OffensiveForBurp 0.2

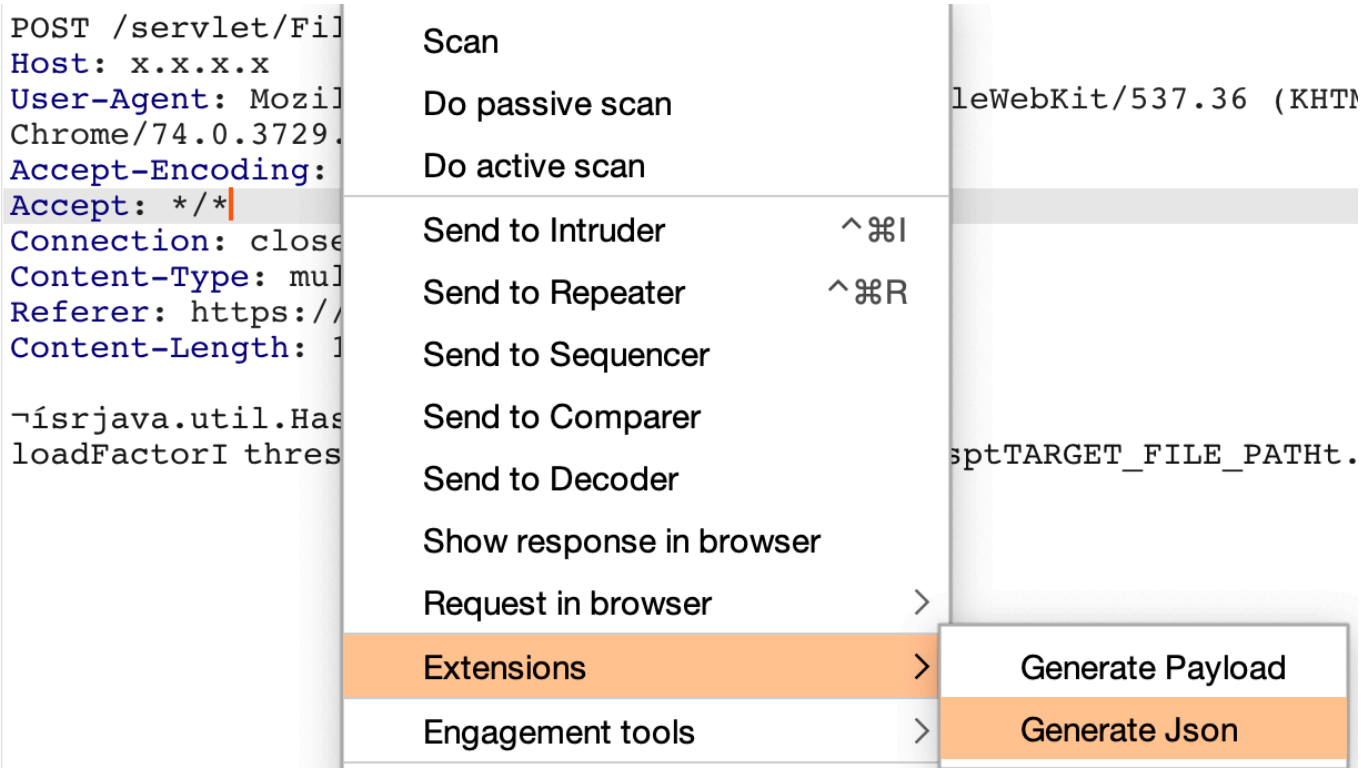
BurpSuite插件 - 漏洞利用

Update

- 每个请求的body换为base64形式，避免部分漏洞的请求体存为json，从Burp插件调用时出错。例如反序列化漏洞，请求体中包含byte类型数据，存为json文件时增加一层base64转为String，当使用漏洞时直接从base64转为字节，避免出错。
- 新增生成漏洞json文件辅助工具

Generate Json

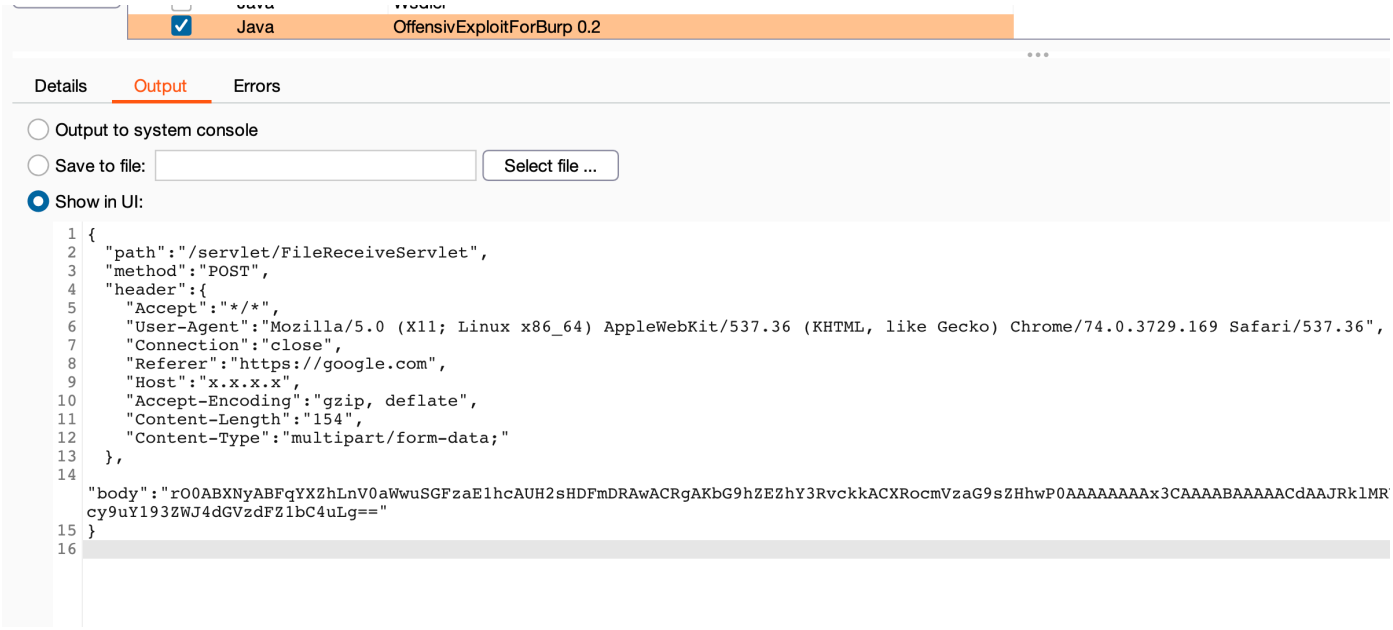
成功复现漏洞时，在repeater右键，选择Generate Json



在Extender的Output输出exploit

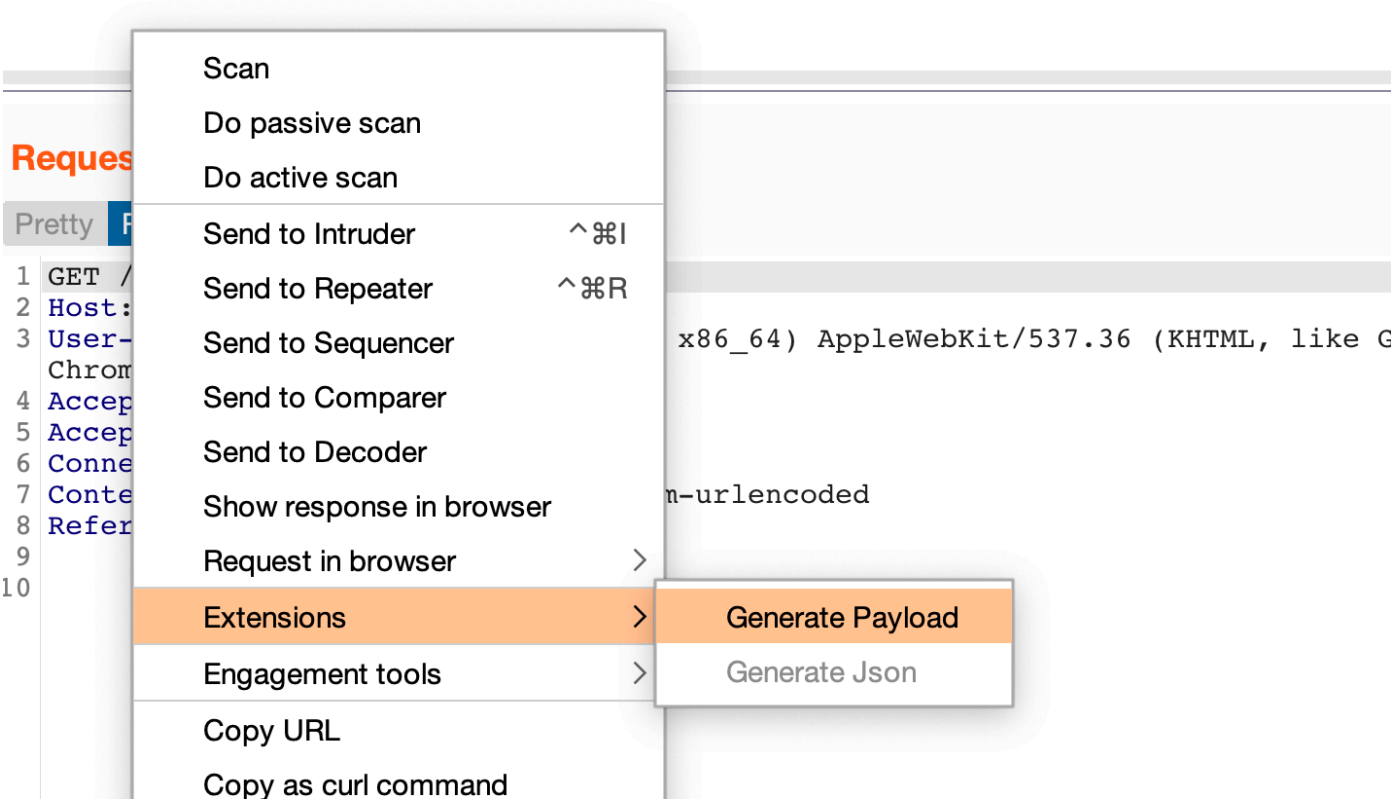
PS： 尽量只留需要使用的Header，把HOST等字段去掉，避免使用时将Cookie，HOST等信息刷掉。

PS：此处生成的只是单步exploit，在json文件中exploit是一个list，原因是部分漏洞需要多个请求完成。

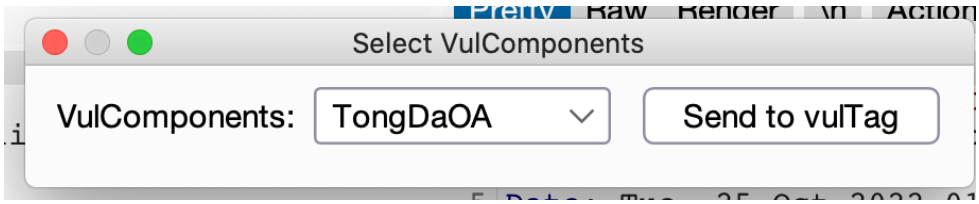


Generate Payload

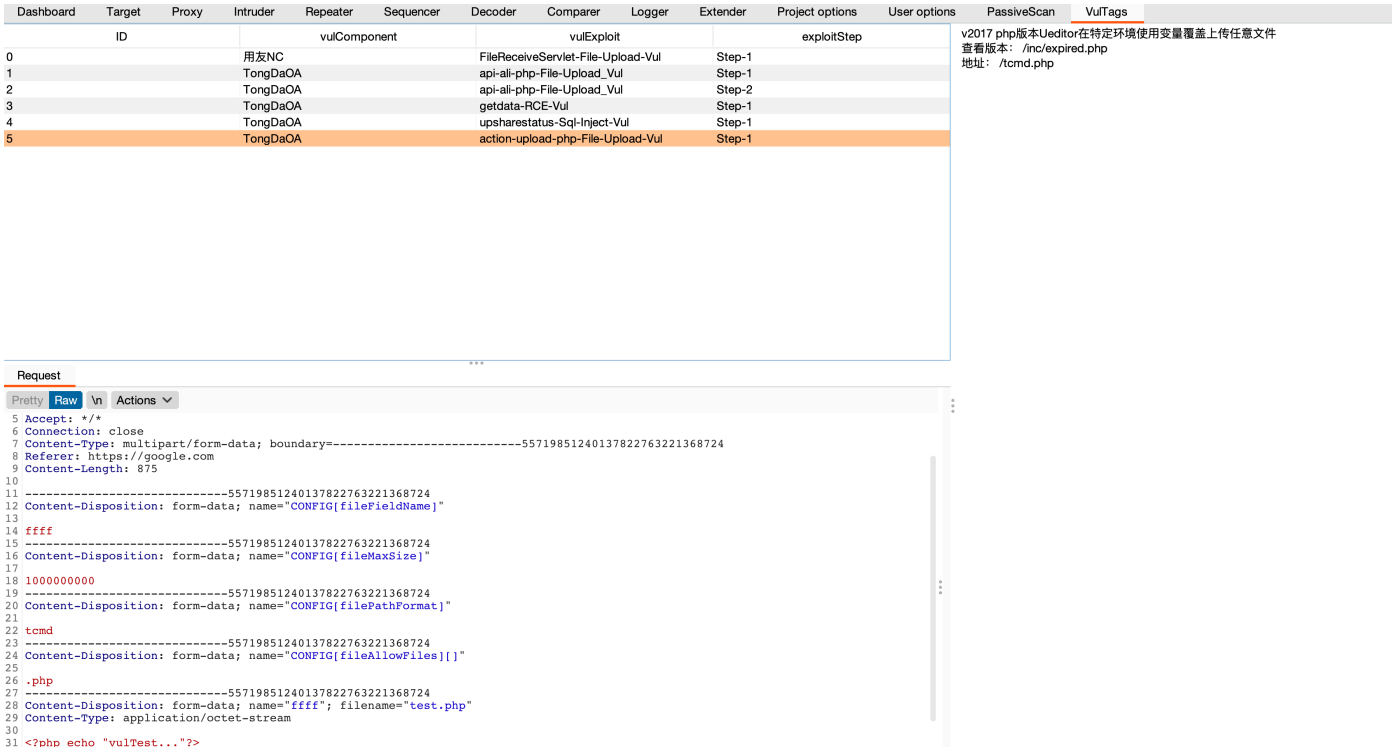
在Proxy界面选择请求（后台漏洞选择有认证头的请求）右键选择 Generate Payload



在弹出界面选择漏洞组件，点击 Send to VulTag



在VulTag界面选择漏洞 Send to Repeater



漏洞模版

Default.json

```
{
    "step": 3,
    //漏洞描述
    "description": "\u6d4b\u8bd5\u63cf\u8ff0",
    //漏洞利用，LIST -> dict
    //漏洞利用需要几个请求就写几个dict
    "exploit": [
        {
            // 请求方法
            "method": "GET",
            // 漏洞路径
            "path": "/testpath/test.php",
            // 请求头 -> dict
            "header": {
                "User-Agent": "i'm come in"
            },
            // 请求体
            "body": ""
        },
        {
            "method": "POST",
            "path": "/testpostpath/test.php",
            "header": {
                "User-Agent": "i'm come in",
```

```
                "Content-Type": "text/xml"
            },
            "body": "dGVzdFBPU1RwYXJhbXM="
        },
        {
            "method": "PUT",
            "path": "/",
            "header": {
                "User-Agent": "i'm come in"
            },
            "body": "PD9waHAgcGhwaW5mbygpOz8+"
        }
    ]
}
```