

CVE-2020-1957

于3月 27, 2020由SWRhapsody发布

Introduction

这个漏洞需要1.5.2 版本以下的 shiro 并与 spring 一同使用才能触发，也就是说打包成war发布到 jetty 或 tomcat 的不受影响，具体通告[1]。

Code

准备

```
1 git clone https://github.com/threedr3am/learnjavabug
2 cd learnjavabug/tree/master/shiro/auth-bypass(shiro%3C1.5.2)
3 mvn package
```

比较1.5.1 和1.5.2，修改最多的代码是 WebUtils.java



这一看可能还不太清楚问题出在哪，但 WebUtilsTest.groovy写出了所有的payload



可以看到添加了很多的

SWRhapsody

先看下代码 `src\main\java\com\threedr3am\bug\shiro\bypass\auth\config\ShiroConfig.java` 中配置了访问权限：

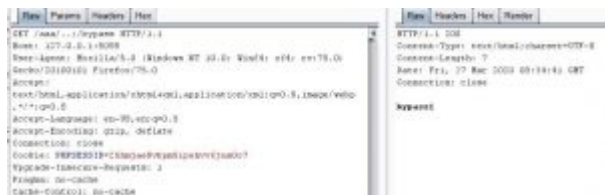
```

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

```

可以看到 `bypass` 下的url 都需要认证

用 IDE 或直接运行jar包的方式启动环境，直接请求 `/bypass` 系统会要求认证，使用 `“/aaa/./bypass”` 却可以直接访问。



这个漏洞的关键在与 `shiro` 对于 `servlet url` 的识别和 `spring` 对于 `servlet url` 的识别不一致导致的。在检查用户请求的 `URL` 是否需要认证的时候请求会进入

`org.apache.shiro.web.filter.mgt.PathMatchingFilterChainResolver` 中的

`public FilterChain getChain(ServletRequest request, ServletResponse response, FilterChain originalChain)`

```

60 public FilterChain getChain(ServletRequest request, ServletResponse response, FilterChain originalChain) {
61     FilterChainManager filterChainManager = getFilterChainManager();
62     if (filterChainManager.hasChain()) {
63         return null;
64     }
65     String requestURL = getPatternMatchingUtil(request);
66     // In spring web, the requestURL "/resource/resource" ----> "resource/resource" base can access the resource
67     // but the pattern match "/resource/resource" can not match "resource/resource"
68     // user can use requestURL + "/" to simply bypassed chain filter, to bypass shiro protect
69     if (requestURL != null && !requestURL.contains("/")) {
70         requestURL = requestURL + "/";
71     }
72 }

```

SWRhapsody

```
String contextUri = getContextPath(request);
String requestUri = getServletInfo(request);
if (StringUtils.startsWithIgnoreCase(requestUri, contextPath)) {
    // Normal case: URI contains context path.
    String path = requestUri.substring(contextPath.length());
    return (StringUtils.hasText(path) ? path : "/");
} else {
    // Special case: rather unusual.
    return requestUri;
}
```

```
public static String getRequestUri(HttpServletResponse request) {
    String url = (String) request.getAttribute(INCLUDE_REQUEST_URI_ATTRIBUTE);
    if (url == null) {
        url = request.getRequestURI();
    }
    return normalize(decodedUriLowestIString(request, url));
}

// request.getRequestURI()
// "/aaaa.../bypass"
// decodedUriLowestIString(request, url)
// "/aaaa..."
// url
// "/aaaa.../bypass"
```

可以看到 `PathMatchingFilterChainResolver` 中 `requestURI` 最后会是 `"/aaa/.."`, `decodeAndCleanUriString` 中会过滤掉 `“..”` 以后的所有内容。

接下来 shiro 会用 `ShiroConfig.java` 中预定的设置和这个 URL 进行匹配，当然一个也不会匹配成功，所以就绕过了所有的认证。

[illegible]

```

154 public string getRawPathFromRequestingObject(Request request) {
155     string pathToImage = getRawPathFromRequest(request);
156     string servicePath = getRawPathFromRequest();
157     string sanitizedImageApp = getSanitizedPath(pathToImage);
158     string path;
159
160     // If the app container sanitized the servicePath, check against the sanitized version
161     if (servicePath.Contains(sanitizedServicePath)) {
162         path = getRawPathFromRequest(sanitizedServicePath, servicePath, false);
163     }
164     else {
165         path = getRawPathFromRequest(pathToImage, servicePath, false);
166     }
167 }

```

```

301 // return the service path for the given request, regardless an include request
302 // or, if called within a RequestHandler::doGet()
303 // via the value returned by (Route request.getHandlerPath()) is already
304 // decided by the service container, this method will not attempt to decide it.
305 // given request current HTTP request
306 // returns the "base"
307
308 const R
309 public String
310     handlerPath: false // service.include_handler_new_address;
311     // (value: true)
312     // (value: false)
313     servicePath = request.getHandlerPath();
314
315 // (value: true)
316 // (value: false)
317 // on servers, in non-compile mode, for a "true" case that could be "false"
318 // on all other server containers: removing trailing slash, preventing with
319 // that removing slash as final lookup path...
320 servicePath = servicePath.substring(0, servicePath.length() - 1);
321
322 return servicePath;
323
324 }

```

因此对于“/aaa/./bypass”这样的url shiro 认为它是“/aaa/.”， spring 认为它是“/bypass”，导致绕过了 shiro 的认证。

SWRhapsody



杂念

这个漏洞我一开始想到了利用方式但环境没搭出来，折腾了半天的 jetty+spring mvc。

Reference

[1] <https://www.openwall.com/lists/oss-security/2020/03/23/2>

分类： EXPLOIT



0 条评论

发表评论

名称 *

电子邮件 *

网站

在想些什么?

SWRhapsody

[发表评论](#)

近期文章

[携程Apollo YAML 反序列化](#)[CVE-2020-5410](#)[CodeQL部分源码简读](#)[服务器与网关的不一致](#)[CodeQL 部分使用记录](#)

近期评论

文章归档

[2020年8月](#)[2020年6月](#)[2020年5月](#)[2020年3月](#)[2020年1月](#)[2019年12月](#)[2019年11月](#)[2019年8月](#)[2019年7月](#)[2019年5月](#)[2019年4月](#)

SWRhapsody

2018年11月

2018年10月

2018年9月

2018年4月

2018年3月

2018年2月

2018年1月

分类目录

Article Collection

Cheat Sheet

cryptography

Exercise

Exploit

HackTheBox

Penetration Test

Uncategorized

相关文章

SWRhapsody

携程Apollo YAML 反序列化

Introduction 3月份发现的一个问题，7月份提交给的携程SR [阅读更多...](#)

EXPLOIT

CVE-2020-5410

Introduction 补天挖的 spring-cloud-conf [阅读更多...](#)

EXPLOIT

CVE-2020-5405

Introduction 记得我刚开始写博客的时候是无比兴奋的，觉得终 [阅读更多...](#)

ABOUT

Hestia |由Themelsle开发