

□ 단국대학교 > 사이버보안 학과 > 인터넷 보안 > 보고서 템플릿

25년 1학기 인터넷 보안 수업

# 인터넷 뱅킹 모의해킹

: JWT 인증값 변조, DB 정보 탈취

2025년 6월 10일

학번 : 32231594

이름 : 박기쁨

## 1. 인터넷 뱅킹(Shield Bank) 10 번 문항 : JWT 인증값 변조

### <과정 설명>

JWT(JSON Web Token)는 정보를 JSON 객체로 안전하게 주고받기 위한 개방형 표준(RFC 7519)이다. 토큰은 비공개 시크릿 키 또는 공개/비공개 키를 사용하여 서명된다.

본 실습에서는, JWT 를 변조하여 타 사용자로 위장한 뒤, 원하는 계좌로 간편 이체를 수행하는 공격을 진행한다.

### Step 1. 페이지 동작 방식 확인

#### Step 1-1. 페이지 동작 방식 확인

: 간편 이체 과정을 따라가며 페이지 동작 방식을 확인한다.

문제 10번. 간편이체에서 수행되는 JWT 서명키가 노출되었습니다. 타사용자로 위장하여 자신의 계좌로 간편이체를 수행하고 거래 내역 조회에서 FLAG를 확인하세요.

Hint 타사용자의 계좌번호는 "110-22102-46358". ID는 "jwtproblem"이고 노출된 JWT 서명키는 "jwtauthkey"입니다.

#### 간편 이체

출금 계좌번호

계좌를 선택하세요.

입금 계좌번호

자주 사용하는 계좌에 등록된 계좌만 표시됩니다.

입금 금액

1000만원 미만 이체 가능

PIN

이체하기

입금 은행

Shield Bank

: 출금 계좌로는 본인의 계좌만 선택할 수 있다.

#### 간편 이체

출금 계좌번호

계좌를 선택하세요.

계좌를 선택하세요.

21354 - 001 - 0479001

자주 사용하는 계좌에 등록된 계좌만 표시됩니다.

입금 금액

1000만원 미만 이체 가능

PIN

이체하기

입금 은행

Shield Bank

: 입금 계좌로는 '자주 사용하는 계좌'에 등록된 계좌만 선택할 수 있다.

간편 이체

출금 계좌번호

21354 - 001 - 0479001

입금 계좌번호

자주 사용하는 계좌에 등록된 계좌만 표시됩니다.

자주 사용하는 계좌에 등록된 계좌만 표시됩니다.

주거래계좌 (21354 - 001 - 0479001)

1000만원 미만 이체 가능

입금 은행

Shield Bank

PIN

이체하기

: 입금 금액과 PIN 을 입력한 후 이체하기 버튼을 누른다.

간편 이체

출금 계좌번호

21354 - 001 - 0479001

입금 계좌번호

주거래계좌 (21354 - 001 - 0479001)

입금 은행

Shield Bank

입금 금액

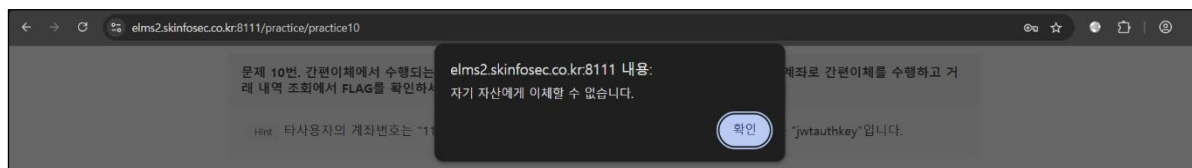
100

PIN

\*\*\*\*\*

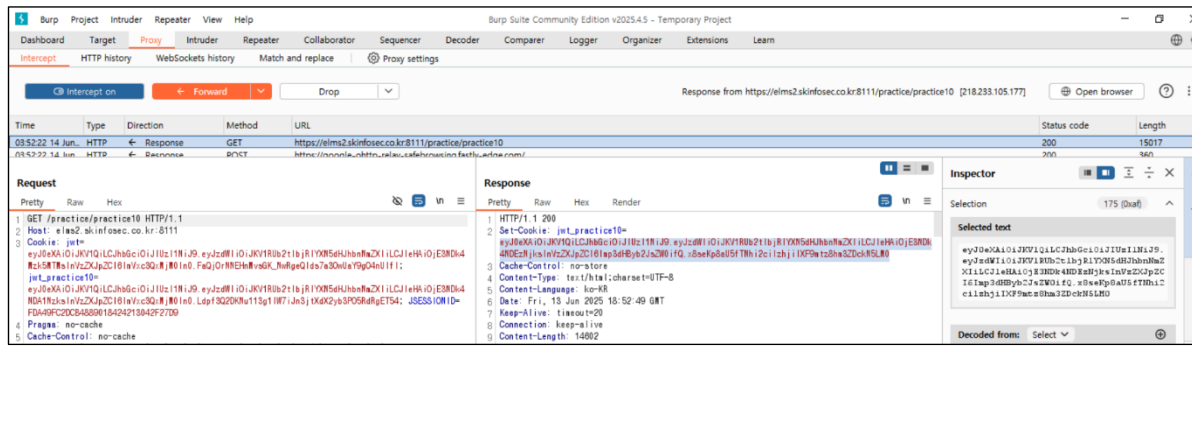
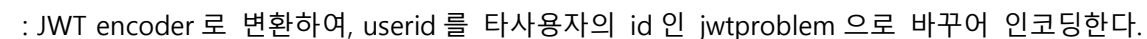
이체하기

: 자기 자신에게 이체할 수 없다는 안내문이 뜬다.





: 서명키란에 노출된 서명키인 jwtauthkey를 입력해보면 서명이 확인된다는 문구가 뜬다.



: 타사용자의 JWT 로 인증한 페이지로 돌아온다.

간편 이체

출금 계좌번호

계좌를 선택하세요.

입금 계좌번호

자주 사용하는 계좌에 등록된 계좌만 표시됩니다.

입금 은행

Shield Bank

입금 금액

1000만원 미만 이체 가능

PIN

이체하기

### Step 3. 출금 계좌 변조하기

#### Step 3-1. 개발자도구 확인하기

: 개발자도구의 Elements 탭을 열어 출금 계좌번호 란과 관련된 코드를 찾는다.

문제 10번. 간편이체에서 수행되는 JWT 서명키가 노출되었습니다. 타사용자로 위장하여 자신의 계좌로 간편이체를 수행하고 거래 내역 조회에서 FLAG를 확인하세요.

Hint 타사용자의 계좌번호는 "110-22102-46358". ID는 "jwtproblem"이고 노출된 JWT 서명키는 "jwtauthkey"입니다.

select#outaccount 435.2 x 40

Color #363636

Font 16px BlinkMacSystemFont, -apple-system...

Background #FFFFFF

Padding 7px 40px 7px 11px

ACCESSIBILITY

Name

Role

Keyboard-focusable

combobox

계좌를 선택하세요.

입금 계좌번호

자주 사용하는 계좌에 등록된 계좌만 표시됩니다.

입금 은행

Shield Bank

입금 금액

1000만원 미만 이체 가능

PIN

이체하기

DevTools is now available in Korean

Don't show again Always match Chrome's language Switch DevTools to Korean

Elements Console Sources Network Performance >> 1

<div class="box">

<div class="columns"> (new)

<div class="column is-6">

<div class="field">

<label class="label">출금 계좌번호</label>

<div class="control">

<div class="select is-fullwidth">

<select id="outaccount">...</select>

<div>

</div>

</div>

</div>

html body

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Y filter

element.style {

}

body {

color: #4a4a4a;

font-size: 1em;

font-weight: 400;

line-height: 1.5;

}

Console AI assistance What's new X Coverage

What's new in DevTools 137

See all new features

new See the highlights from Chrome 137

### Step 3-2. 계좌번호 변경하기

: 출금 계좌번호의 value에 공격자의 계좌번호가 저장되어 있는 것을 확인할 수 있다.

문제 10번. 간편이체에서 수행되는 JWT 서명키가 노출되었습니다. 타사용자로 위장하여 자신의 계좌로 간편이체를 수행하고 거래 내역 조회에서 FLAG를 확인하세요.

Hint 타사용자의 계좌번호는 "110-22102-46358". ID는 "jwtproblem"이고 노출된 JWT 서명키는 "jwtauthkey"입니다.

### 간편 이체

출금 계좌번호

계좌를 선택하세요.

입금 계좌번호

자주 사용하는 계좌에 등록된 계좌만 표시됩니다.

입금 은행

Shield Bank

입금 금액

1000만원 미만 이체 가능

PIN

이체하기

DevTools is now available in Korean

Don't show again Always match Chrome's language Switch DevTools to Korean

Elements Console Sources Network Performance

div class="column is-6" field label="출금 계좌번호" control select is-fullwidth id="outaccount" value="110-22102-46358" selected

What's new in DevTools 137

See all new features

: value 값을 타사용자의 계좌번호로 바꿔치기 한다.

문제 10번. 간편이체에서 수행되는 JWT 서명키가 노출되었습니다. 타사용자로 위장하여 자신의 계좌로 간편이체를 수행하고 거래 내역 조회에서 FLAG를 확인하세요.

Hint 타사용자의 계좌번호는 "110-22102-46358". ID는 "jwtproblem"이고 노출된 JWT 서명키는 "jwtauthkey"입니다.

**간편 이체**

출금 계좌번호

계좌를 선택하세요.

입금 계좌번호

자주 사용하는 계좌에 등록된 계좌만 표시됩니다.

입금 은행

Shield Bank

입금 금액

1000만원 미만 이체 가능

PIN

이체하기

DevTools is now available in Korean

Don't show again Always match Chrome's language Switch DevTools to Korean

Elements Console Sources Network Performance

```

<div class="column is-6">
  <div class="field">
    <label class="label">출금 계좌번호</label>
    <div class="control">
      <div class="select is-fullwidth">
        <select id="outaccount">
          <option value="0">계좌를 선택하세요 </option>
          <option value="1102210246358">21594 - 001 - 047000</option>
          <option value="00">
        </select>
      </div>
    </div>
  </div>

```

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Y Filter

element style {

```

{
  font-family: "Lato", sans-serif;
}
:after, :before {
  box-sizing: inherit;
}

```

Console AI assistance What's new X Coverage

What's new in DevTools 137

See all new features

new See the highlights from Chrome 137

### Step 3-3. 이체 절차 마무리하기

: (value 로 타사용자의 계좌번호를 저장하게 된) 출금 계좌번호, 입금 계좌번호, 입금 금액과 PIN 을 입력한 후 이체하기 버튼을 눌러 이체 절차를 마무리한다.

문제 10번. 간편이체에서 수행되는 JWT 서명키가 노출되었습니다. 타사용자로 위장하여 자신의 계좌로 간편이체를 수행하고 거래 내역 조회에서 FLAG를 확인하세요.

Hint 타사용자의 계좌번호는 "110-22102-46358", ID는 "jwtproblem"이고 노출된 JWT 서명키는 "jwtauthkey"입니다.

#### 간편 이체

출금 계좌번호  
21354 - 001 - 0479001

입금 계좌번호  
주거래계좌 (21354 - 001 - 0479001)

입금 금액  
9999999

PIN  
\*\*\*\*\*

입금 은행  
Shield Bank

이체하기

DevTools is now available in Korean

Don't show again Always match Chrome's language Switch DevTools to Korean

Elements Console Sources Network

1 6

Elements

```
<div class="columns"> (7x)
  <div class="column is-6">
    <div class="field">
      <label class="label">출금 계좌번호</label>
      <div class="control">
        <div class="select is-fullwidth">
          <select id="outaccount">
            <option value="0">계좌를 선택하세요.</option> (x 1x)
            <option value="1102210246358">21354 - 001 - 0479001</option> (x 2x)
          </select>
        </div>
      </div>
    </div>
  </div>
  <div class="column is-6">
    <div class="field">
      <label class="label">입금 계좌번호</label>
      <div class="control">
        <div class="select is-fullwidth">
          <select id="inaccount">
            <option value="0">계좌를 선택하세요.</option> (x 1x)
            <option value="1102210246358">21354 - 001 - 0479001</option> (x 2x)
          </select>
        </div>
      </div>
    </div>
  </div>
  <div class="column is-6">
    <div class="field">
      <label class="label">입금 금액</label>
      <div class="control">
        <input type="text" value="9999999">
      </div>
    </div>
  </div>
  <div class="column is-6">
    <div class="field">
      <label class="label">PIN</label>
      <div class="control">
        <input type="password" value="*****">
      </div>
    </div>
  </div>
  <div class="column is-6">
    <div class="field">
      <label class="label">입금 은행</label>
      <div class="control">
        <div class="select is-fullwidth">
          <select id="bank">
            <option value="0">은행을 선택하세요.</option> (x 1x)
            <option value="1">Shield Bank</option> (x 2x)
          </select>
        </div>
      </div>
    </div>
  </div>
  <div class="column is-6">
    <div class="field">
      <label class="label">이체하기</label>
      <div class="control">
        <button type="button">이체하기</button>
      </div>
    </div>
  </div>
</div>
```

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

Filter

element.style { }

font-family: "Lato", sans-serif; base.css:1

...after, :before { box-sizing: inherit; bulma.min.css:1

Console AI assistance What's new X Coverage

What's new in DevTools 137

See all new features

See the highlights from Chrome 137

: 간편 이체가 문제 없이 이루어진다.

#### 완료

간편 이체가 완료되었습니다!

: 거래 내역 조회 페이지를 확인하여 정답을 획득한다. (정답: JWTVULNERABILITY)

#### 거래 내역 조회

계좌번호  
21354 - 001 - 0479001

시작 날짜  
연도-월-일

종료 날짜  
연도-월-일

거래 유형  
전체

검색

거래일시	보낸분/받는분	금액	잔액	거래 유형
2025-06-14	FLAG: JWTVULNERABILITY	9,999,999원	4,820,999,998원	입금
2025-06-10	FLAG: JWTVULNERABILITY	9,999,999원	4,810,999,999원	입금

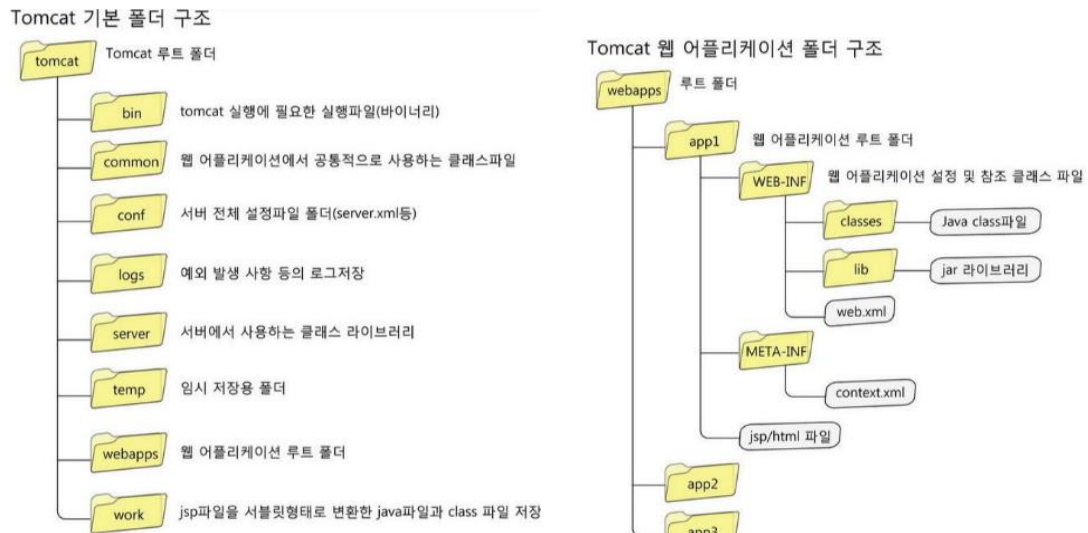
2025년 6월 14일 토요일

## 2. 인터넷 뱅킹(Shield Bank) 9 번 문항 : DB 정보 탈취

### <과정 설명>

본 실습에서는 고의적으로 500 에러를 발생시켜 서버 구조를 확인하고, 필요한 정보를 획득하여 DB 정보를 탈취하고자 한다.

Tomcat 의 폴더 구조를 숙지하면 보다 수월하게 필요한 정보를 획득할 수 있다.



(사진 출처: <https://sallykim5087.tistory.com/130>)

### Step 1. 페이지 동작 방식 확인하기

#### Step 1-1. 페이지 동작 방식 확인하기

: 문제 사이트로 이동 버튼을 눌러본다.

← → elms2.skinfosec.co.kr:8111/practice/practice9 ☆ 🔍 🔄 🌐

문제 9번. WEB 서버와 WAS가 DocumentRoot를 공유하여 구성되어있습니다. 내부 서버 에러(500)를 발생시켜 DocumentRoot의 구조를 유추하고 DB정보를 탈취하여 DB 패스워드를 알아내세요.

Hint Apache - Tomcat - Spring

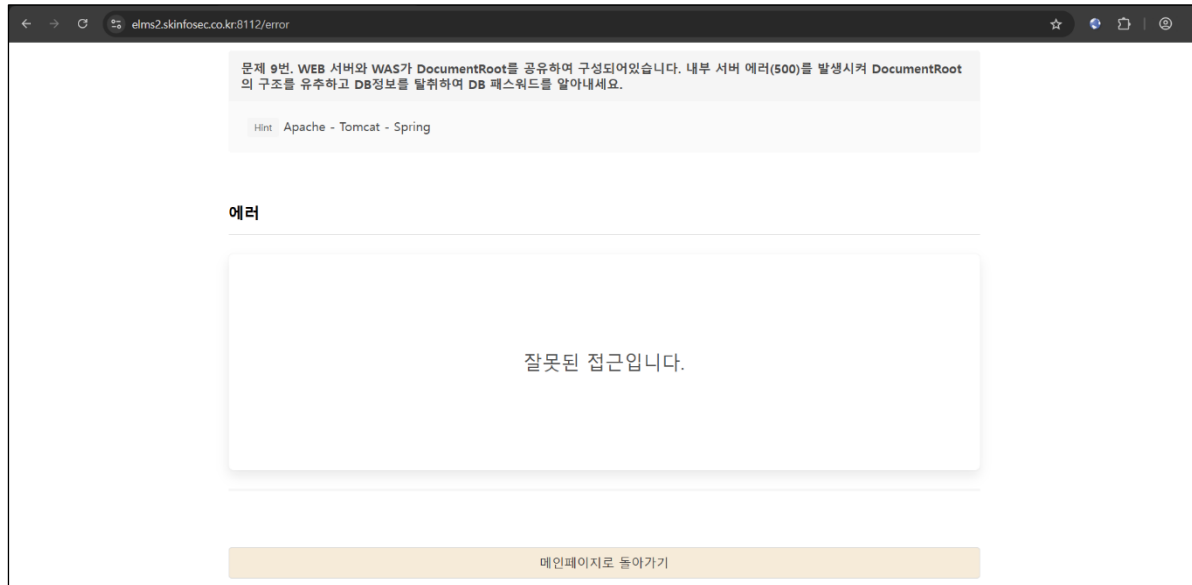
이동

문제 사이트로 이동하세요.

문제 사이트로 이동

: 잘못된 접근이라는 에러 페이지로 이어진다.

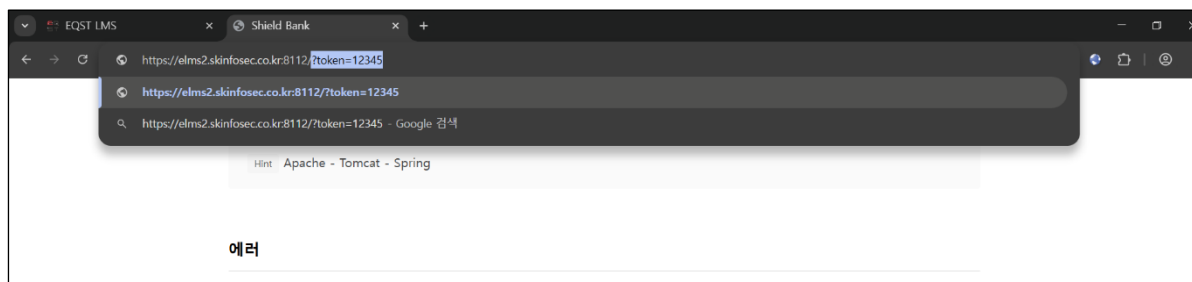
: 처음 페이지의 주소는 :8111 이었지만, 이동한 에러 페이지는 :8112 의 주소를 갖는다.



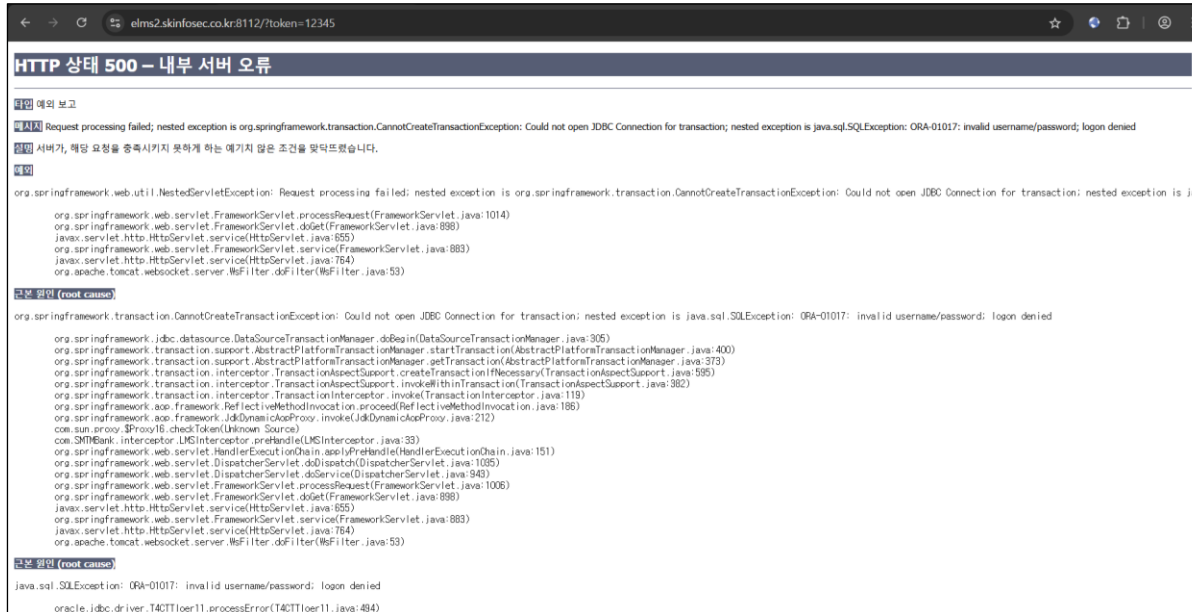
## Step 2. 500 에러를 이용해 구조 유추하기

### Step 2-1. 500 에러 발생시키기

: 에러 페이지의 URL 뒤에 token 파라미터를 추가하고, 파라미터 값으로는 아무 글자나 입력한 후 이동하여 500 에러를 고의적으로 발생시킨다.



: 500 에러가 발생하며 서버에 대한 정보가 노출된다.



: 페이지의 하단의 정보를 통해 실습 서버가 Apache Tomcat/9.0.56 임을 확인할 수 있다.

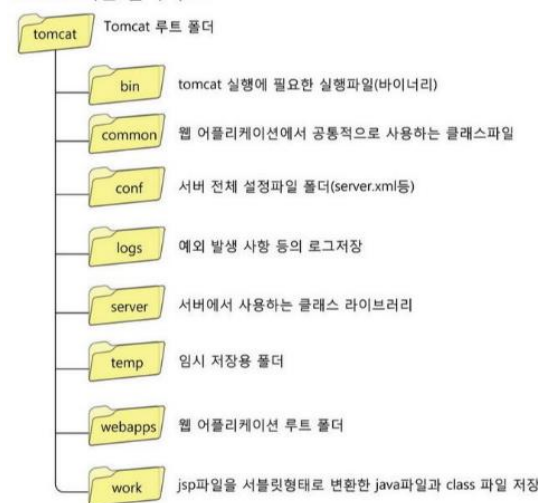


### Step 3. 서버 정보 탈취하기

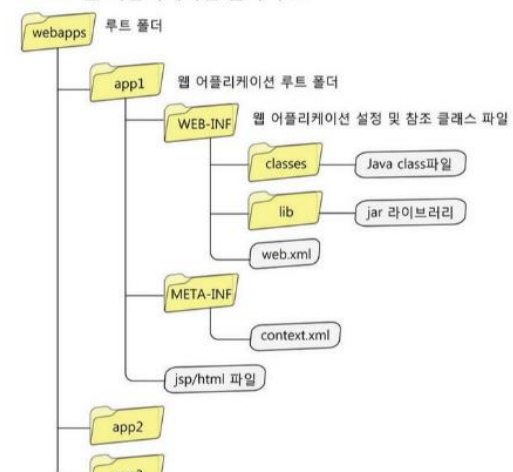
#### Step 3-1. web.xml 확인하기

: 정보를 획득하기 위해, 웹 어플리케이션의 설정파일인 /WEB-INF/web.xml 로 이동해본다.

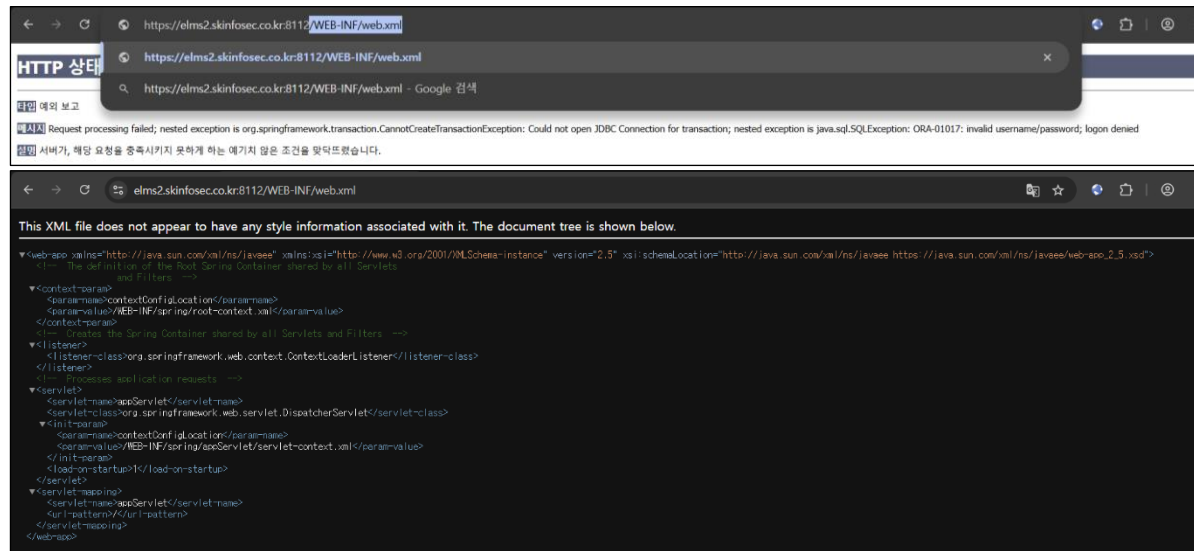
#### Tomcat 기본 폴더 구조



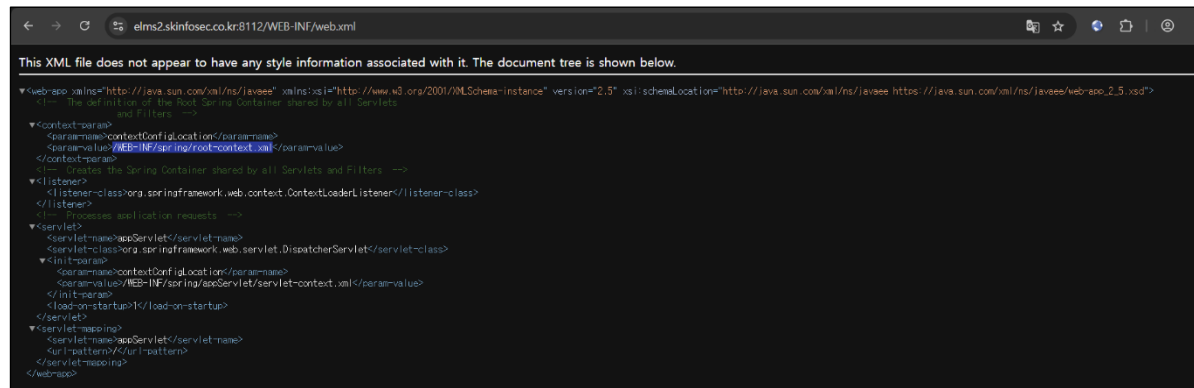
#### Tomcat 웹 어플리케이션 폴더 구조



(사진 출처: <https://sallykim5087.tistory.com/130>)

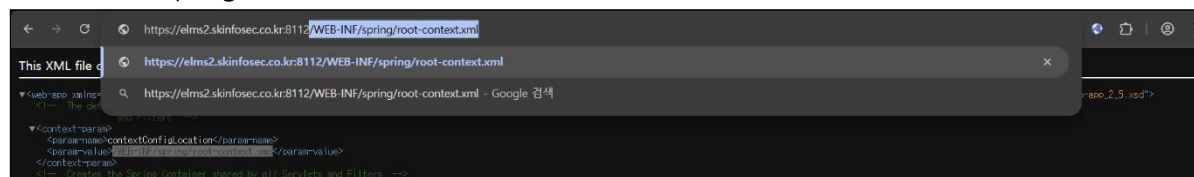


: /WEB-INF/spring/root-context.xml 라는 이름의 파일을 발견할 수 있다.



### Step 3-2. root-context.xml 확인하기

: /WEB-INF/spring/root-context.xml 로 이동해본다.



```
← → ↻ elms2.skinfosec.co.kr:8112/WEB-INF/spring/root-context.xml
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx" xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">
    <!-- Root Context: defines shared resources visible to all other web components -->
    <bean id="environmentVariablesConfiguration" class="org.springframework.javascript.rhino.config.EnvironmentStringPEDEncryptor">
        <property name="algorithm" value="PEWTHMD5xDES"/>
        <property name="password" value="hellomyfriend"/>
    </bean>
    <bean id="configurationEncryptor" class="org.springframework.javascript.rhino.config.StandardPEDEncryptor">
        <property name="config" ref="environmentVariablesConfiguration"/>
    </bean>
    <bean id="propertyConfigurer" class="org.springframework.javascript.rhino.config.EncryptablePropertyPlaceholderConfigurer">
        <constructor-arg ref="configurationEncryptor"/>
    </bean>
    <property name="locations">
        <list>
            <value>classpath:/property/dbinfo.properties</value>
        </list>
    </property>
    </bean>
    <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="${db.driverClassName}" />
        <property name="url" value="${db.url}" />
        <!-- <property name="username" value="${db.username}" />
        <!-- <property name="password" value="${db.password}" />
        </bean>
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <constructor-arg ref="dataSource"/>
    </bean>
    <!-- enable transaction demarcation with annotations -->
    <tx:annotation-driven/>
    <!-- simplest possible SqlSessionFactory configuration -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="mapperLocations" value="classpath:/com/SMTBank/mapper/*.xml"/>
    </bean>
    <!-- Mapper config -->
    <bean id="smtBankMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface" value="com.SMTBank.mapper.SMTBankMapper"/>
        <property name="sqlSessionFactory" ref="sqlSessionFactory"/>
    </bean>
    <!-- Service config -->
    <bean id="smtBankService" class="com.SMTBank.service.SMTBankServiceImpl">
        <constructor-arg ref="smtBankMapper"/>
    </bean>
</beans>
```

: jasypt 관련 password 값이 hellomyfriend 임을 확인할 수 있다.

```
← → ↻ elms2.skinfosec.co.kr:8112/WEB-INF/spring/root-context.xml
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx" xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">
    <!-- Root Context: defines shared resources visible to all other web components -->
    <bean id="environmentVariablesConfiguration" class="org.springframework.javascript.rhino.config.EnvironmentStringPEDEncryptor">
        <property name="algorithm" value="PEWTHMD5xDES"/>
        <property name="password" value="hellomyfriend"/>
    </bean>
    <bean id="configurationEncryptor" class="org.springframework.javascript.rhino.config.StandardPEDEncryptor">
        <property name="config" ref="environmentVariablesConfiguration"/>
    </bean>
    <bean id="propertyConfigurer" class="org.springframework.javascript.rhino.config.EncryptablePropertyPlaceholderConfigurer">
        <constructor-arg ref="configurationEncryptor"/>
    </bean>
    <property name="locations">
        <list>
            <value>classpath:/property/dbinfo.properties</value>
        </list>
    </property>
    </bean>
    <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="${db.driverClassName}" />
        <property name="url" value="${db.url}" />
        <!-- <property name="username" value="${db.username}" />
        <!-- <property name="password" value="${db.password}" />
        </bean>
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <constructor-arg ref="dataSource"/>
    </bean>
    <!-- enable transaction demarcation with annotations -->
    <tx:annotation-driven/>
    <!-- simplest possible SqlSessionFactory configuration -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="mapperLocations" value="classpath:/com/SMTBank/mapper/*.xml"/>
    </bean>
    <!-- Mapper config -->
    <bean id="smtBankMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface" value="com.SMTBank.mapper.SMTBankMapper"/>
        <property name="sqlSessionFactory" ref="sqlSessionFactory"/>
    </bean>
    <!-- Service config -->
    <bean id="smtBankService" class="com.SMTBank.service.SMTBankServiceImpl">
        <constructor-arg ref="smtBankMapper"/>
    </bean>
</beans>
```

: locations 부분을 통해 dbinfo.properties 라는 이름의 파일의 존재를 확인할 수 있다.

: classpath 가 적혀 있지는 않지만, JAVA class 파일은 WEB-INF 아래 classes 폴더에 위치한다.  
(Step 3-1 의 사진 참고)

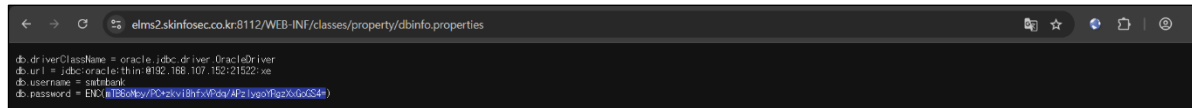
```
← → ↻ elms2.skinfosec.co.kr:8112/WEB-INF/spring/root-context.xml
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx" xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.3.xsd">
    <!-- Root Context: defines shared resources visible to all other web components -->
    <bean id="environmentVariablesConfiguration" class="org.springframework.javascript.rhino.config.EnvironmentStringPEDEncryptor">
        <property name="algorithm" value="PEWTHMD5xDES"/>
        <property name="password" value="hellomyfriend"/>
    </bean>
    <bean id="configurationEncryptor" class="org.springframework.javascript.rhino.config.StandardPEDEncryptor">
        <property name="config" ref="environmentVariablesConfiguration"/>
    </bean>
    <bean id="propertyConfigurer" class="org.springframework.javascript.rhino.config.EncryptablePropertyPlaceholderConfigurer">
        <constructor-arg ref="configurationEncryptor"/>
    </bean>
    <property name="locations">
        <list>
            <value>classpath:/property/dbinfo.properties</value>
        </list>
    </property>
    </bean>
    <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="${db.driverClassName}" />
        <property name="url" value="${db.url}" />
        <!-- <property name="username" value="${db.username}" />
        <!-- <property name="password" value="${db.password}" />
        </bean>
    <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <constructor-arg ref="dataSource"/>
    </bean>
    <!-- enable transaction demarcation with annotations -->
    <tx:annotation-driven/>
    <!-- simplest possible SqlSessionFactory configuration -->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="mapperLocations" value="classpath:/com/SMTBank/mapper/*.xml"/>
    </bean>
    <!-- Mapper config -->
    <bean id="smtBankMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface" value="com.SMTBank.mapper.SMTBankMapper"/>
        <property name="sqlSessionFactory" ref="sqlSessionFactory"/>
    </bean>
    <!-- Service config -->
    <bean id="smtBankService" class="com.SMTBank.service.SMTBankServiceImpl">
        <constructor-arg ref="smtBankMapper"/>
    </bean>
</beans>
```

### Step 3-3. dbinfo.properties 확인하기

: /WEB-INF/property/dbinfo.properties 로 이동해본다.



: DB 패스워드가 암호화된 채로 저장되어 있다.



### Step 4. DB 패스워드 복호화

#### Step 4-1. Jasypt Decrypt 하기

: Jasypt Decrypt 사이트에 암호화된 DB 패스워드를 붙여 넣는다.

#### Jasypt Encryption

Enter Plain Text to Encrypt

Select Type of Encryption

One Way Encryption (Without Secret Text)

Enter Secret Key

Encrypt

Jasypt Encrypted String

Result goes here

#### Jasypt Decryption

Enter Jasypt Encrypted Text

Select Action Type

Match Password

Enter the Plain Text to Match

Secret Key Used during Encryption

Match/Decrypt

Result:

Result goes here

#### Online File Encrypt Decrypt

Online Text Encrypt Decrypt

ai 개발 7개월간 제대로 배우자  
코드잇 부트캠프 전액 국비지원  
온라인 100%로 전국 어디서나 합류 가능.  
오프라인 네트워킹과 멘토링도 탄탄하게  
채워드려요.  
코드잇 스프린트

알기 >

: Action Type 을 Decrypt 로 설정해준다.

Jasypt Encryption

Enter Plain Text to Encrypt

Enter plain text to hash

Select Type of Encryption

One Way Encryption (Without Secret Text)

Enter Secret Key

Enter Secret Key

Encrypt

Jasypt Encrypted String

Result goes here

Jasypt Decryption

Enter Jasypt Encrypted Text

mTB6oMpy/PC+zkvI8hfxVPdq/APzlygoYRgzXxGoGS4=

Select Action Type

Match Password

Decrypt Password

Match Password

Secret Key Used during Encryption

Enter Secret Key

Match/Decrypt

Result:

Result goes here

Online File Encrypt Decrypt

Online Text Encrypt Decrypt

ai 개발 7개월간 제대로 배우자

코드잇 부트캠프 전액 국비지원

온라인 100%로 전국 어디서나 합류 가능.

오프라인 네트워킹과 멘토링도 탄탄하게 채워드려요.

코드잇 스프린트

열기

: Secret Key 란에는 root-context.xml 에서 획득했던 hellomyfriend 패스워드를 입력한다.

Jasypt Encryption

Enter Plain Text to Encrypt

Enter plain text to hash

Select Type of Encryption

One Way Encryption (Without Secret Text)

Enter Secret Key

Enter Secret Key

Encrypt

Jasypt Encrypted String

Result goes here

Jasypt Decryption

Enter Jasypt Encrypted Text

mTB6oMpy/PC+zkvI8hfxVPdq/APzlygoYRgzXxGoGS4=

Select Action Type

Decrypt Password

Enter the Plain Text to Match

Enter Plain text to match

Secret Key Used during Encryption

hellomyfriend

Match/Decrypt

Result:

Result goes here

Online File Encrypt Decrypt

Online Text Encrypt Decrypt

ai 개발 7개월간 제대로 배우자

코드잇 부트캠프 전액 국비지원

온라인 100%로 전국 어디서나 합류 가능.

오프라인 네트워킹과 멘토링도 탄탄하게 채워드려요.

코드잇 스프린트

열기

: Decrypt 를 수행하여 정답을 획득한다. (정답: OPERATIONALINFOEXPOSURE)

Jasypt Encryption

Enter Plain Text to Encrypt

Enter plain text to hash

Select Type of Encryption

One Way Encryption (Without Secret Text)

Enter Secret Key

Enter Secret Key

Encrypt

Jasypt Encrypted String

Result goes here

Jasypt Decryption

Enter Jasypt Encrypted Text

mTB6oMpy/PC+zkvI8hfxVPdq/APzlygoYRgzXxGoGS4=

Select Action Type

Decrypt Password

Enter the Plain Text to Match

Enter Plain text to match

Secret Key Used during Encryption

hellomyfriend

Match/Decrypt

Result:

OPERATIONALINFOEXPOSURE

Online File Encrypt Decrypt

Online Text Encrypt Decrypt

ai 개발 7개월간 제대로 배우자

코드잇 부트캠프 전액 국비지원

온라인 100%로 전국 어디서나 합류 가능.

오프라인 네트워킹과 멘토링도 탄탄하게 채워드려요.

코드잇 스프린트

열기

성명	프로젝트 후 소감
박기쁨	<p>보다 고차원적인 웹 해킹을 위해서는 서버의 구조에 대한 이해가 필수적이라는 것을 느낀 실습이었던 것 같다. 정해진 정답대로 따라가며 수행하는 해킹이 아닌, 주도적으로 탐구하고 부딪혀보는 해킹을 위해서는, 해킹 방식 외적인 부분들, 해킹하려는 대상 자체에 대한 공부가 필요할 것 같다.</p>