

CVE-2020-11989: Apache Shiro权限绕过复现

原创 hatjwe Timeline Sec

2020-11-19原文

收录于话题

#漏洞复现文章合集

70个

上方蓝色字体关注我们，一起学安全！

作者：hatjwe@Timeline Sec

本文字数：1142

阅读时长：3~4min

声明：请勿用作违法用途，否则后果自负

0x01 简介

Apache

Shiro作为常用的Java安全框架，拥有执行身份验证、授权、密码和会话管理等功能，通常会和Spring等框架一起搭配使用来开发Web应用。

0x02 漏洞概述

编 号 : CVE-2020-11989

Apache Shiro 1.5.3 之前的版本中，当将 Apache Shiro与Spring动态控制器一起使用时，精心编制的请求可能会导致绕过身份验证。

0x03 影响版本

Apache Shiro < 1.5.3

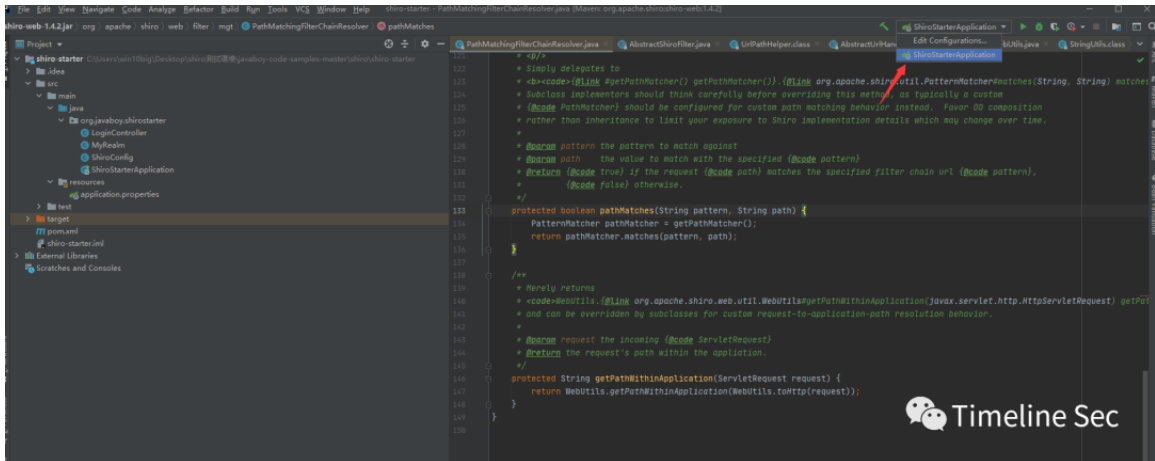
0x04 环境搭建

我测试的demo:

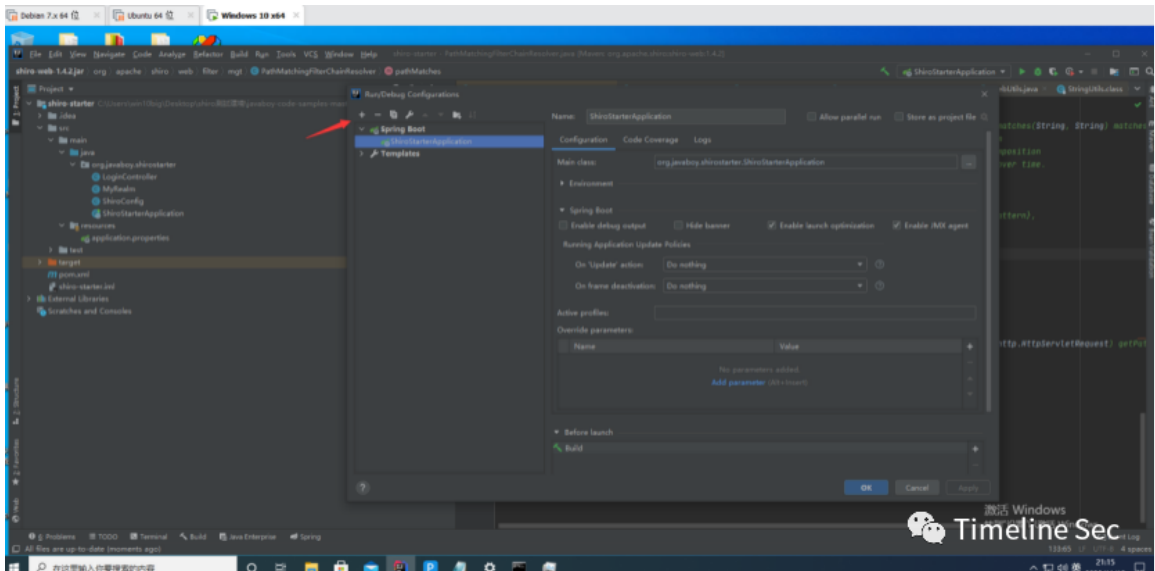
<https://github.com/lenve/javaboy-code-samples/tree/master/shiro/shiro-starter>

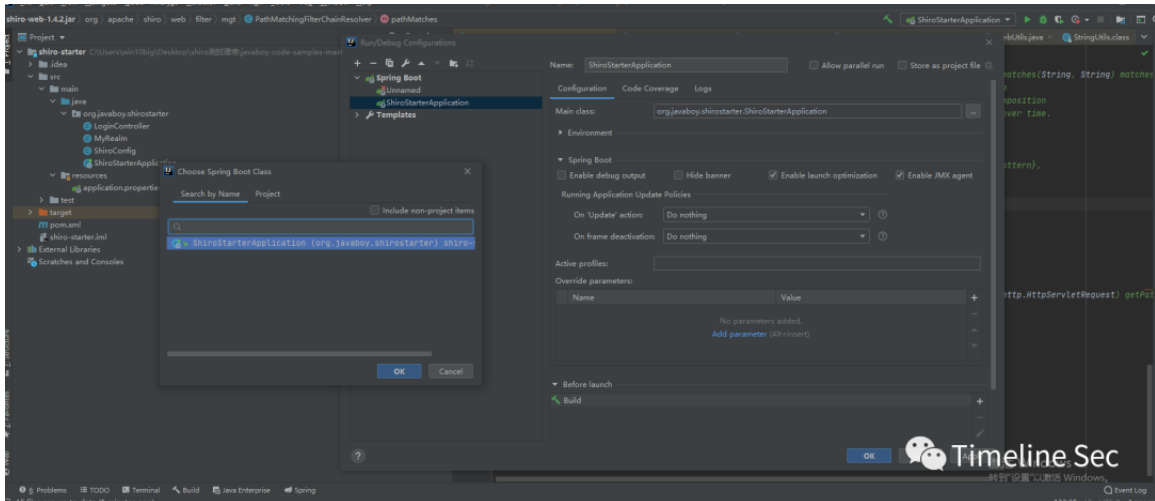
下载后在IDEA打开我们下载的项目文件夹

接着设置 Configurations ， 打开 edit Configurations(这里我之前已经设置过了)



配置 Configurations 点击加号 添加 spring Boot, 填写配置名称设置main class之后我们便可以运行了





Shiro主要的三个文件：

ShiroConfig, LoginController, Myrealm

权限配置：ShiroConfig

其中/doLogin无需权限验证即可访问，用于登录界面

而test/文件下目录需要登陆权限认证后才可以进行访问

```
20 *
21 * 1. 首先需要提供一个 Realm 的实例
22 * 2. 需要配置一个 SecurityManager, 在 SecurityManager 中配置 Realm
23 * 3. 配置一个 ShiroFilterFactoryBean, 在 ShiroFilterFactoryBean 中指定路径拦截规则等
24 */
25 @Configuration
26 public class ShiroConfig {
27     @Bean
28     MyRealm myRealm() { return new MyRealm(); }
31
32     @Bean
33     DefaultWebSecurityManager securityManager() {
34         DefaultWebSecurityManager manager = new DefaultWebSecurityManager();
35         manager.setRealm(myRealm());
36         return manager;
37     }
38
39     @Bean
40     ShiroFilterChainDefinition shiroFilterChainDefinition() {
41         DefaultShiroFilterChainDefinition definition = new DefaultShiroFilterChainDefinition();
42         definition.addPathDefinition( antPath: "/doLogin", definition: "anon");
43         definition.addPathDefinition( antPath: "/test/**", definition: "authc");
44         definition.addPathDefinition( antPath: "/hello/**", definition: "au
45     }
46 }
```

登陆控制：LoginController

其中设置了访问目录返回的内容

```
1 package org.javaboy.shirostarter;
2
3 import org.springframework.web.bind.annotation.*;
4
5 /**
6  * @author 江海一点雨
7  * @date www.javaboy.org 2019-06-05 11:24
8  */
9
10 @RestController
11 public class LoginController {
12     @GetMapping("/doLogin")
13     public void doLogin(String username, String password) {
14         Subject subject = SecurityUtils.getSubject();
15         try {
16             subject.login(new UsernamePasswordToken(username, password));
17             System.out.println("登录成功");
18         } catch (AuthenticationException e) {
19             e.printStackTrace();
20             System.out.println("登录失败");
21         }
22     }
23
24     @GetMapping("/hello/{currentPage}")
25     public String hello(@PathVariable Integer currentPage) { return "hello"; }
26
27     @GetMapping("/secret")
28     public String secret() { return "secret"; }
29
30     @GetMapping("/login")
31     public String login() { return "please login!"; }
32 }
```

Myrealm：其中储存着用户类信息像我们登陆所用的密码。

```
1 package org.javaboy.shirostarter;
2
3 import org.apache.shiro.authc.*;
4 import org.apache.shiro.authz.*;
5 import org.apache.shiro.realm.AuthorizingRealm;
6
7 /**
8  * @author 江海一点雨
9  * @date www.javaboy.org 2019-06-05 11:18
10 */
11
12 public class MyRealm extends AuthorizingRealm {
13     @Override
14     protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) { return null; }
15
16     @Override
17     protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws AuthenticationException {
18         String username = (String) token.getPrincipal();
19         if (!"javaboy".equals(username)) {
20             throw new UnknownAccountException("账户不存在!");
21         }
22         return new SimpleAuthenticationInfo(username, "123", getName());
23     }
24 }
```

0x05 漏洞复现

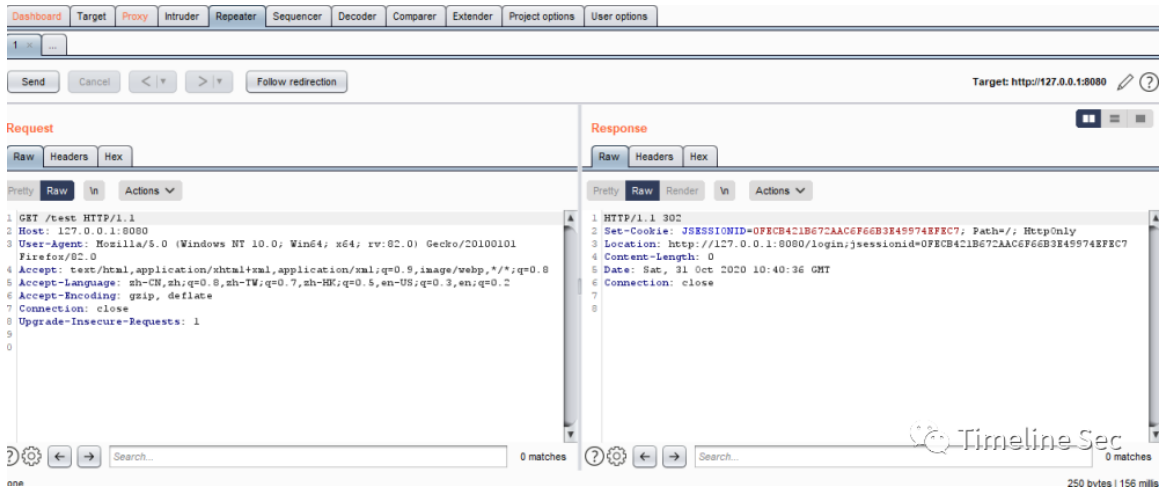
IDEA运行时，我们先访问界面通过doLogin进行登陆
(这里需要注意demo中doLogin传参方式为@PostMapping()需要改成@GetMapping())



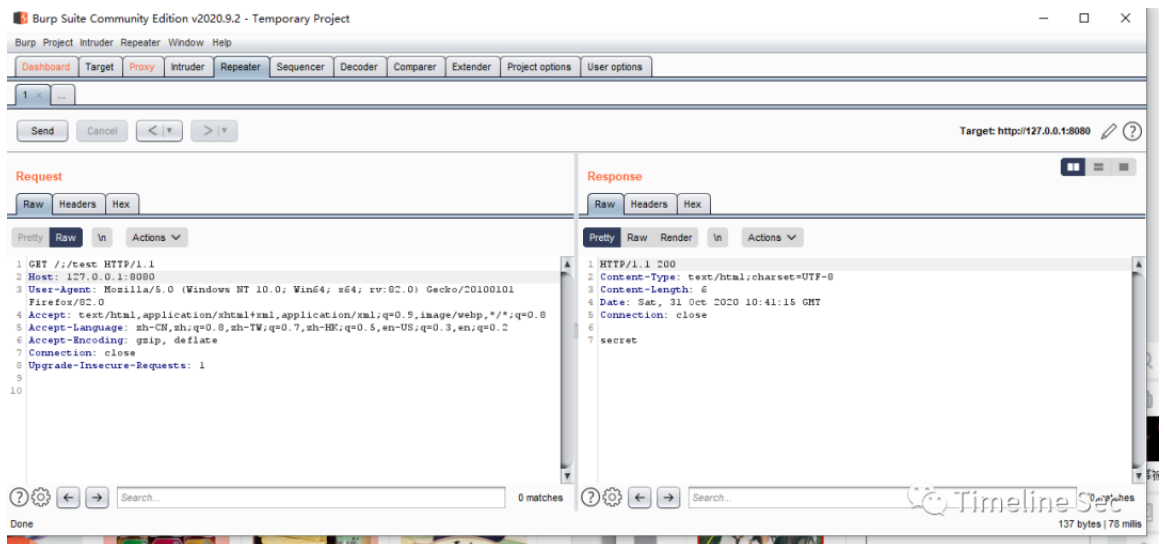
我们访问界面可以看到我们设置的回显信息，那么接下来我们开始通过未授权绕过shiro控制访问到这个信息



我清除了cookie,抓包进行访问test发现需要登陆



但是当访问`/;/test`时我们便可以绕过shiro认证查看需要登录认证的信息了



0x06 漏洞分析

根据参考文章漏洞初始成因定位到

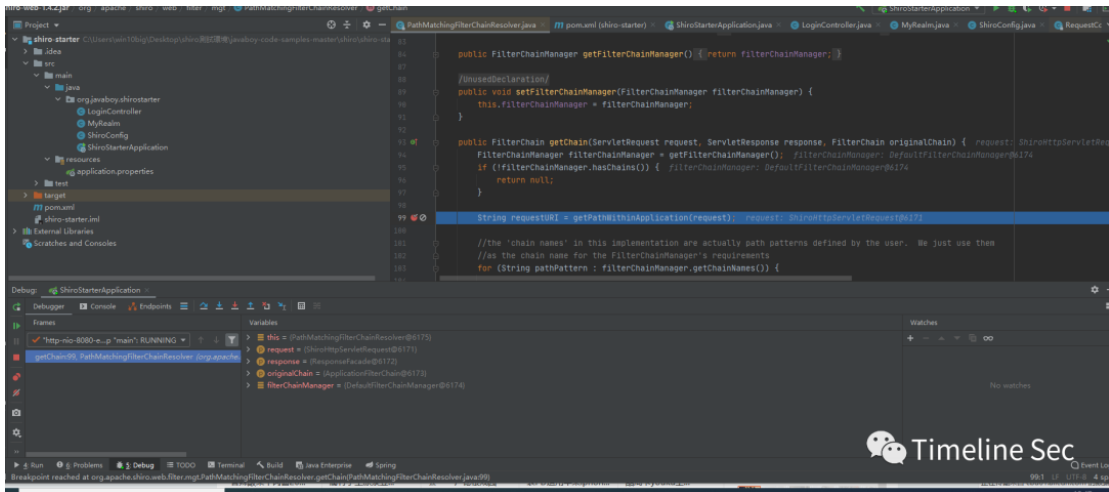
PathMatchingFilterChainResolver的getChain函数下

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ShiroFiter执行权限操作的

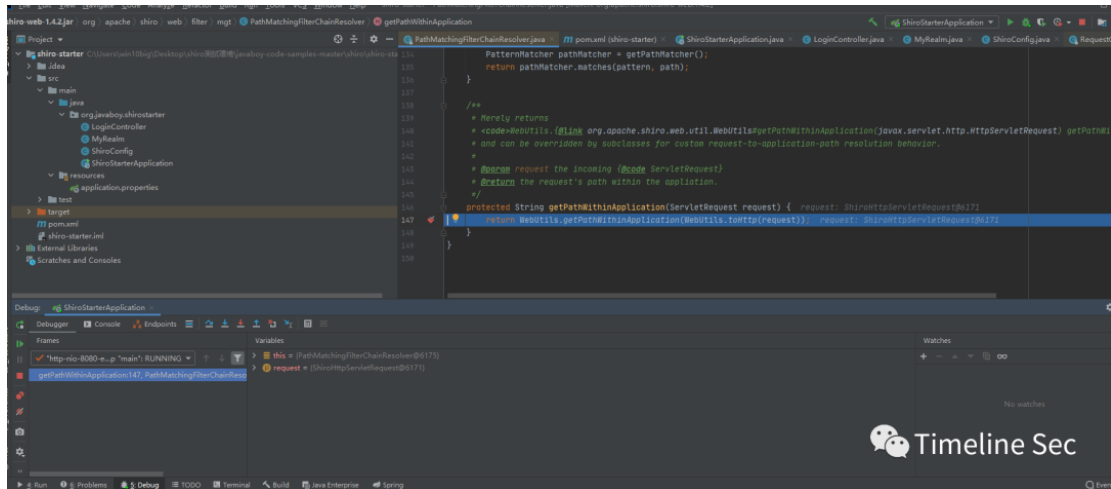
其中对于URL路径表达式和输入URL的匹配主要通过pathMathches函数进行匹配。

接下来我们在IDEA中进行debug,首先在PathMatchingFilterChainResolver.java文件中的getPathWithinApplication(Request)处下断点进行调试

浏览器访问：<http://127.0.0.1:8080/test>



我们跟进getPathWithinApplication



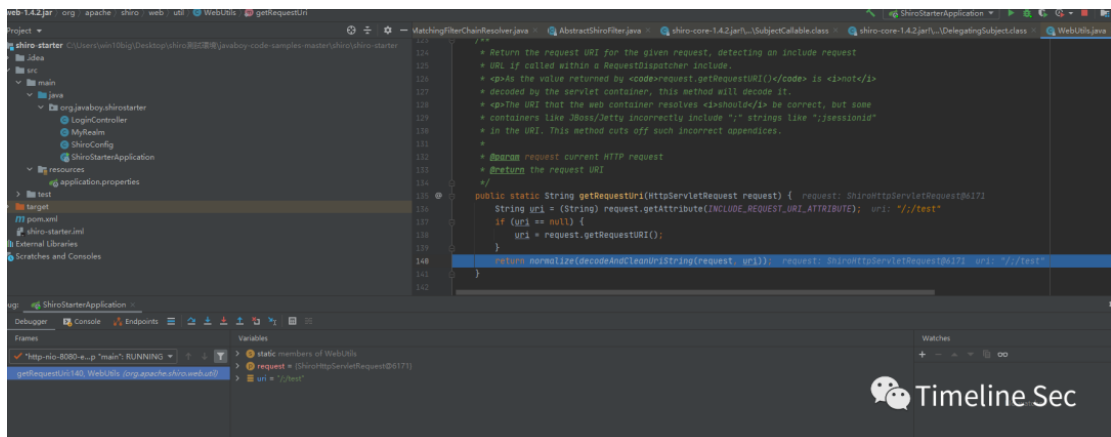
跟进到

org.apache.shiro.web.util.WebUtils#getPathWithinApplication

继续跟进到

org.apache.shiro.web.util.WebUtils#getRequestUri

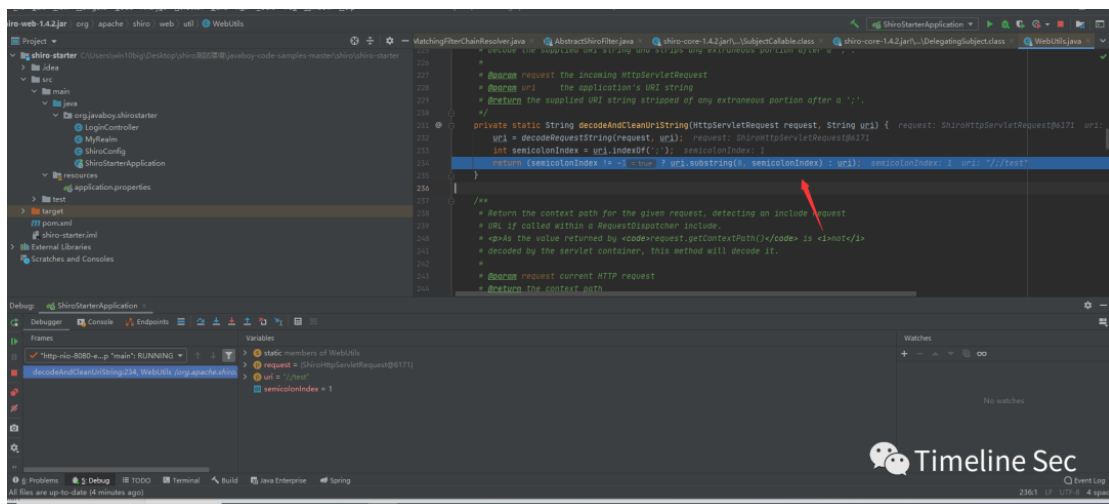
获取的是我们输入的路径



跟进到

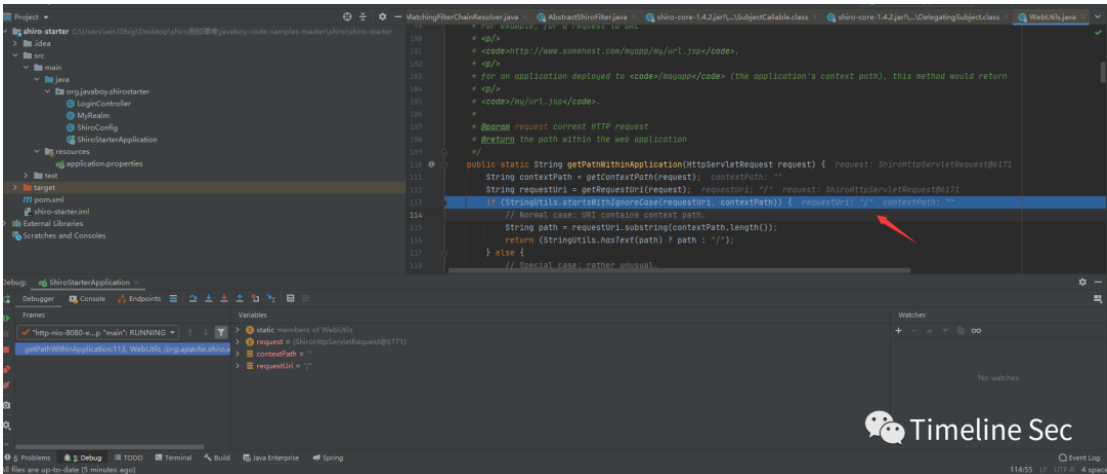
org.apache.shiro.web.util.WebUtils#normalize(decodeAndCleanUriString(request, uri))

这里会进行判断将;后边进行截断

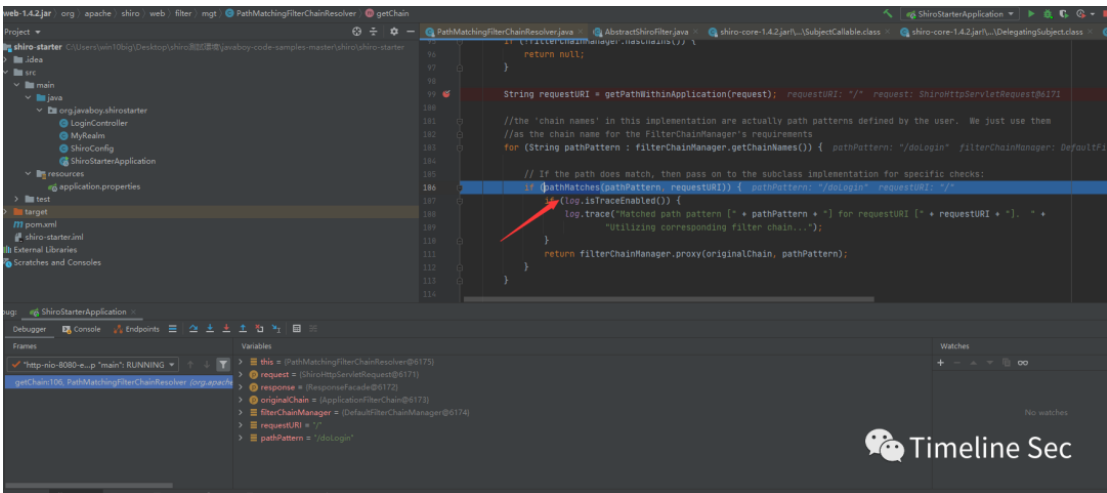


之后可以看到

org.apache.shiro.web.util.WebUtils#getRequestUri
获取到的是/



之后回到最开始的最开始的getPathVariableWithinApplication往下调试到pathMatches进行权限的判断



但我们接下来继续跟踪会发现我们直接绕过了if (pathMatches(pathPattern, requestURI))的逻辑跳出了shiro权限判断

0x07 修复方式

通过WAF检测请求的uri中是否包含%25%32%66关键词

通过WAF检测请求的uri开头是否为/;关键词

升级至Apache Shiro 1.5.3 或更高版本

参考链接：

<https://www.bilibili.com/video/BV1Ca4y1L7j3?t=5>

<https://paper.seebug.org/1196/>



阅读原文看更多复现文章

Timeline Sec 团队

安全路上，与你并肩前行

精选留言

用户设置不下载评论

[阅读全文](#)