源代码缺陷检测报告
cmstoreos-sdk-java-1215

2024-04-29

1. 工程摘要

工程名称	cmstoreos-sdk-java-1215
检测时间	2024-04-29 13:09:56—2024-04-29 14:10:51
检测总用时	01时00分55秒
编程语言	JAVA
可执行代码行数	25132
文件数	367
代码总行数	44788

2. 检测结果概述

审计情况		
总量	未审计	已审计
59	27	32

已审计					
未判定问题，暂不处理	忽略	误报警	存在缺陷	存在缺陷，在下个版本处理	存在缺陷，以后再考虑处理
0	30	2	0	0	0

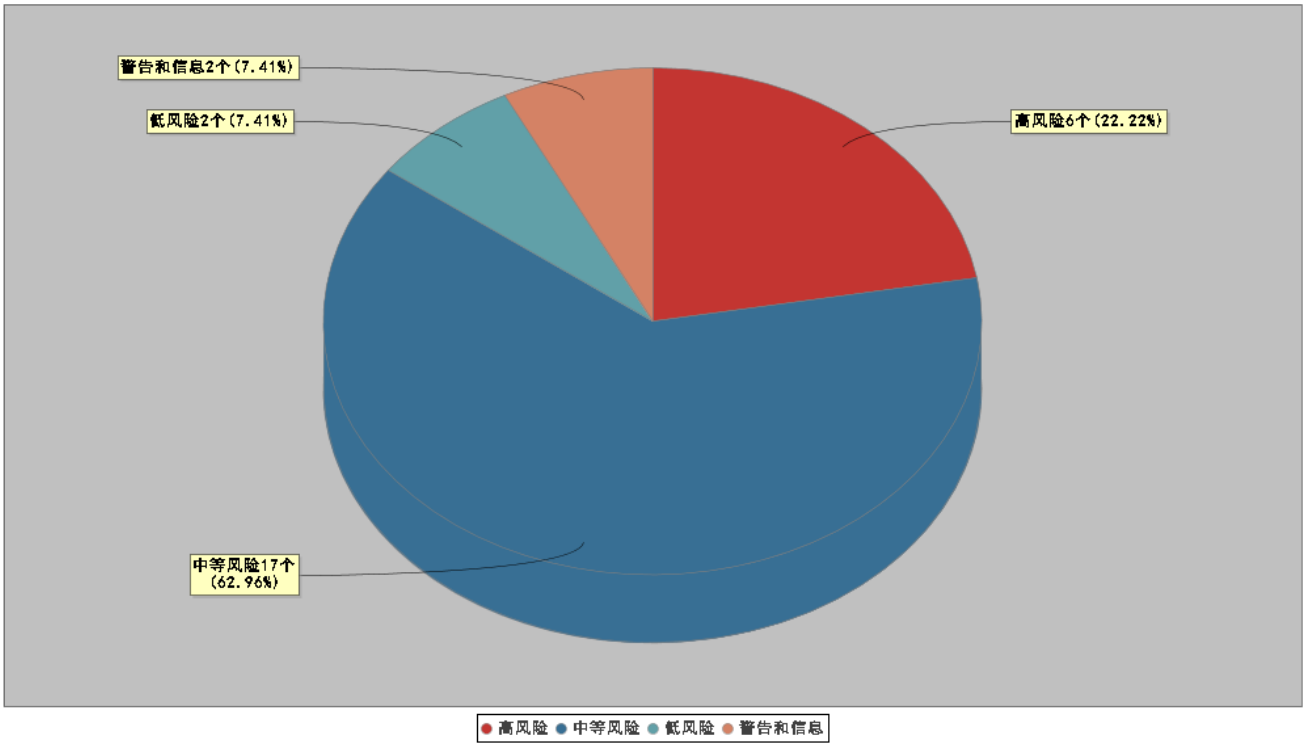
安全缺陷				
严重	高危	中等	低风险	警告和信息
0	6	1	0	0

质量缺陷				
严重	高危	中等	低风险	警告和信息
0	0	16	2	2

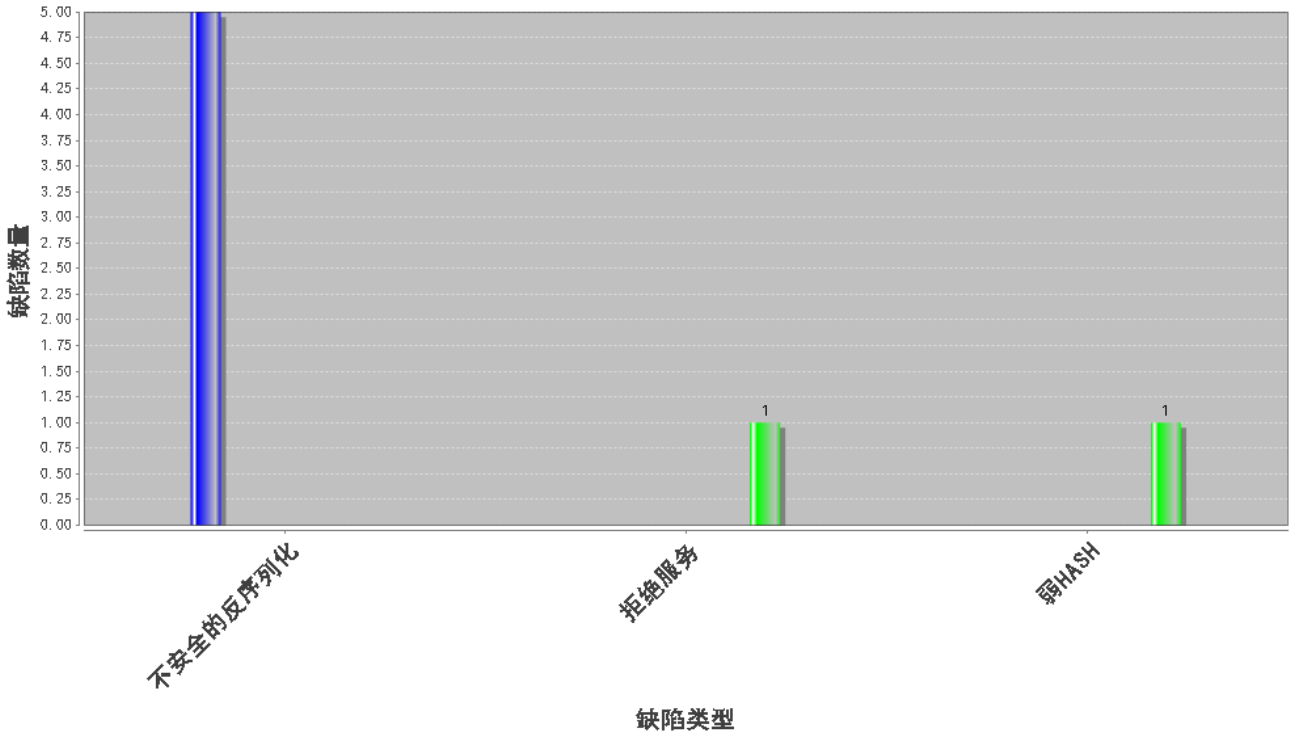
开源组件漏洞检测	
JAR文件	CVE漏洞
0	0

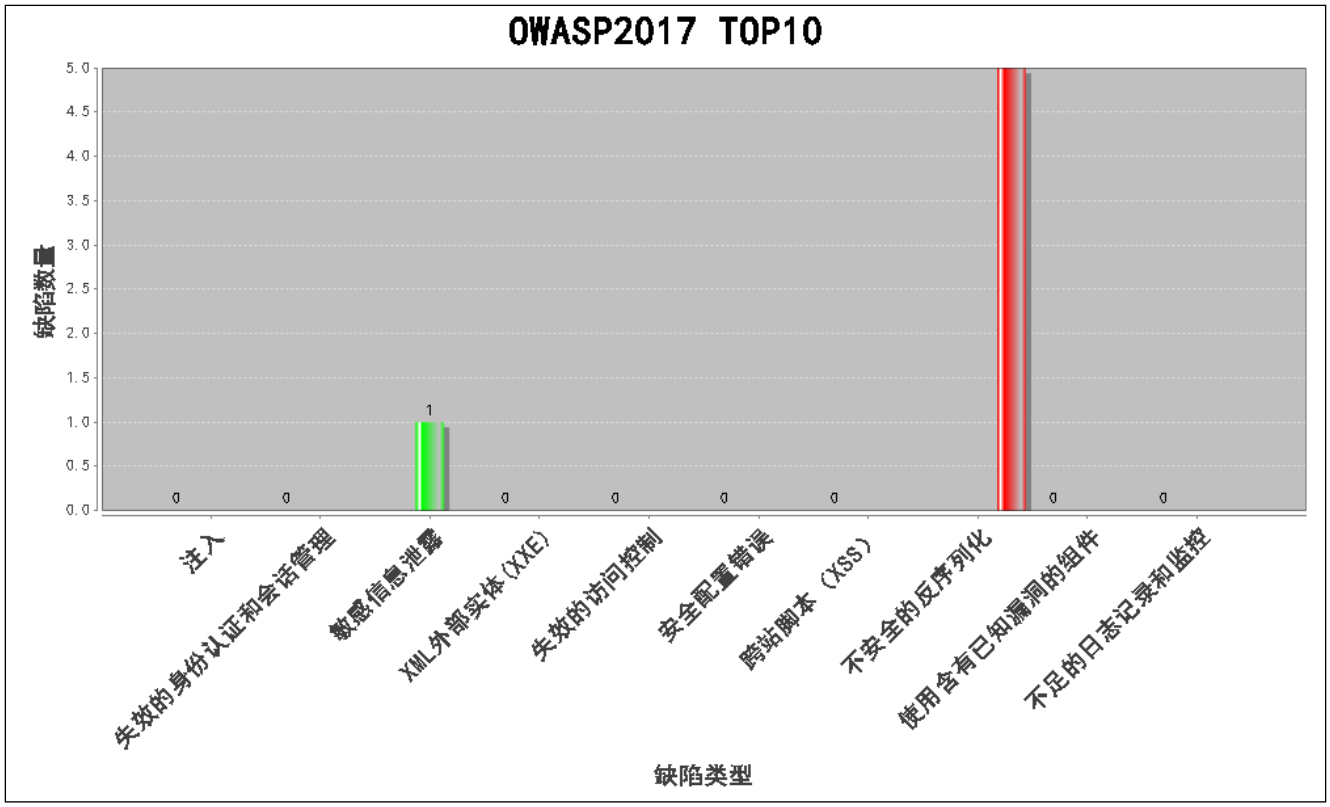
代码评分	
代码评分	千行代码缺陷密度
3星级	1.19‰

缺陷风险分布图



安全缺陷类型TOP10




OWASP2017 TOP10
缺陷数量
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
0
1
0
0
0
0
0
0
0
注入
失效的身份认证和会话管理
敏感信息泄露
XML外部实体(XXE)
失效的访问控制
安全配置错误
跨站脚本（XSS）
不安全的反序列化
使用含有已知漏洞的组件
不足的日志记录和监控
缺陷类型

3. 详细信息

3.1 高风险(6)

3.1.1. 安全缺陷

3.1.1.1. 不安全的反序列化(5)

Java 序列化会将对象图转换为字节流（包含对象本身和必要的元数据），以便通过字节流进行重构。开发人员可以创建自定义代码，以协助 Java 对象反序列化过程，在此期间，他们可以使用其他对象或代理替代反序列化对象。在对象重构过程中，并在对象返回至应用程序并转换为预期的类型之前，会执行自定义反序列化过程。到开发人员尝试强制执行预期的类型时，代码可能已被执行。

在必须存在于运行时类路径中且无法由攻击者注入的可序列化类中，会自定义反序列化例程，所以这些攻击的可利用性取决于应用程序环境中的可用类。令人遗憾的是，常用的第三方类，甚至 JDK 类都可以被滥用，导致 JVM 资源耗尽、部署恶意文件或运行任意代码。

原理、风险及预防

在运行时对用户控制的对象流进行反序列化，会让攻击者有机会在服务器上执行任意代码、滥用应用程序逻辑和/或导致 Denial of Service。

示例

正例:继承ObjectInputStream，并实现resolveClass，在内部设置白名单机制，只允许序列化已知的类。

```
public class SecureObjectInputStream extends ObjectInputStream {
  // Constructor here

  @Override
  protected Class<?> resolveClass(ObjectStreamClass osc) throws IOException,
ClassNotFoundException {
    // Only deserialize instances of AllowedClass
    if (!osc.getName().equals(AllowedClass.class.getName())) {
      throw new InvalidClassException("Unauthorized deserialization",
osc.getName());
    }
    return super.resolveClass(osc);
  }
}
```

```
public class RequestProcessor {
  protected void processRequest(HttpServletRequest request) {
    ServletInputStream sis = request.getInputStream();
    SecureObjectInputStream sois = new SecureObjectInputStream(sis);
    Object obj = sois.readObject();
  }
}
```

反例:

```
public class RequestProcessor {
  protected void processRequest(HttpServletRequest request) {
    ServletInputStream sis = request.getInputStream();
    ObjectInputStream ois = new ObjectInputStream(sis);
    Object obj = ois.readObject(); // Noncompliant
  }
}
```

序号: 1
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3DownloadOperation.java:71 (不安全的反序列化)
一级缺陷: 输入验证
二级缺陷: 不安全的反序列化
审计结果: 未审计
审计信息: 无
函数:
说明:
在运行时对用户控制的对象流进行反序列化，会让攻击者有机会在服务器上执行任意代码、滥用应用程序逻辑和/或导致 Denial of Service。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3DownloadOperation.java:71

序号: 2
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSDownloadOperation.java:71 (不安全的反序列化)
一级缺陷: 输入验证
二级缺陷: 不安全的反序列化
审计结果: 未审计
审计信息: 无
函数:
说明:
在运行时对用户控制的对象流进行反序列化，会让攻击者有机会在服务器上执行任意代码

、滥用应用程序逻辑和/或导致 Denial of Service。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSDownloadOperation.java:71

序号: 3
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSUploadOperation.java:73 (不安全的反序列化)
一级缺陷: 输入验证
二级缺陷: 不安全的反序列化
审计结果: 未审计
审计信息: 无
函数:
说明:
在运行时对用户控制的对象流进行反序列化，会让攻击者有机会在服务器上执行任意代码、滥用应用程序逻辑和/或导致 Denial of Service。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSUploadOperation.java:73

序号: 4
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/swift/operation/SwiftDownloadOperation.java:50 (不安全的反序列化)
一级缺陷: 输入验证
二级缺陷: 不安全的反序列化
审计结果: 未审计
审计信息: 无
函数:
说明:
在运行时对用户控制的对象流进行反序列化，会让攻击者有机会在服务器上执行任意代码、滥用应用程序逻辑和/或导致 Denial of Service。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/swift/operation/SwiftDownloadOperation.java:50

序号: 5
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3UploadOperation.java:73 (不安全的反序列化)
一级缺陷: 输入验证
二级缺陷: 不安全的反序列化
审计结果: 未审计
审计信息: 无
函数:

说明:
在运行时对用户控制的对象流进行反序列化，会让攻击者有机会在服务器上执行任意代码、滥用应用程序逻辑和/或导致 Denial of Service。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3UploadOperation.java:73

3.1.1.2. 拒绝服务(1)

Denial of Service 攻击试图让目标用户无法访问计算机或网络资源。如果应用程序存在 Denial Of Service (DoS) 漏洞，攻击者就能阻止合法用户访问由该应用程序提供的服务。通过控制网络数据包、编程漏洞、逻辑漏洞或资源处理等漏洞，可以采用多种方式阻止合法用户访问服务。攻击者可能通过对应用程序发送大量请求，而使它拒绝对合法用户的服务，但是这种攻击形式经常会在网络层就被排除掉了。导致出现更多问题的原因之一是资源释放不当的 bug，这种 bug 会导致应用程序在运行时停止。

原理、风险及预防
拒绝服务（DoS）攻击的目标是使得资源（站点、应用程序、服务器）不能实现它预订的设计目的。
如果一项服务收到了大量的请求，那么它可能会停止向合法用户提供。
同样的道理，当程序漏洞被利用时，服务会可能会停止或者处理它使用的资源的方式。
DoS攻击严重地降低了合法用户的使用体验。这些攻击会导致大量响应延迟，过度损耗和服务中断，从而直接影响可用性。

示例
修复建议：避免因流无限大，导致的死循环或内存溢出
反例:

```
 BufferedReader in = new BufferedReader(new FileReader(fileName));
        String line = null;
        while ((line = in.readLine()) != null)
        {
}
```

序号: 1
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/parser/ResponseParsers.java:1339 (拒绝服务)
一级缺陷: 输入验证
二级缺陷: 拒绝服务
审计结果: 未审计
审计信息: 无
函数:
说明: 如果应用程序存在拒绝服务漏洞，攻击者就能阻止合法用户访问由该应用程序提供

的服务。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/parser/ResponseParsers.java:1339

3.2 中等风险(17)

3.2.1. 安全缺陷

3.2.1.1. 弱HASH(1)

MD2、MD4、MD5、RIPEMD-160 和 SHA-1 是常用的加密散列算法，通常用于验证消息和其他数据的完整性。然而，由于最近的密码分析研究揭示了这些算法中存在的根本缺陷，因此它们不应该再用于安全性关键的上下文中。
由于有效破解 MD 和 RIPEMD 散列的技术已得到广泛使用，因此不应该依赖这些算法来保证安全性。对于 SHA-1，目前的破坏技术仍需要极高的计算能力，因此比较难以实现。然而，攻击者已发现了该算法的致命弱点，破坏它的技术可能会导致更快地发起攻击。

原理、风险及预防
弱加密散列值无法保证数据完整性，且不能在安全性关键的上下文中使用。

示例
修复建议: 停止使用 MD2、MD4、MD5、RIPEMD-160 和 SHA-1 对安全性关键的上下文中的数据进行验证。
目前，SHA-224、SHA-256、SHA-384、SHA-512 和 SHA-3 都是不错的备选方案。
但是，由于安全散列算法 (Secure Hash Algorithm) 的这些变体并没有像 SHA-1 那样得到仔细研究，因此请留意可能影响这些算法安全性的未来研究结果。

序号: 1
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/utils/BinaryUtil.java:34 (弱HASH)
一级缺陷: 安全问题(认证/访问控制/加密/权限等)
二级缺陷: 弱HASH
审计结果: 未审计
审计信息: 无
函数:
说明:
弱加密散列值无法保证数据完整性，且不能在安全性关键的上下文中使用。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/utils/BinaryUtil.java:34

3.2.2. 质量缺陷

3.2.2.1. 资源未释放：流(16)

程序可能无法成功释放某一项系统资源。
资源泄露至少有两种常见的原因：
- 错误状况及其他异常情况。
- 未明确程序的哪一部份负责释放资源。

大部分 Unreleased Resource 问题只会导致一般的软件可靠性问题，但如果攻击者能够故意触发资源泄漏，该攻击者就有可能通过耗尽资源池的方式发起 denial of service 攻击。

原理、风险及预防
程序可能无法成功释放某一项系统资源。

示例
正例：

```
OutputStream os = null;
try {
    URL url = new URL(strUrl);

    HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
    conn.connect();
    os = conn.getOutputStream();
    os.write(content.getBytes("UTF-8"));
    os.flush();} catch (Exception ex) {
    ex.printStackTrace();
} finally {
    safeCloseOutputStream(os);
}private void safeCloseOutputStream(@NonNull final OutputStream os) {
    if (os != null) {
        try {
            os.close();
        } catch (IOException e) {
            ex.printStackTrace();
        }
    }
}
```

反例：

```
try {
    URL url = new URL(strUrl);

    HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
```

```
    conn.connect();
    OutputStream os = conn.getOutputStream();
    os.write(content.getBytes("UTF-8"));
    os.flush();
    os.close();} catch (Exception ex) {
    ex.printStackTrace();
}
```

序号: 1
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSDownloadOperation.java:70 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSDownloadOperation.java:70

序号: 2
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSDownloadOperation.java:83 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSDownloadOperation.java:83

序号: 3
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSUploadOperation.java:85 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计

审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSUploadOperation.java:85

序号: 4
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/utils/IOUtils.java:165 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/utils/IOUtils.java:165

序号: 5
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3DownloadOperation.java:70 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3DownloadOperation.java:70

序号: 6
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3DownloadOperation.java:795 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。

传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3DownloadOperation.java:795

序号: 7
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3DownloadOperation.java:83 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3DownloadOperation.java:83

序号: 8
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3UploadOperation.java:72 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3UploadOperation.java:72

序号: 9
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/utils/AgentUtils.java:47 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/utils/AgentUtils.java:47

序号: 10
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSUploadOperation.java:72 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSUploadOperation.java:72

序号: 11
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSDownloadOperation.java:795 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSDownloadOperation.java:795

序号: 12
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/utils/IOUtils.java:125 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/utils/IOUtils.java:125

序号: 13
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/swift/operation/SwiftDownloadOperation.java:62 (资源未释放

：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/swift/operation/SwiftDownloadOperation.java:62

序号: 14
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3UploadOperation.java:85 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3UploadOperation.java:85

序号: 15
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/swift/operation/SwiftDownloadOperation.java:49 (资源未释放：流)
一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/swift/operation/SwiftDownloadOperation.java:49

序号: 16
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/swift/operation/SwiftDownloadOperation.java:710 (资源未释放：流)

一级缺陷: 代码质量
二级缺陷: 资源未释放：流
审计结果: 未审计
审计信息: 无
函数:
说明:
程序可能无法成功释放某一项系统资源。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/swift/operation/SwiftDownloadOperation.java:710

3.3 低风险(2)

3.3.1. 质量缺陷

3.3.1.1. 不良的异常处理(空Catch块)(2)

几乎每一个对软件系统的严重攻击都是从违反程序员的假设开始的。攻击后，程序员的假设看起来既脆弱又拙劣，但攻击前，许多程序员会在午休时间为自己的种种假设做很好的辩护。
在代码中，很容易发现两个令人怀疑的假设：“一是这个方法调用不可能出错；二是即使出错了，也不会对系统造成什么重要影响。”因此当程序员忽略异常时，这其实就表明了他们是基于上述假设进行的操作。

原理、风险及预防
忽略异常会导致程序无法发现意外状况和情况。

示例
反例：

```
public void bad(HttpServletRequest request, HttpServletResponse response){
        try {
            response.getWriter().write("You cannot shut down this application, only
the admin can");
          } catch (IOException e) {
          }
   }
```

正例：

```
public void good(HttpServletRequest request, HttpServletResponse response){

       try {
          response.getWriter().write("You cannot shut down this application, only the
admin can");
```

```
        } catch (IOException e) {
            log.info("bad");
        }
    }
```

序号: 1
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/request/DefaultServiceClient.java:378 (不良的异常处理(空Catch块))
一级缺陷: 错误处理(异常处理)
二级缺陷: 不良的异常处理(空Catch块)
审计结果: 未审计
审计信息: 无
函数:
说明:
忽略异常会导致程序无法发现意外状况和情况。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/request/DefaultServiceClient.java:378

序号: 2
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/comm/ServiceClient.java:294 (不良的异常处理(空Catch块))
一级缺陷: 错误处理(异常处理)
二级缺陷: 不良的异常处理(空Catch块)
审计结果: 未审计
审计信息: 无
函数:
说明:
忽略异常会导致程序无法发现意外状况和情况。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/comm/ServiceClient.java:294

3.4 警告和信息(2)

3.4.1. 质量缺陷

3.4.1.1. 忽略返回值(2)

Java 程序员常常会误解包含在许多
<code>java.io</code> 类中的
<code>read()</code> 及相关方法。在 Java 结果中，将大部分错误和异常事件都作为异常抛出。（这是 Java 相对于 C 语言等编程语言的优势：各种异常更加便于程序员考虑是哪里出现了问题。）但是，如果只有少量的数据可用，stream 和 reader 类并不认为这

是异常的情况。这些类只是将这些少量的数据添加到返回值缓冲区，并且将返回值设置为读取的字节或字符数。所以，并不能保证返回的数据量一定等于请求的数据量。
 这样，程序员就需要检查
 <code>read()</code> 和其他 IO 方法的返回值，以确保接收到期望的数据量。

原理、风险及预防
忽略方法的返回值会导致程序无法发现意外状况和情况。

示例
反例：

```
public String bad() {
        String filePath = "C:" + File.separator + "test" ;
        File f = new File(filePath);
        f.mkdir();
        return "ok";
    }
```

正例：

```
public String good() {
        String filePath = "C:" + File.separator + "test" ;
        File f = new File(filePath);
        boolean tag = f.mkdir();
        if(tag){
            return "ok";
        }else
            return "bad";
    }
```

序号: 1
缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3UploadOperation.java:449 (忽略返回值)
一级缺陷: API滥用(不安全的API)
二级缺陷: 忽略返回值
审计结果: 未审计
审计信息: 无
函数:
说明:
 忽略方法的返回值会导致程序无法发现意外状况和情况。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/s3/operation/S3UploadOperation.java:449

序号: 2

缺陷标识: share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSUploadOperation.java:449 (忽略返回值)
一级缺陷: API滥用(不安全的API)
二级缺陷: 忽略返回值
审计结果: 未审计
审计信息: 无
函数:
说明:
忽略方法的返回值会导致程序无法发现意外状况和情况。
传播路径:
share/code/cmstoreos-sdk-java-1215/com/heredata/hos/operation/HOSUploadOperation.java:449