CVE-2022-22978_Spring-security 认证绕过漏洞

漏洞描述

当 Spring-security 使用 RegexRequestMatcher 进行权限配置，由于 RegexRequestMatcher 正则表达式配置权限的特性，正则表达式中包含“.”时，未经身份验证攻击者可以通过构造恶意数据包绕过身份认证。

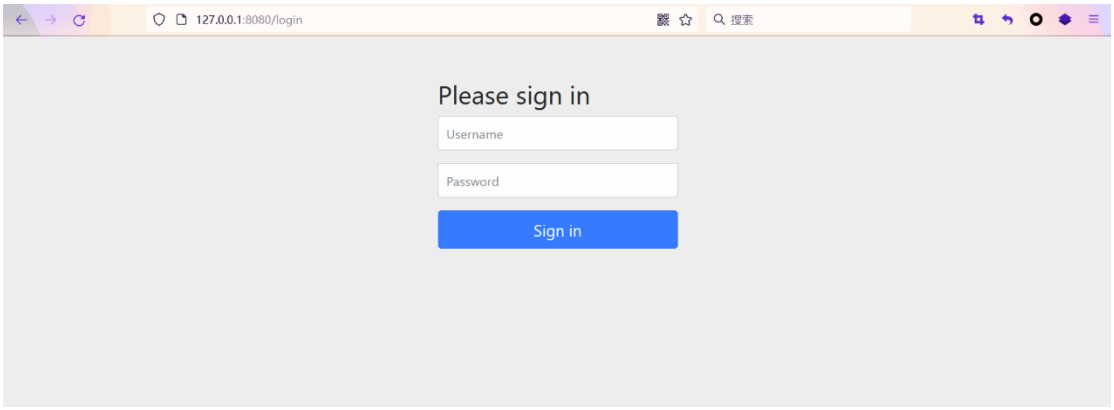
影响版本

Spring Security 5.5.x < 5.5.7

Spring Security 5.6.x < 5.6.4

环境搭建

使用 github 上的漏洞环境



漏洞分析

具体漏洞产生的原理分析可参考 CVE-2022-32532　Apache Shiro RegExPatternMatcher 认证绕过漏洞

https://mp.weixin.qq.com/s?__biz=Mzg3NDcwMDk3OA==&mid=2247483864&idx=1&sn=6c02552494d488652ed20f20425854f1&chksm=cecd8805f9ba01130cbf0f11d0277eac9aee8a448132587f75041727915160ea27410a104fc8&scene=126&&sessionid=1662033547#rd

构造的漏洞应用场景

创建一个 Controller，自定义接口

```java
package person.xu.vulEnv;


import ...

@Controller
public class WebController {
    @GetMapping("/")
    @ResponseBody
    public String index() { return "welcome"; }

    @GetMapping("/admin/{name}")
    @ResponseBody
    public String admin(@PathVariable String name) { return "welcome " + name; }
}

```

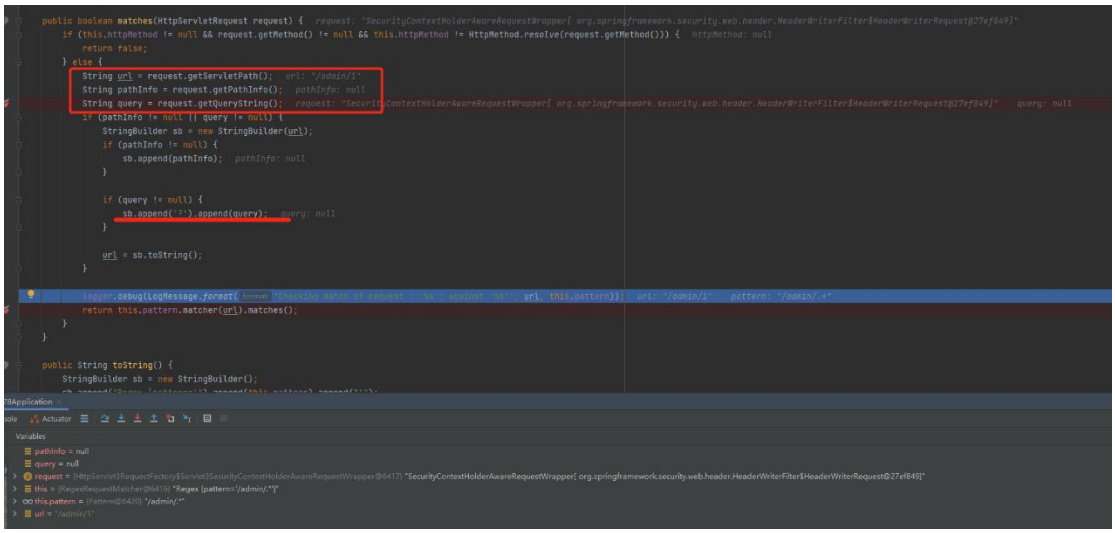
通过正则表达式添加认证配置

```java
import ...

@Configuration
@EnableWebSecurity
public class AuthConfig {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {  http: HttpSecurity@5562
        // @formatter:off
        http  http: HttpSecurity@5562
                .authorizeHttpRequests((authorize) -> authorize
                        .regexMatchers( ...regexPatterns: "/admin/.*").authenticated()
                )
                .httpBasic(withDefaults())
                .formLogin(withDefaults());
        // @formatter:on
        return http.build();
    }

```

在访问/admin/{name}接口时，需要认证才能访问

因为之前对 CVE-2022-32532　Apache Shiro RegExPatternMatcher 认证绕过的分析，此次很快就将漏洞点定位在 spring-security-web-5.6.3.jar 中

org.springframework.security.web.uti.matcher.RegexRequestMatcher#matchers



request.getServletPath()会对字符解码 并且会将;之后的字符到/字符删除，随后通过 getServletPath 获取 URL，尝试提取？后的参数进行拼接，然后使用正则表达式匹配

联想到在 HttpServletRequest 中 URL 解析函数中利用;进行权限绕过的场景：

其中 HTTPServletRequest 中对 URL 路径的几种解析方法

```
SQL
request.getRequestURL()：返回全路径；
request.getRequestURI()：返回除去 Host（域名或 IP）部分的路径；
request.getContextPath()：返回工程名部分，如果工程映射为/，则返回为空；
request.getServletPath()：返回除去 Host 和工程名部分的路径；
request.getPathInfo()：仅返回传递到 Servlet 的路径，如果没有传递额外的路径信息，则此返回 Null；
```

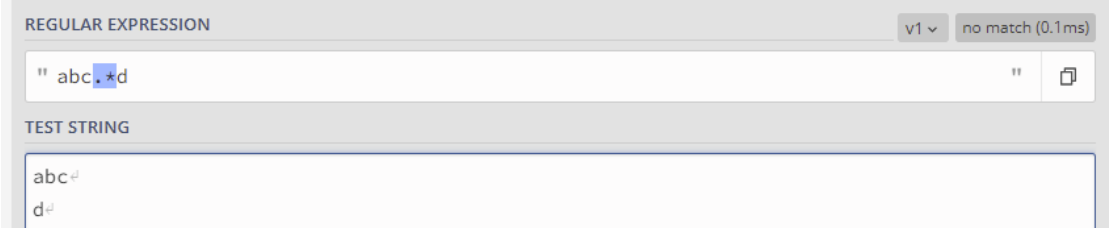
当认证接口使用 getRequestURI()或 getRequestURL()函数来解析用户请求的 URL 时，若 URL 中包含了一些特殊符号，如分号；就可能产生限制绕过。

而在 spring-security 中，存在 StrictHttpfirewall 机制默认对特殊字符进行过滤，所以使用/admin/..;/1 这种方式也就无法绕过

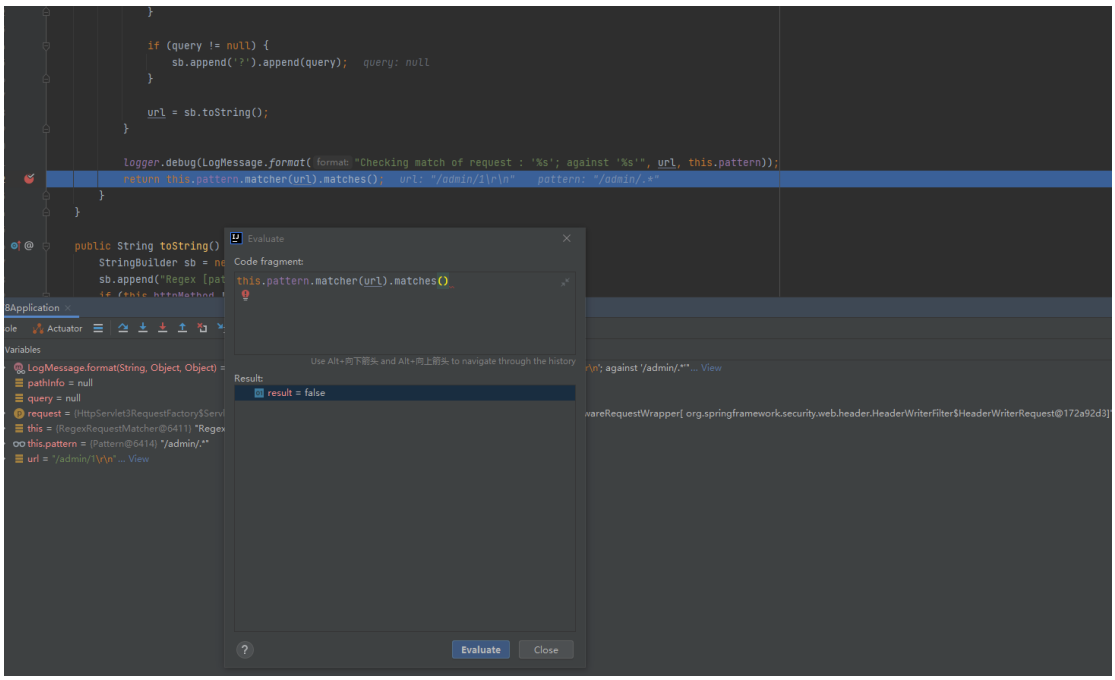
```
package org.springframework.security.web.firewall;

import ...

public class StrictHttpFirewall implements HttpFirewall {
    private static final Set<String> ALLOW_ANY_HTTP_METHOD = Collections.emptySet();
    private static final String ENCODED_PERCENT = "%25";
    private static final String PERCENT = "%";
    private static final List<String> FORBIDDEN_ENCODED_PERIOD = Collections.unmodifiableList(Arrays.asList("%2e", "%2E"));
    private static final List<String> FORBIDDEN_SEMICOLON = Collections.unmodifiableList(Arrays.asList(";", "%3b", "%3B"));
    private static final List<String> FORWARDSLASH = Collections.unmodifiableList(Arrays.asList("%2f", "%2F"));
    private static final List<String> FORBIDDEN_DOUBLE_FORWARDSLASH = Collections.unmodifiableList(Arrays.asList("//", "%2f%2f", "%2f%2F", "%2F%2f", "%2F%2F"));
    private static final List<String> FORBIDDEN_BACKSLASH = Collections.unmodifiableList(Arrays.asList("\\", "%5c", "%5C"));
    private static final List<String> FORBIDDEN_NULL = Collections.unmodifiableList(Arrays.asList("\u0000", "%00"));
    private Set<String> encodedUrlBlocklist = new HashSet();
    private Set<String> decodedUrlBlocklist = new HashSet();
    private Set<String> allowedHttpMethods = createDefaultAllowedHttpMethods();
    private Predicate<String> allowedHostnames = (hostname) -> {
        return true;
    };
    private static final Pattern ASSIGNED_AND_NOT_ISO_CONTROL_PATTERN = Pattern.compile("[\\p{IsAssigned}&&[^\\p{IsControl}]]*");
    private static final Predicate<String> ASSIGNED_AND_NOT_ISO_CONTROL_PREDICATE = (s) -> {
        return ASSIGNED_AND_NOT_ISO_CONTROL_PATTERN.matcher(s).matches();
    };
    private Predicate<String> allowedHeaderNames;
    private Predicate<String> allowedHeaderValues;
    private Predicate<String> allowedParameterNames;
    private Predicate<String> allowedParameterValues;

    public StrictHttpFirewall() {
        this.allowedHeaderNames = ASSIGNED_AND_NOT_ISO_CONTROL_PREDICATE;
        this.allowedHeaderValues = ASSIGNED_AND_NOT_ISO_CONTROL_PREDICATE;
        this.allowedParameterNames = ASSIGNED_AND_NOT_ISO_CONTROL_PREDICATE;
        this.allowedParameterValues = (value) -> {
            return true;
        };
        this.urlBlocklistsAddAll(FORBIDDEN_SEMICOLON);
        this.urlBlocklistsAddAll(FORWARDSLASH);
        this.urlBlocklistsAddAll(FORBIDDEN_DOUBLE_FORWARDSLASH);
        this.urlBlocklistsAddAll(FORBIDDEN_BACKSLASH);
        this.urlBlocklistsAddAll(FORBIDDEN_NULL);
```

不过在之前分析 shiro 认证绕过也提到，在正则表达式中元字符“.”是匹配除换行符

（\n、\r）之外的任何单个字符，在 java 中的正则默认情况下“.”也同样不会包含\n、\r 字符，所以 RegexRequestMatcher 在进行正则匹配时不会处理\n、\r



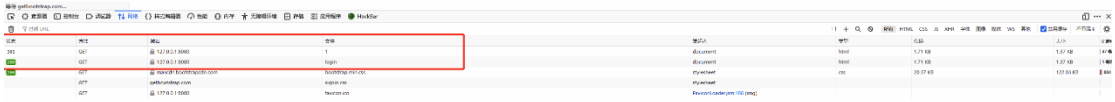
在 spring-security 中利用换行符可实现权限认证进行绕过，\r 的 URl 编码为%0d，\n 的 URL 编码为%0a



这样的绕过方式也和 CVE-2022-32532　Apache Shiro RegExPatternMatcher 认证绕过如出一辙。

漏洞复现

访问/admin/1 会 302 跳转至登陆页面进行权限验证



使用/admin/1%0d%0a，成功绕过登陆认证