Research & Technique

Spring Security 권한 우회 취약점 (CVE-2022-22978)

■ 취약점 개요

2022년 5월, Spring Security에서 권한을 우회할 수 있는 취약점이 공개되었다. Spring Security는 Spring의 인증, 인가 등 보안을 담당하는 프레임워크이다.

Spring Security 권한 우회(CVE-2022-22978) 취약점은 정규표현식을 이용하여 권한을 부여하는 기능인 RegexRequestMatcher에서 발생한다. 정규표현식에 '.'이 포함된 RegexRequestMatcher 사용 시 개행 문자 [1]에 대한 처리가 누락되어 발생하는 취약점으로, URL에 개행 문자를 추가하는 것으로 권한을 우회할 수 있다. 부여된 권한을 우회하여 관리자 페이지 등 권한이 없는 페이지에 접근이 가능한 만큼 CVSS 9.8점으로 평가되었다.

또한, Spring 프레임워크는 국내 웹 개발 시 사용하는 전자정부 프레임워크의 기반 기술로 사용되고 있다. 많은 개발자에 의해 사용되고 있는 만큼 주의가 필요하며, 취약한 버전의 Spring Security를 사용하고 있다면 업데이트를 고려해야 한다.

■ 영향 받는 소프트웨어 버전

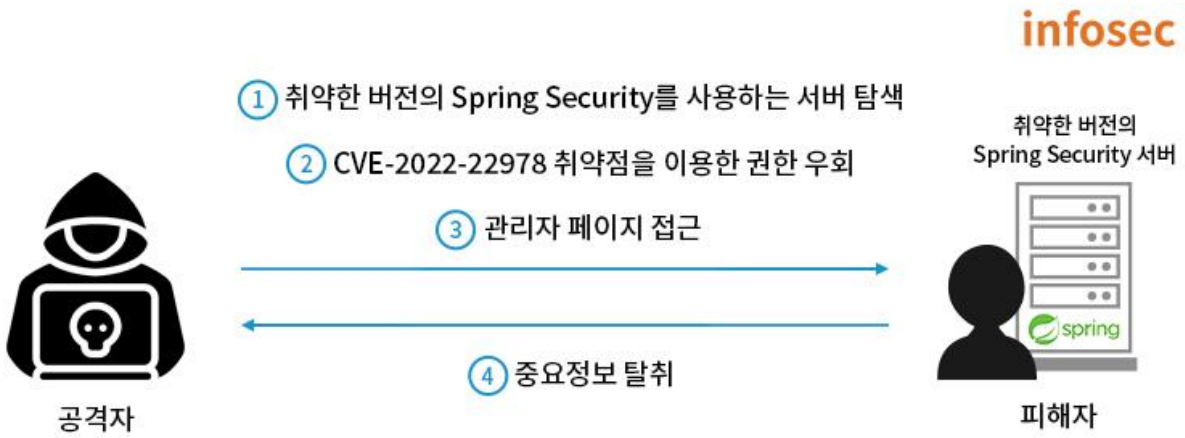
CVE-2022-22978에 취약한 소프트웨어는 다음과 같다.

S/W 구분	취약 버전
Spring Security	5.5.x ~ 5.5.7 이전 버전 5.6.x ~ 5.6.4 이전 버전 다른 하위 버전도 영향을 받음

[1] 개행 문자는 텍스트의 한 줄이 끝났음을 표시하는 문자 또는 문자열이다. CR(Carriage Return, ₩r), LF(Line Feed, ₩n)으로 표현하며, URL 인코딩의 경우 %0d, %0a로 표현한다.

■ 공격 시나리오

CVE-2022-22978 를 이용한 공격 시나리오는 다음과 같다.



[공격 시나리오]

① 공격자는 취약한 버전의 Spring Security를 사용하는 서버 탐색
② 공격자는 CVE-2022-22978 취약점을 이용한 권한 부여 우회
③ 관리자 페이지 접근 등 권한 우회를 통해 서버 접근
④ 중요정보 탈취, 웹쉘 업로드 등 공격 수행

■ 테스트 환경 구성 정보

취약한 버전의 Spring Security를 사용하는 테스트 환경을 구축하여 CVE-2022-22978의 동작 과정을 살펴본다.

이름	정보
피해자	Spring Security 5.6.2

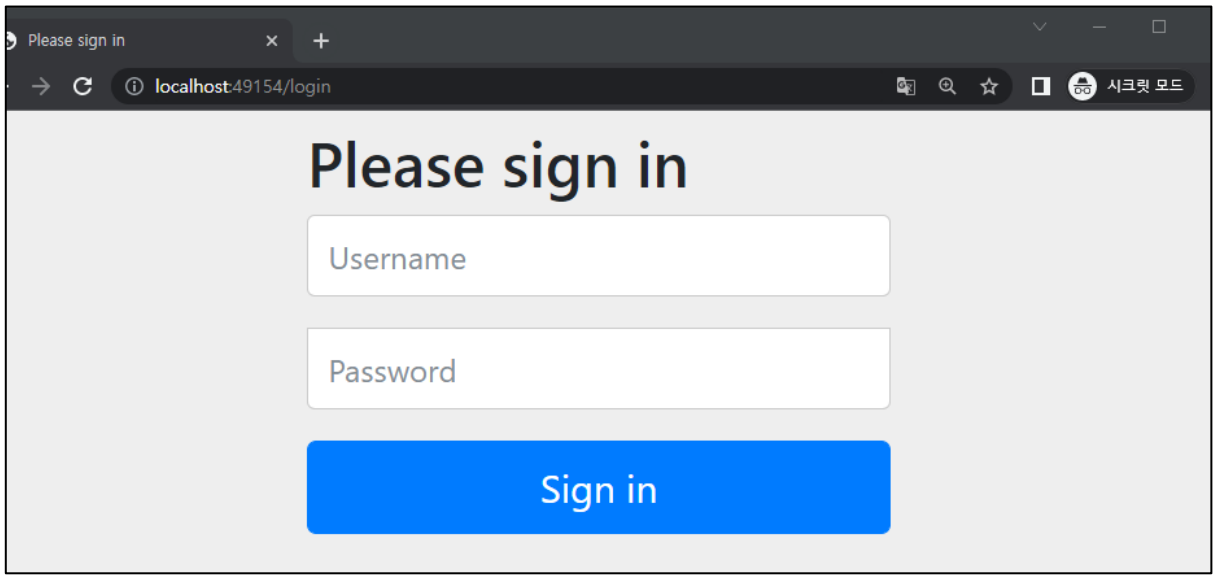
■ 취약점 테스트

Step 1. PoC 테스트

테스트를 위한 PoC가 저장된 github URL은 다음과 같다.

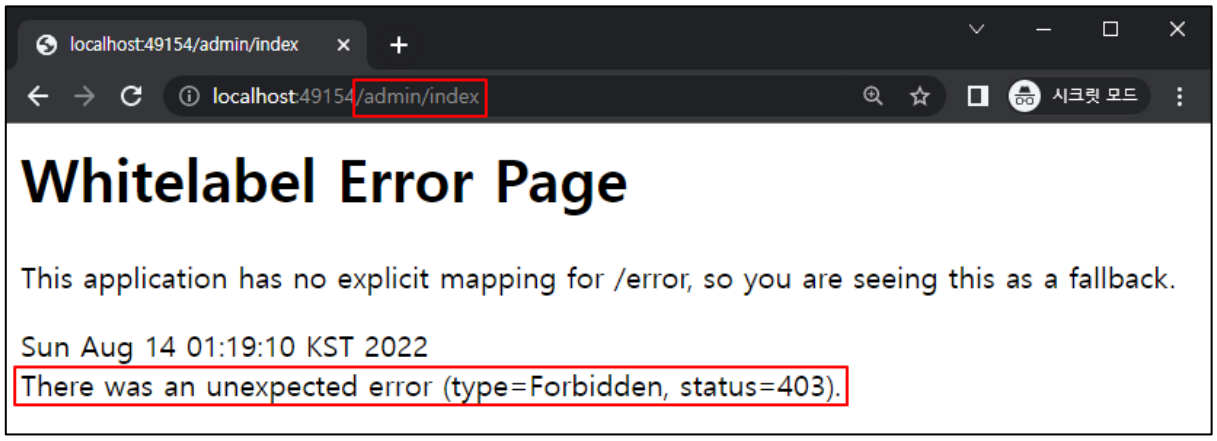
URL : https://github.com/aeifkz/CVE-2022-22978

step 1) 웹 브라우저를 통해 생성된 웹 사이트에 접근한다. 로그인 화면을 확인할 수 있다.



[웹 사이트 접근]

step 2) 로그인을 하지 않고 URL을 통해 관리자 페이지(/admin/index)로 접근 시, 403 Forbidden 에러를 반환한다.



[관리자 페이지 접근 시 에러 발생]

step 3) URL에 개행 문자를 추가하여 관리자 페이지로 접근을 시도한다. 권한 우회에 성공하여 관리자 페이지에 접근한 것을 확인할 수 있다.

[관리자 페이지 접근 성공]

■ 취약점 상세 분석

Step 1. PoC 분석

CVE-2022-22978의 PoC를 통해 생성한 웹 사이트는 다음과 같이 구성되어 있다.

1) pom.xml 파일

프로젝트 빌드 옵션이 설정된 pom.xml 파일 확인 결과, CVE-2022-22978 취약점이 존재하는 버전의 Spring Security를 사용 중인 것을 볼 수 있다.

```
<description>CVE-2022-22978</description>
<properties>
    <java.version>1.8</java.version>
    <spring-security.version>5.6.2</spring-security.version>
</properties>
```

[취약한 Spring Security 버전]

2) Spring Controller 설정

웹 사이트 사용자가 관리자 권한만이 접근할 수 있는 경로인 '/admin/*' 으로 접근 시, 'This is manage page' 라는 문구를 반환하는 것을 확인할 수 있다.

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

/*@RestController*/
@Controller
public class Demo {
    @GetMapping("/admin/*")
    public String Manage(){
        /*return "Manage page";*/
        return "This is manage page";
    }
```

[Spring Controller 파일]

3) URL 접근 권한 설정

Spring Security 는 인가 처리를 위해 사용자가 접근할 수 있는 리소스를 제어하는 기능을 제공한다. 접근 정책을 통해 리소스에 대한 접근 허용 여부를 결정하는데, 이때 ant 형식 [2]과 정규표현식 등으로 경로를 지정할 수 있다. CVE-2022-22978 의 경우 정규표현식을 통해 발생한 취약점으로 URL 에 대한 접근 권한을 설정한 HttpSecurity 의 내용은 다음과 같다.

```
package cc.saferoad.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurit
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfi

@Configuration
@EnableWebSecurity
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception{
        httpSecurity.authorizeRequests().regexMatchers("/admin/.*").authenticated();
    }
}
```

[Spring Security 설정 파일]

① httpSecurity.authorizeRequests( ): 인증을 통한 URL 접근 권한 제어
② regexMatchers("/admin/.*"): 정규표현식으로 지정한 경로(/admin/.*)
③ authenticated( ): 인증을 통한 접근 제어

사용자가 URL 을 통해 정규표현식으로 지정한 경로(/admin/.*)에 접근 시, 인증을 요구하도록 설정되어 있다.

[2] ant형식은 Apache Ant Style로 ?, *, **와 같은 문자를 사용하여 패턴을 매핑한다.

정규표현식으로 지정한 '/admin/.*'의 의미는 다음과 같다.

정규표현식	의미
.	임의의 한 문자
*	0 개 이상의 모든 문자

/admin/ 아래의 경로 접근 시 인증을 요구하는데, 정규표현식 .*로 인해 모든 문자에 대한 인증을 요구한다. 하지만, CVE-2022-22978 취약점이 존재하는 경우 URL 경로에 개행 문자를 의미하는 %0d, %0a 를 추가하면 필터에서 제외되어 권한 없이 관리자 페이지에 접근이 가능하다.

Step 2. 취약점 동작 과정

Spring Security 는 여러 개의 Filter 를 통해 인증 및 인가를 진행한다. CVE-2022-22978 의 경우, 정규표현식을 이용한 Filter 에서 발생한 취약점이며 동작 과정은 다음과 같다.

step 1) RegexRequestMatcher.java

정규표현식을 처리하는 RegexRequestMatcher 클래스의 옵션이 DEFAULT 모드로 지정된 것을 볼 수 있다. DEFAULT 모드는 정규표현식 처리 시, 개행 문자를 포함하지 않는다. 따라서 URL 에 개행 문자를 입력 시 패턴에서 제외된다.

```
public RegexRequestMatcher(String pattern, String httpMethod, boolean caseInsensitive) {
    this.pattern = Pattern.compile(pattern, caseInsensitive ? Pattern.CASE_INSENSITIVE : DEFAULT);
    this.httpMethod = StringUtils.hasText(httpMethod) ? HttpMethod.valueOf(httpMethod) : null;
}
```

[RegexRequestMatcher 클래스 설정(1)]

또한, 사용자로부터 입력받은 URL을 pattern.matcher(url)을 통해 패턴을 검증한다.

```
public boolean matches(HttpServletRequest request) {
    if (this.httpMethod != null && request.getMethod() != null
            && this.httpMethod != HttpMethod.resolve(request.getMethod())) {
        return false;
    }
    String url = request.getServletPath();
    String pathInfo = request.getPathInfo();
    String query = request.getQueryString();
    if (pathInfo != null || query != null) {
        StringBuilder sb = new StringBuilder(url);
        if (pathInfo != null) {
            sb.append(pathInfo);
        }
        if (query != null) {
            sb.append('?').append(query);
        }
        url = sb.toString();
    }
    logger.debug(LogMessage.format("Checking match of request : '%s'; against '%s'", url, this.pattern));
    return this.pattern.matcher(url).matches();
}
```

[RegexRequestMatcher 클래스 설정(2)]

step 2) 정규표현식을 이용한 인가 처리

정규표현식으로 지정한 경로인 '/admin/.*' 접근 시 인증을 요구한다.

```
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception{
        httpSecurity.authorizeRequests().regexMatchers("/admin/.*").authenticated();
    }
}
```

[정규표현식을 통한 URL 접근 권한 설정]

jshell 을 통해 지정한 정규표현식에 대해 정상 문자열과 개행 문자가 포함된 문자열을 비교한 결과는 다음과 같다.

```
jshell> import java.util.regex.Matcher;

jshell> import java.util.regex.Pattern;

jshell> Pattern pattern = Pattern.compile("/admin/.*");
pattern ==> /admin/.*

jshell> System.out.println(pattern.matcher("/admin/index").matches());
true

jshell> System.out.println(pattern.matcher("/admin/index₩r").matches());
false

jshell> System.out.println(pattern.matcher("/admin/in₩rdex").matches());
false
```

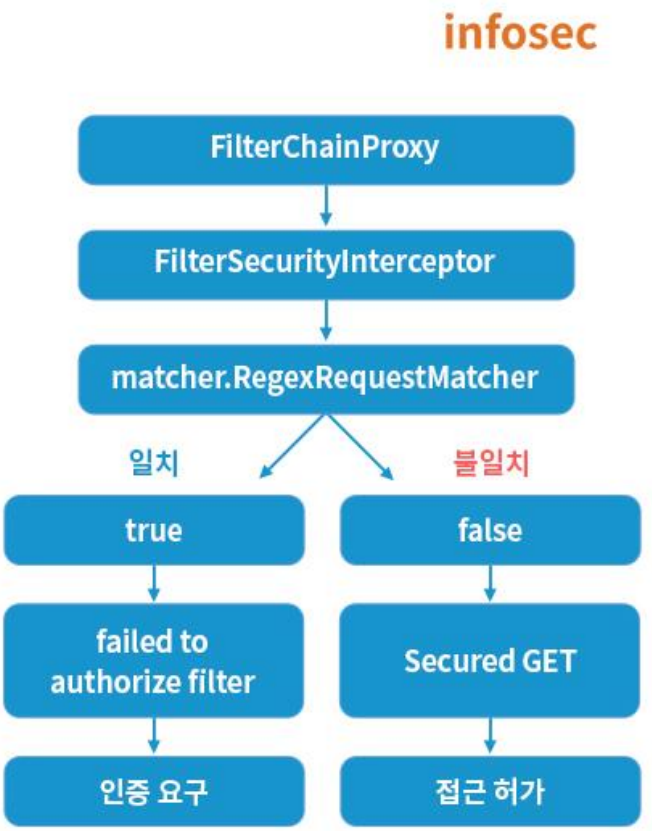
* ₩r 의 URL 인코딩 값은 %0d 이다.

[jshell을 통한 정규표현식 검증 결과]

/admin/ 경로 아래에 개행 문자를 포함한 문자에 대해서 false 값을 반환하는 것을 확인할 수 있다.

step 3) 반환 결과에 따른 인가 처리

Spring Security 에서 정규표현식을 통해 URL 접근 권한을 부여하는 순서는 다음과 같다. FilterChainProxy 로 URL 요청을 통과시킬 Filter Chain(필터 종류)을 결정하고, FilterSecurityInterceptor 를 통해 URL 요청에 보안 제약사항 적용 여부를 결정한다. 이후 RegexRequestMatcher 클래스에서 정규표현식을 통한 검증을 진행한다. 이때 CVE-2022-22978 취약점이 존재할 경우, 정규표현식과 URL 값이 일치할 경우 true 를 반환하고 일치하지 않을 경우 false 를 반환한다.



[URL 접근 권한 부여 순서]

1) URL의 값이 정규표현식과 일치할 경우(True)

```
rity.web.FilterChainProxy        : Securing GET /admin/index
SecurityContextPersistenceFilter : Set SecurityContextHolder to empty SecurityContext
a.AnonymousAuthenticationFilter  : Set SecurityContextHolder to anonymous SecurityContext
session.SessionManagementFilter  : Request requested invalid session id FF8F80B791DD1D34558B05EB3D141D67
u.matcher.RegexRequestMatcher    : Checking match of request : '/admin/index'; against '/admin/.*'
a.i.FilterSecurityInterceptor    : Failed to authorize filter invocation [GET /admin/index] with attributes [authenticated]
s.HttpSessionRequestCache        : Saved request http://localhost:49154/admin/index to session
legatingAuthenticationEntryPoint : Trying to match using And [Not [RequestHeaderRequestMatcher [expectedHeaderName=X-Requeste
HeaderValue=XMLHttpRequest]], MediaTypeRequestMatcher
NegotiationStrategy=org.springframework.web.accept.ContentNegotiationManager@6184ee74, matchingMediaTypes=[application/xhtml+
l, text/plain], useEquals=false, ignoredMediaTypes=[*/*]]]
legatingAuthenticationEntryPoint : Match found! Executing
ngframework.security.web.authentication.LoginUrlAuthenticationEntryPoint@4cbd03e7
b.DefaultRedirectStrategy        : Redirecting to http://localhost:49154/login
SessionSecurityContextRepository : Did not store empty SecurityContext
SessionSecurityContextRepository : Did not store empty SecurityContext
SecurityContextPersistenceFilter : Cleared SecurityContextHolder to complete request
rity.web.FilterChainProxy        : Securing GET /login
```

[값이 Ture일 때]

입력값	/admin/index
결과	Failed to authorize filter invocation [GET /admin/index] with attributes [authenticated]

RegexRequestMatcher 를 호출하여 지정한 정규표현식과 비교하여 입력받은 URL 에 대한 검증을 진행한다. /admin/index 에 대해 인증을 요구하는 것을 볼 수 있다. 따라서 /admin/index 에 대한 접근 권한이 없어 로그인 페이지를 리다이렉트하는 것을 볼 수 있다.

2) URL 의 값이 정규표현식과 일치하지 않을 경우(False)

```
o.s.security.web.FilterChainProxy        : Securing GET /admin/index
s.s.w.c.SecurityContextPersistenceFilter : Set SecurityContextHolder to empty SecurityContext
o.s.s.w.a.AnonymousAuthenticationFilter  : Set SecurityContextHolder to anonymous SecurityContext
o.s.s.w.session.SessionManagementFilter  : Request requested invalid session id FF8F80B791DD1D34558B05EB3D141D67
'; against '/admin/.*'
]
o.s.security.web.FilterChainProxy        : Secured GET /admin/index
w.c.HttpSessionSecurityContextRepository : Did not store anonymous SecurityContext
w.c.HttpSessionSecurityContextRepository : Did not store anonymous SecurityContext
s.s.w.c.SecurityContextPersistenceFilter : Cleared SecurityContextHolder to complete request
```

[값이 False일 때]

입력값	/admin/index%0d
결과	Secured GET /admin/index

입력값이 개행 처리되어 필터에서 누락된 것을 확인할 수 있다. 따라서 인증을 요구하는 /admin/이하의 경로를 우회하여 인증 없이 접근이 가능하기 때문에 Secured GET /admin/index 를 반환하는 것을 확인할 수 있다.

이처럼 취약한 버전의 Spring Security 가 RegexRequestMatcher 를 통해 정규표현식으로 접근 권한을 부여하고, 정규표현식에 '.'를 사용하고 있을 때 개행 문자에 대한 필터링 누락으로 인해 권한 우회가 가능하다.

Step 3. 취약점 패치

CVE-2022-22978 취약점의 패치 내역은 다음과 같다. 정규표현식을 다루는 클래스인 RegexRequestMatcher 클래스에 대한 코드가 수정된 것을 확인할 수 있다.

```
43  43    */
44  44   public final class RegexRequestMatcher implements RequestMatcher {
45  45
46      -    private static final int DEFAULT = 0;
        46   +    private static final int DEFAULT = Pattern.DOTALL;
        47   +
        48   +    private static final int CASE_INSENSITIVE = DEFAULT | Pattern.CASE_INSENSITIVE;
47  49
48  50        private static final Log logger = LogFactory.getLog(RegexRequestMatcher.class);
49  51
          @@ -68,7 +70,7 @@ public RegexRequestMatcher(String pattern, String httpMethod) {
68  70         * {@link Pattern#CASE_INSENSITIVE} flag set.
69  71         */
70  72        public RegexRequestMatcher(String pattern, String httpMethod, boolean caseInsensitive) {
71      -        this.pattern = Pattern.compile(pattern, caseInsensitive ? Pattern.CASE_INSENSITIVE : DEFAULT);
        73   +        this.pattern = Pattern.compile(pattern, caseInsensitive ? CASE_INSENSITIVE : DEFAULT);
72  74            this.httpMethod = StringUtils.hasText(httpMethod) ? HttpMethod.valueOf(httpMethod) : null;
73  75        }
74  76
```

[취약점 패치]

기존의 46번째 줄을 보면, 정규표현식을 처리하는 클래스 옵션이 DEFAULT모드에서 Pattern.DOTALL 모드로 변경된 것을 볼 수 있다.

패치 이전의 DEFAULT모드는 정규표현식을 처리할 때 개행 문자를 포함하지 않아, URL에 개행 문자를 추가할 경우 패턴에서 제외되어 우회가 가능했다. 수정된 Pattern.DOTALL 모드는 정규표현식의 '.'과 모든 문자가 매칭되며, 취약점의 원인인 개행 문자도 매칭에 포함되어 있어 우회가 불가능하다.

또한 기존의 71번째 줄에 대한 변경 사항도 확인할 수 있다. Pattern.comile()은 String 값으로 들어온 정규식을 Pattern 객체로 변환하는 역할을 한다. CASE_INSENSITIVE 옵션을 통해 패턴의 대소문자를 구분하지 않는다.

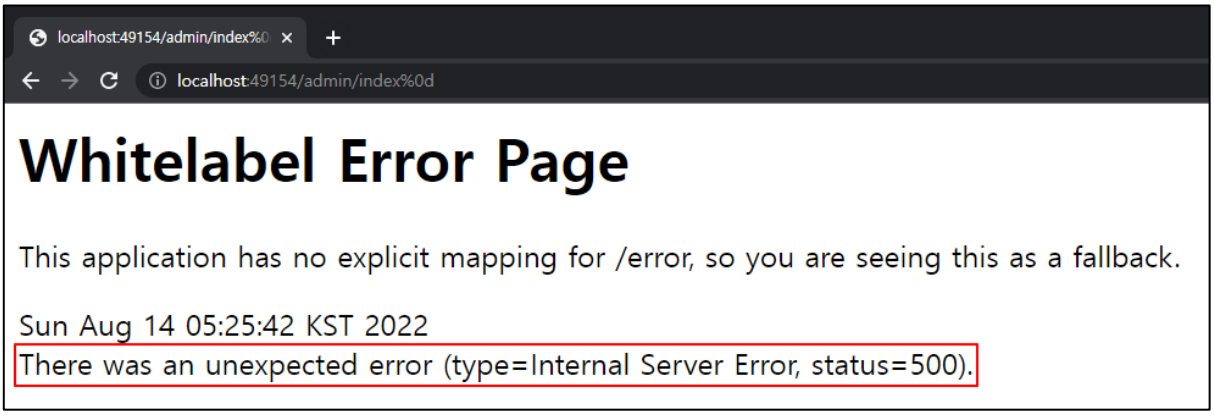
■ 대응 방안

CVE-2022-22978에 취약한 버전의 Spring Security를 사용하고 있다면 버전 업데이트를 고려해야 한다. 업데이트 시, 5.5.x 버전은 5.5.7 이상으로 5.6.x 버전은 5.6.4으로 업데이트해야 한다.

Spring Security 5.5.7
- URL: https://github.com/spring-projects/spring-security/releases/tag/5.5.7

Spring Security 5.6.4
- URL: https://github.com/spring-projects/spring-security/releases/tag/5.6.4

Spring Security 버전 업데이트 이후 CVE-2022-22978 취약점의 동일한 페이로드로 공격을 시도해 보면, 서버 측 에러를 반환하는 것을 볼 수 있다.



[업데이트 이후 공격 결과]

■ 참고 사이트

- URL: https://tanzu.vmware.com/security/cve-2022-22978
- URL: https://github.com/spring-projects/spring-security/commit/70863952aeb9733499027714d38821db05654856