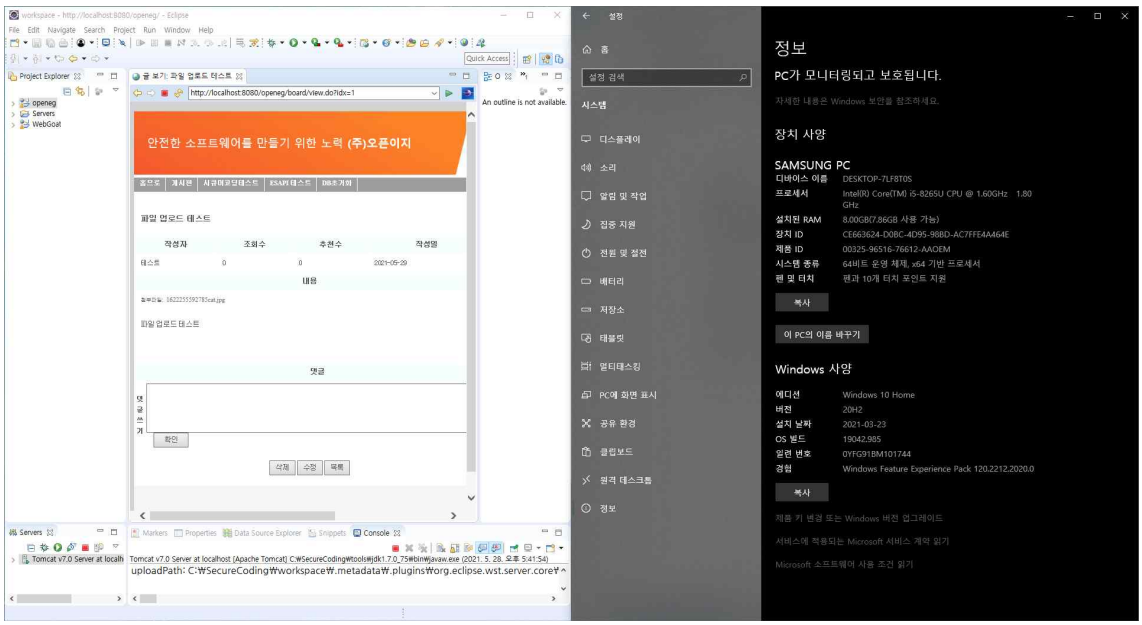
– 업로드 되는 파일의 타입을 제한하지 않는 경우
– 업로드 되는 파일의 크기나 개수를 제한하지 않는 경우
– 업로드 된 파일을 외부에서 직접적으로 접근가능한 경우
– 업로드 된 파일의 이름과 저장된 파일의 이름이 동일하여 공격자가 파일에 대한 인식이 가능한 경우
– 업로드 된 파일이 실행권한이 있는 경우

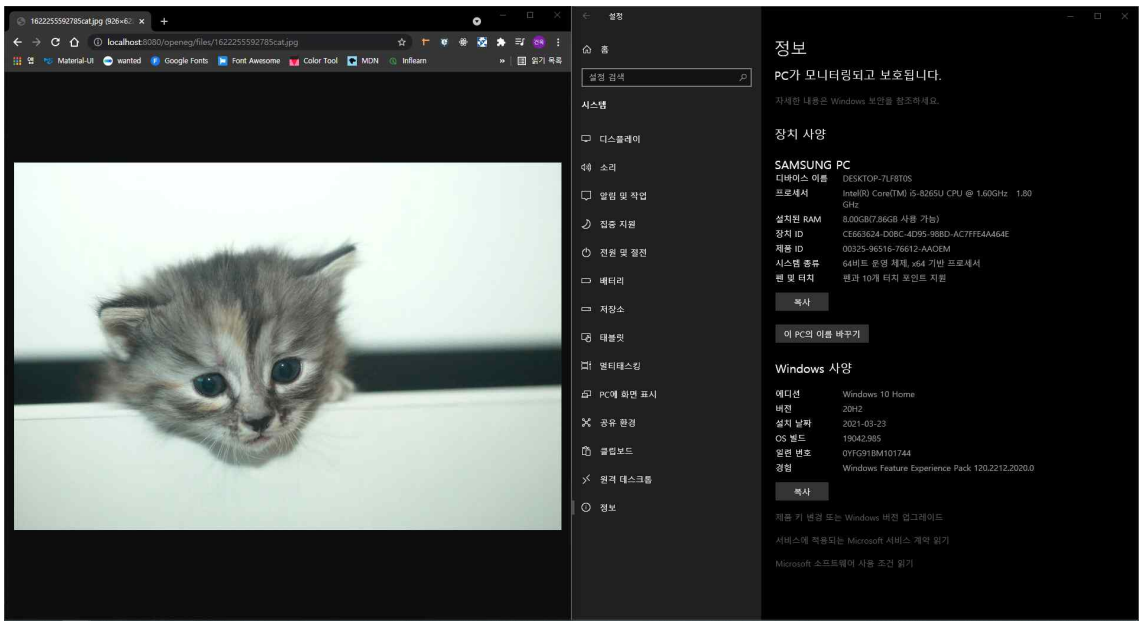
(파일 다운로드)

– 파일에 대한 접근권한이 없는 사용자가 직접적인 경로를 이용하여 파일을 다운로드 할 수 있는 경우
– 악성코드에 감염된 파일을 다운로드 하는 경우

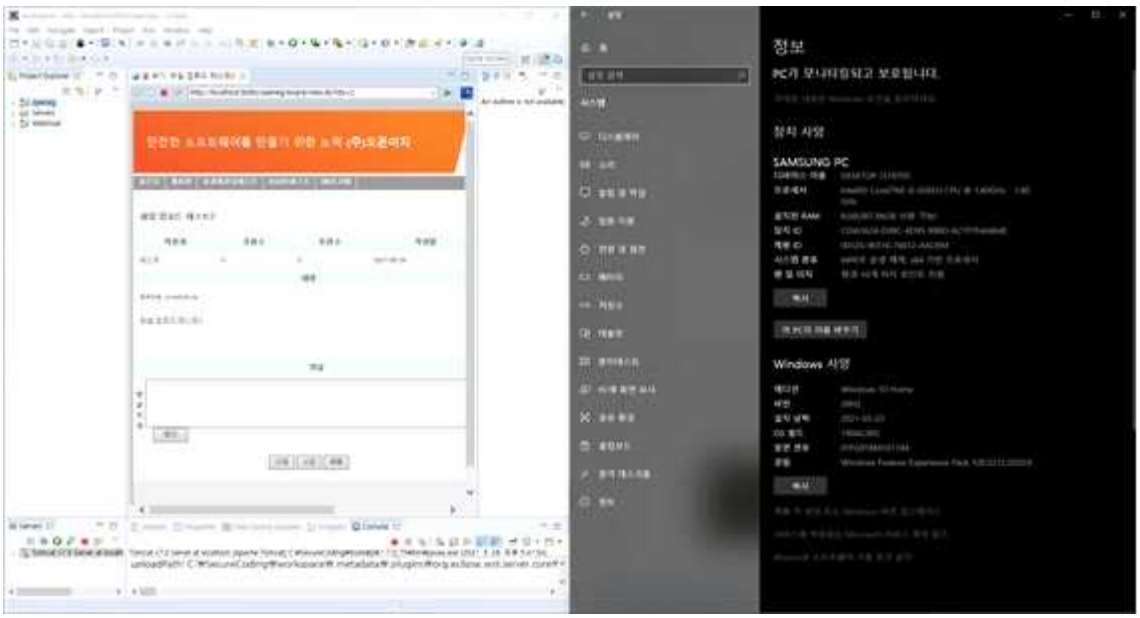
1. 파일 업로드 취약점 공격 실습

– 직접 접근할 수 있는 취약점

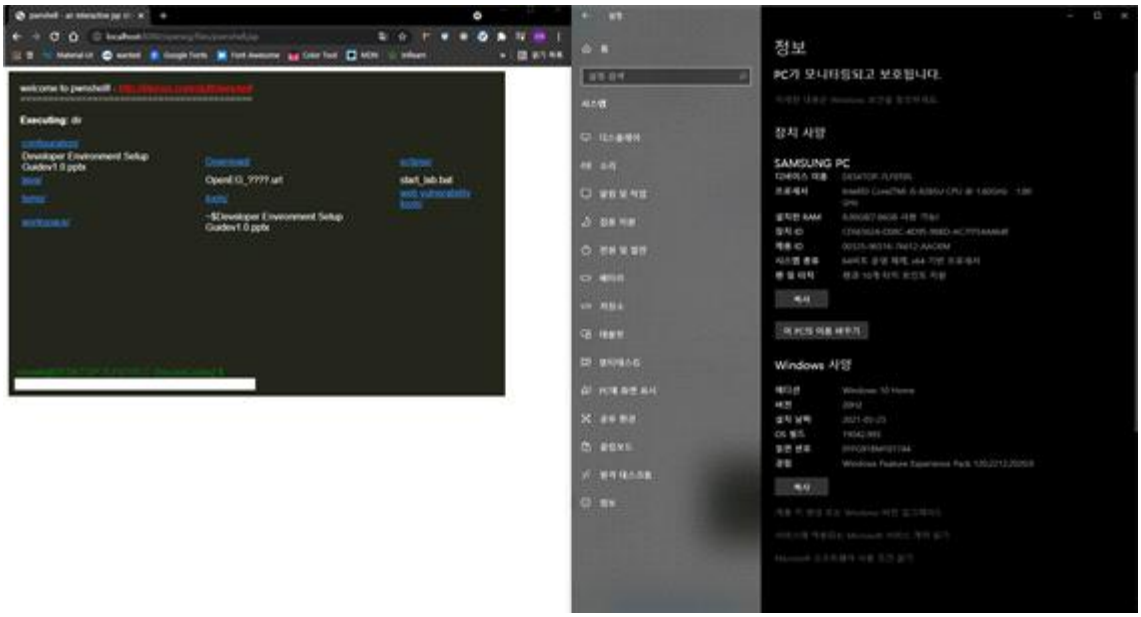




– 웹쉘을 업로드하여 실행 가능 여부 확인



* 웹쉘이 게시판에 게시되어 디렉터리와 파일 구조를 확인할 수 있다.



* 주소를 통해 웹쉘에 직접 접근할 수 있고 명령 또한 웹 서버에서 수행할 수 있다.
* 이처럼 파일의 유형을 확인하지 않는다면 웹쉘과 같은 파일을 업로드하여 공격자의 공격에 적극적으로 활용될 수 있다.

– 파일 업로드 취약점 방어 실습

(코드수정)

BoardController.java

```
// added by secure_coding
@RequestMapping(value="/write.do", method=RequestMethod.POST)
public String boardWriteProc(@ModelAttribute("BoardModel") BoardModel boardModel,
    MultipartHttpServletRequest request, HttpSession session){
  // get upload file
  String uploadPath = session.getServletContext().getRealPath("/")+"WEB-INF/files/";
  System.out.println("uploadPath: "+uploadPath);
  MultipartFile file = request.getFile("file");
  // 업로드 되는 파일 사이즈 제한 / 특정 파일 업로드 제한하여 업로드 차단
  if ( file != null && ! "".equals(file.getOriginalFilename()) && file.getSize() < 1024000 && file.getContentType().contains("image")
    String fileName = file.getOriginalFilename();
    if(fileName.toLowerCase().endsWith(".jpg")) {
      // 저장할 파일명을 랜덤하게 생성하여 사용한다.
      String savedFileName = UUID.randomUUID().toString();
      File uploadFile = new File(uploadPath + savedFileName);
      try {
        file.transferTo(uploadFile); // 실제 파일이 저장되는 라인
      }
      catch(Exception e) {
        System.out.println("upload error");
      }
      boardModel.setFileName(fileName);
      boardModel.setSavedFileName(savedFileName);
    }
  }

  String content =  boardModel.getContent().replaceAll("₩r₩n", "<br />");
  boardModel.setContent(content);

  service.writeArticle(boardModel);

  return "redirect:list.do";
}
```

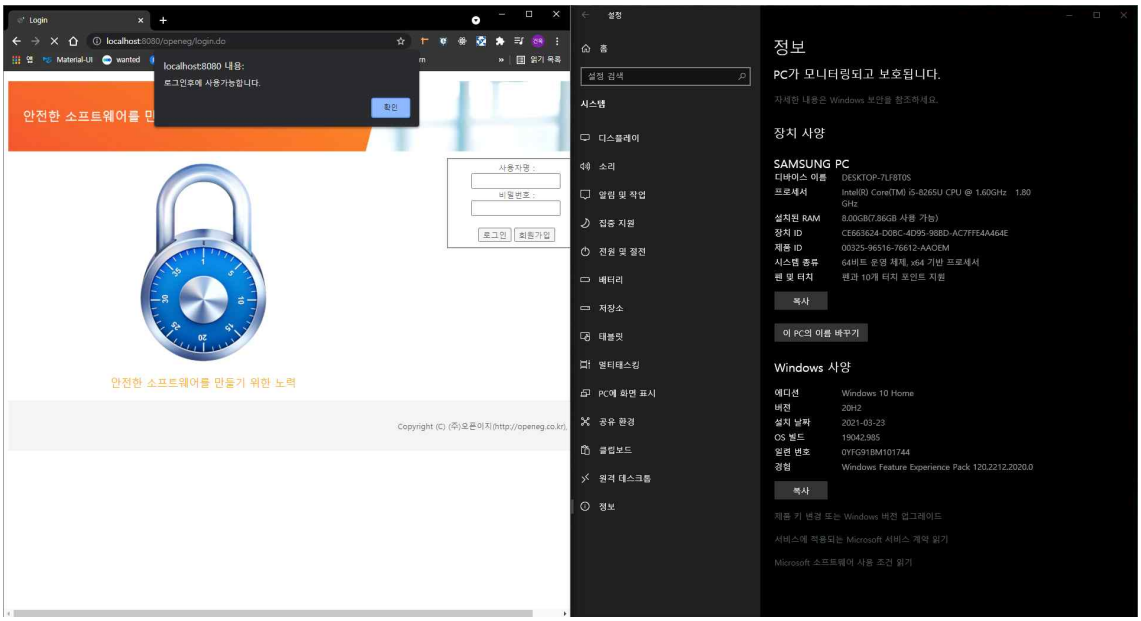
```
// added by secure_coding
@RequestMapping("get_image.do")
public void getimage(HttpServletRequest request, HttpSession session, HttpServletResponse response) {
  int idx = TestUtil.getInt(request.getParameter("idx"));
  // 읽어본 게시물인지 확인
  if(session.getAttribute("idx")==null || (Integer)session.getAttribute("idx") != idx ) {
    return;
  }
  // 저장된 파일명을 읽어오는 작업이 필요
  BoardModel board = service.getOneArticle(idx);

  String filePath = session.getServletContext().getRealPath("/") + "WEB_INF/files/" + board.getSavedFileName
  System.out.println("filename: " + filePath);
  BufferedOutputStream out=null;
  InputStream in=null;
  try {
    response.setContentType("image/jpeg");
    response.setHeader("Content-Disposition",  "inline;filename="+board.getFileName());
    File file = new File(filePath);
    in=new FileInputStream(file);
    out=new BufferedOutputStream(response.getOutputStream());
    int len;
    byte[] buf = new byte[1024];
    while((len=in.read(buf)) > 0) {
      out.write(buf,0,len);
    }
  } catch(Exception e) {
    e.printStackTrace();
    System.out.println("파일 전송 에러");
  } finally {
    if(out != null) try {out.close();}catch(Exception e) {}
    if(in != null) try {in.close();}catch(Exception e) {}
  }
}
```

view.jsp

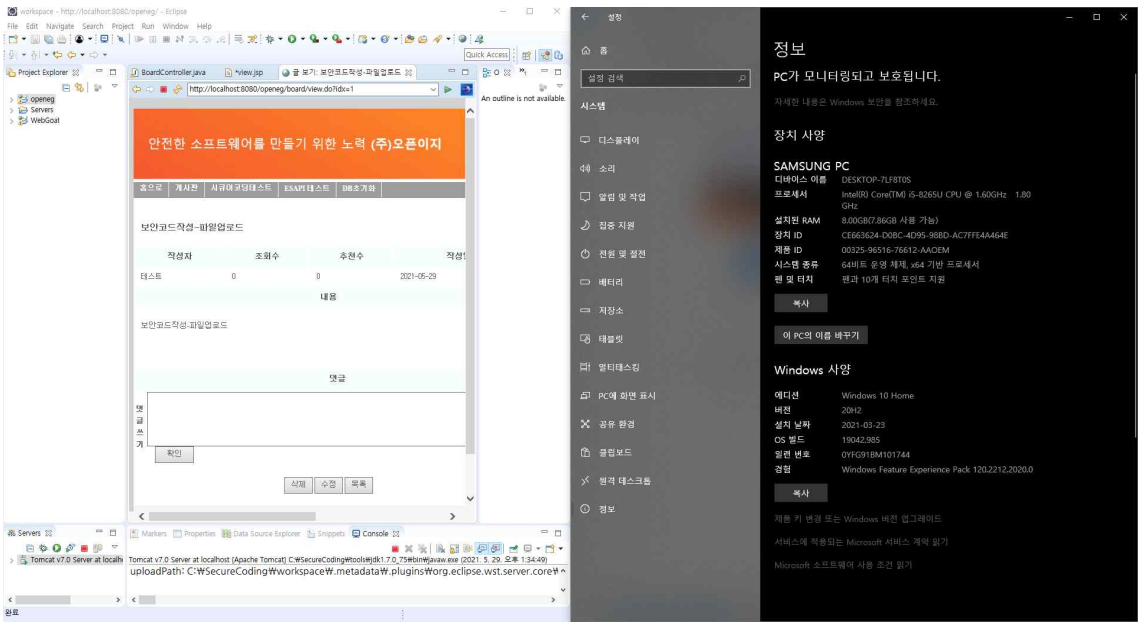
```
<a href="get_image.do?idx=${board.idx}"
target="_blank">${board.fileName}</a></span></td>
```

– 직접 접근 테스트



* 직접 접근이 불가능하도록 방어가 완벽하게 구현되었다.

– 웹쉘 파일 업로드 테스트



* 웹쉘 파일이 업로드 되지 않는 것을 확인할 수 있고 이미지 파일 이외ㅗ에는 모두 필터링하여 업로드를 차단하는 것을 확인할 수 있다.

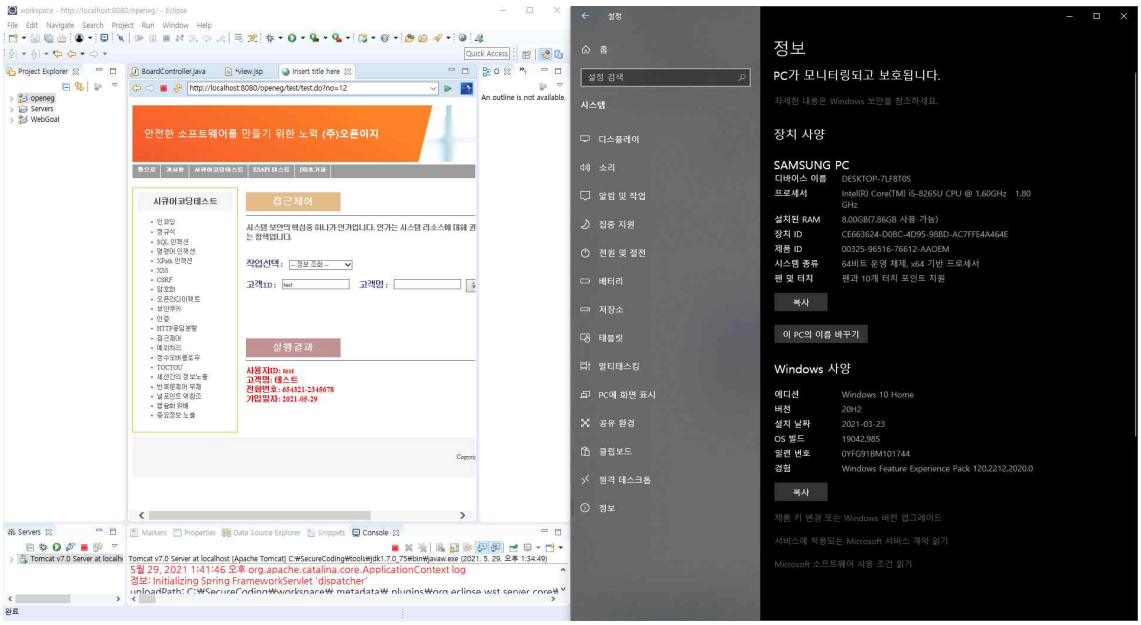
※ 이처럼 파일을 업로드할 때 사전에 필터링하거나 직접 접근할 수 없는 위치에 저장하는 코드를 작성한다면 파일 업로드 취약점을 이용한 공격을 사전에 차단할 수 있다.

〈 파라미터 조작과 잘못된 접근제어 취약점 원인 〉

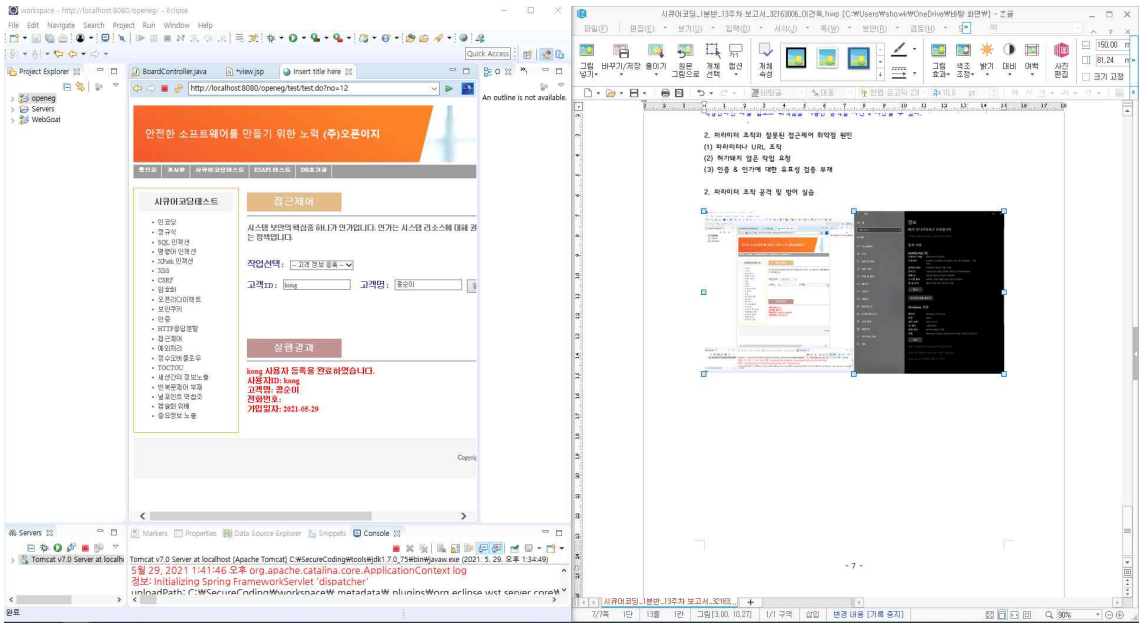
- 파라미터나 URL 조작
- 허가돼지 않은 작업 요청
- 인증 & 인가에 대한 유효성 검증 부재

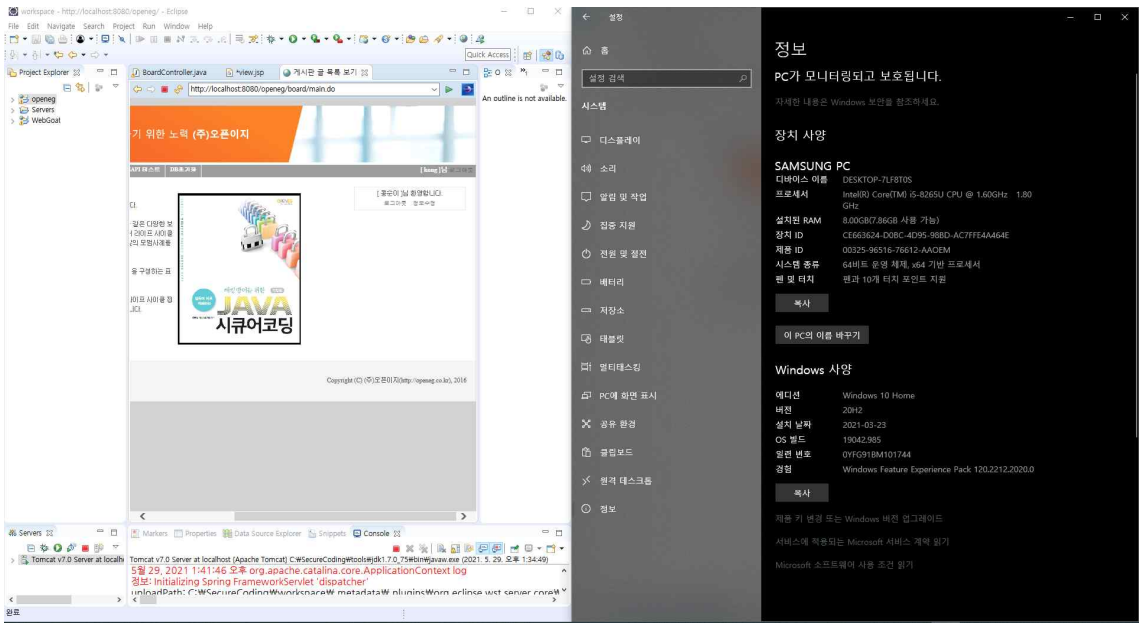
2. 파라미터 조작 공격 및 방어 실습

(사용자 조회)

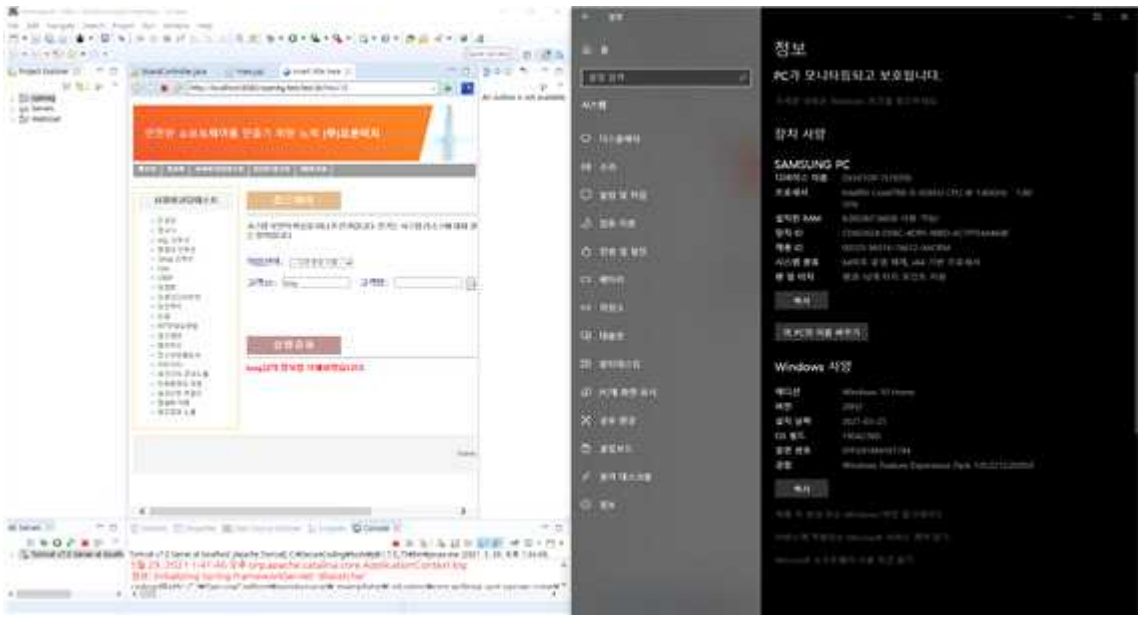


(사용자 등록)

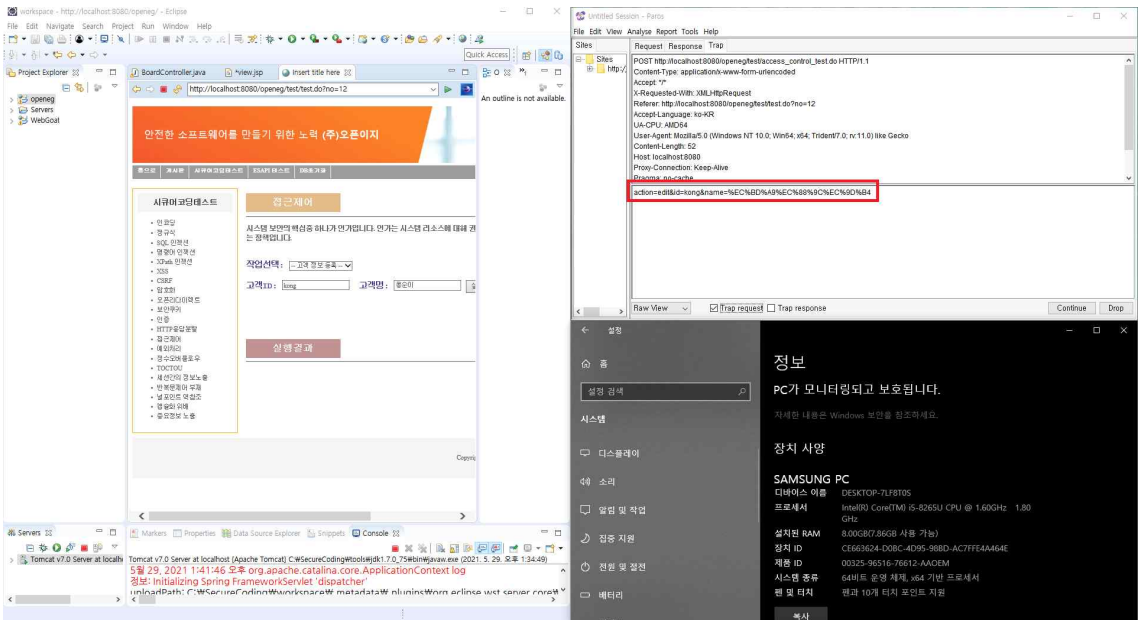

안전한 소프트웨어를 만들기 위한 노력 (주)오픈이지
시큐어코딩테스트
접근제어
실행결과
kong 사용자 등록을 완료하였습니다.
사용자ID: kong
고객명: 콩순이
가입일자: 2021-05-29
5월 29, 2021 1:41:46 오후 org.apache.catalina.core.ApplicationContext log
정보: Initializing Spring FrameworkServlet 'dispatcher'


JAVA
시큐어코딩
정보
PC가 모니터링되고 보호됩니다.
장치 사양
SAMSUNG PC
Windows 사양
5월 29, 2021 1:41:46 오후 org.apache.catalina.core.ApplicationContext log
정보: Initializing Spring FrameworkServlet 'dispatcher'

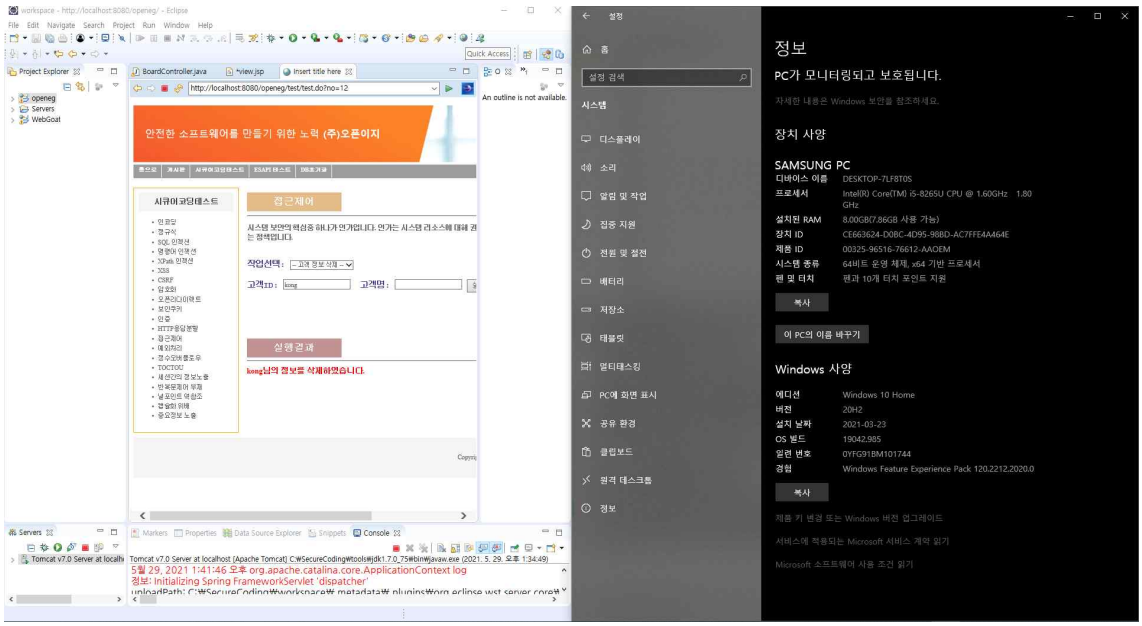
(사용자 삭제)



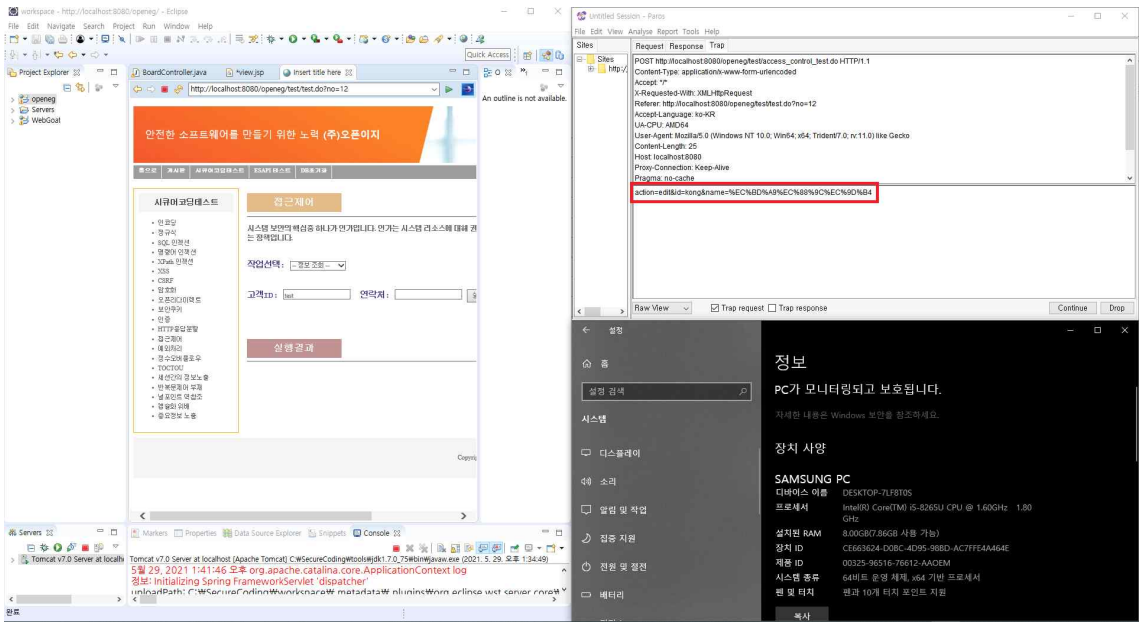
(사용자 등록 - 프록시)



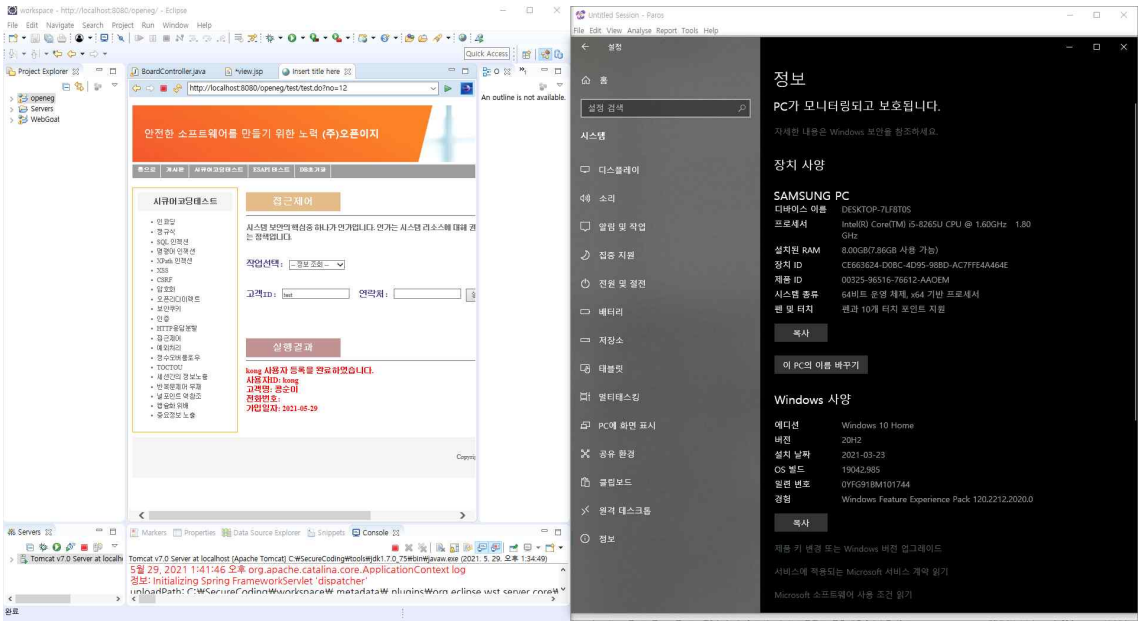
(파라미터 조작 공격을 위한 kong 삭제)

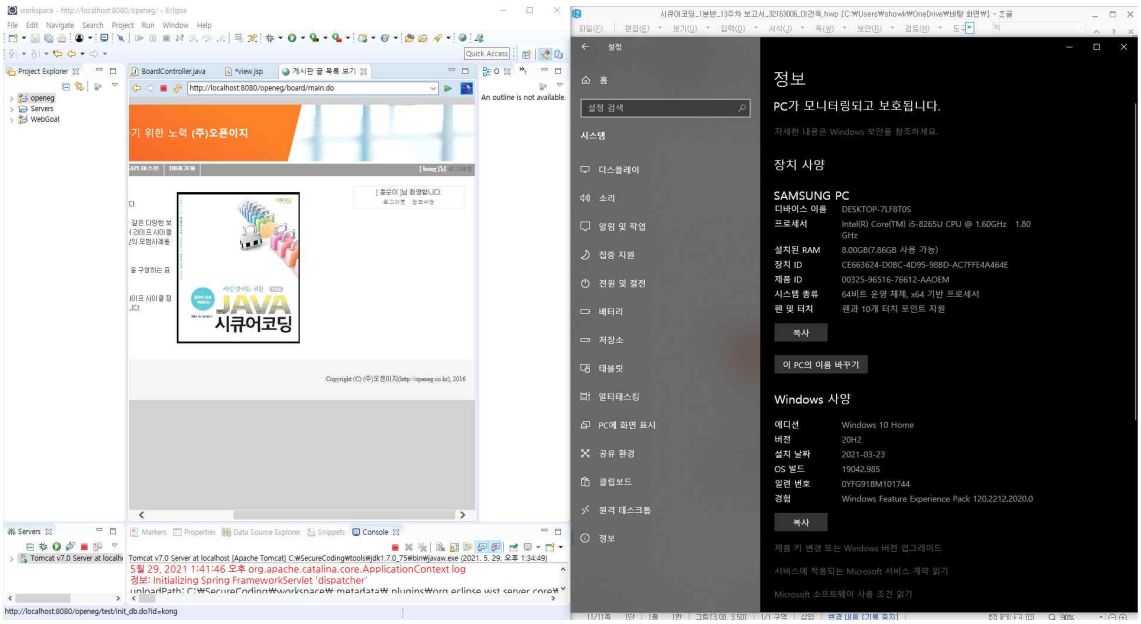


(test 로그인 후 파라미터 조작을 위해 사용자 정보 조회 후 파라미터 변경)



(trap 해제 후 continue시)





– 파라미터 조작 공격 및 방어 실습

(코드 수정)

TestController.java

```
// 새로운 고객 정보 등록
// added by secure_coding
}else if ( "edit".equals(action) && "admin".equals(session.getAttribute("userId"))) {
```