

Multiple vulnerabilities in Apache Foundation Struts 2 framework

Csaba Barta and László Tóth

12. June 2008

Content

| | |
|---|-------------------|
| Content..... | 2 |
| Summary..... | 3 |
| Directory traversal vulnerability in static content serving..... | 3 |
| Double URL decoding vulnerability in findInputStream method..... | 4 |
| Circumventing the “.class filter” in findStaticResource method..... | 6 |
| Impact of the vulnerabilities..... | 7 |

Summary

We identified three critical vulnerabilities in the Struts 2 framework. According to struts.apache.org:

“Struts 2 is an extensible framework for creating enterprise-ready Java web applications. The framework is designed to streamline the full development cycle, from building, to deploying, to maintaining applications over time. (<http://struts.apache.org/2.x/>)”

Directory traversal vulnerability in static content serving

Remote: Yes

Risk: High

Vulnerable systems: Struts 2.0.5 and above were tested including 2.1.2 beta version

Immune Systems: No other version of Struts was tested

If the requested URI starts with the “/struts” string, the `doFilter` method of the `FilterDispatcher` class calls the `findStaticResource` method, which serves the static content, as shown below:

```
if (serveStatic && resourcePath.startsWith("/struts")) {
    String name = resourcePath.substring("/struts".length());
    findStaticResource(name, request, response);
} else {
    // this is a normal request, let it pass through
    chain.doFilter(request, response);
}
...
}
```

The first parameter (“name”) of the `findStaticResource` will contain the URI without the “/struts” string. If the request is the following (testing the struts2 blank example application which is distributed with struts2):

`http://exampletomcat.com:8080/struts2-blank-2.0.11.1/struts..`

The following response is sent by the web server, which is the “lib” folder of the Apache Tomcat:



Figure 1.: The content of the lib directory of Tomcat

We tested some application where the served directory was the parent of the WEB-INF folder and we could browse and download files from it.

Double URL decoding vulnerability in findInputStream method

Remote: Yes

Risk: High

Vulnerable systems: Struts 2.0.5 and above were tested including 2.1.2 beta version.

Immune Systems: No other version of Struts was tested

The `findInputStream` function is responsible for loading the static content from the packages. This function is vulnerable to double URL decoding. The following code snippet shows the vulnerable part in the source code of `FilterDispatcher` class:

```
protected InputStream findInputStream(String name, String packagePrefix) throws
IOException {
...
    resourcePath = URLDecoder.decode(resourcePath, encoding);
    return ClassLoaderUtil.getResourceAsStream(resourcePath, getClass());
}
```

If the attacker supplies double URL encoded special characters in the URL, the first decoding will be done by web server or application server. The `findInputStream` will be supplied with the simple URL encoded version of the character then the call to `URLDecoder.decode` function will result in the special characters decoded. The `ClassLoaderUtil.getResourceAsStream` function will be supplied with this decoded path.

Example:

1. The attacker supplies “%252f” in the URL
2. After the first decoding done by Tomcat the character will be decoded to “%2f”
3. After the call to `URLDecoder.decode` the character will be decoded to “/”

This vulnerability can be exploited by an attacker to circumvent an URL filter or how the web server handles “../”.

Example request:

```
http://exampletomcat.com:8080/struts2-blank-2.0.11.1/struts/..%252f
```

The response sent by the server:



Figure 2.: Double URL decoding

By supplying further “%252f” the attacker can navigate through the search path configured for the application and find files containing sensitive information (e.g.: application configuration, java classes (see the next section)).

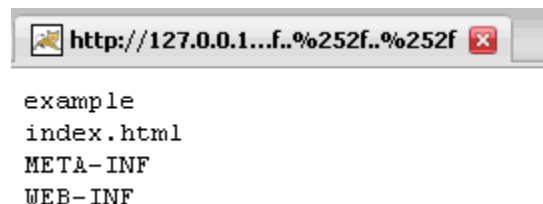


Figure 3.: Further navigating through the search path

Circumventing the “.class filter” in findStaticResource method

Remote: Yes

Risk: High

Vulnerable systems: Struts 2.0.5 and above were tested including 2.1.2 beta version

Immune Systems: No other version of Struts was tested

The `findStaticResource` method implements a “filter” that is responsible for preventing the download of java class files. Here is the code snippet from `findStaticResource`:

```
protected void findStaticResource(String name, HttpServletRequest request,
HttpServletResponse response) throws IOException {
    if (!name.endsWith(".class")) {
        for (String pathPrefix : pathPrefixes) {
            InputStream is = findInputStream(name, pathPrefix);
            ...
        }
    }
    ...
    response.sendError(HttpServletResponse.SC_NOT_FOUND);
}
```

The “filter” inspects the URL, and if the requested content’s name ends with “.class” it denies serving it.

It is possible to circumvent this filter by adding “/” to the end of the file name. For example (testing the struts2 blank example application which is distributed with struts2):

```
http://exampletomcat.com:8080/struts2-blank-2.0.11.1/struts/..%252f
..%252f..%252fWEB-INF/classes/example/Login.class/
```

The response sent by the server:

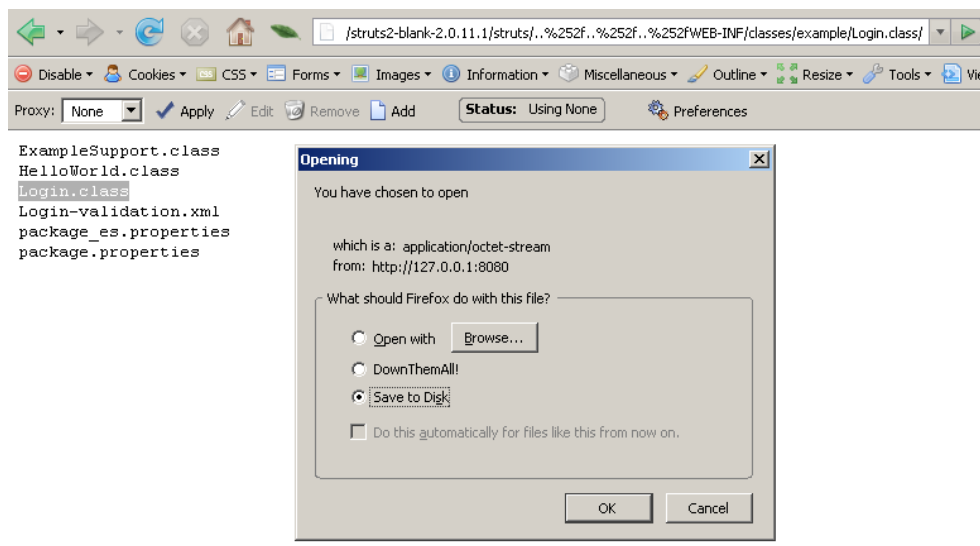


Figure 4.: Downloading java class file

Because of the last “/” after the “.class” the upper mentioned condition will be met (the requested content doesn’t end with “.class” it ends with “.class/”), and `findInputStream` will be called in a loop which will result in the class file served to the client.

This security hole can be exploited by taking advantage of the previous vulnerability.

Impact of the vulnerabilities

A remote attacker may download application configuration files and java classes from the web server running vulnerable version of the framework.