[Crypto]ECB模式攻击

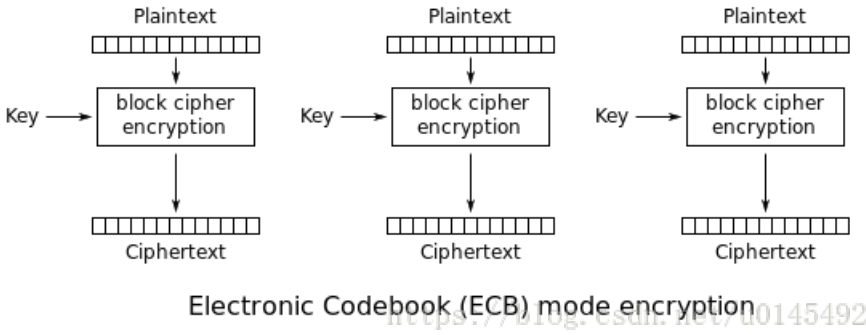
原创

pcy190 于 2018-08-07 18:53:16 发布
分类专栏： CTF CTF 文章标签： ecb Crypto


CTF 同时被 2 个专栏收录
36 篇文章 0 订阅
订阅专栏
CTF
21 篇文章 3 订阅
订阅专栏

【ECB模式攻击原理】

（参考https://zachgrace.com/posts/attacking-ecb/）



Electronic Codebook (ECB) mode encryption

ecb模式使用相同的key分块对明文分别进行加密，相同的明文获得相同的密文输出
根据这一特性，可以构造如下数据进行攻击：
首先输入blocksize-1的填充，这样未知字符串的第一个字符将落到填充块的最后一个字节：

XXXXXXAA AAAAAAAA AAAAAAXX

XXXXXXBA AAAAAAAA AAAAAAAX

XXXXXXBB AAAAAAAA AAAAAAAA

这时会得到填充块的一个密文输出，爆破最后一个字节，直到产生与刚才相同的密文输出，就可以确定未知字符串的一个字节，重复这个过程就可以得到完整的未知字符串。

【攻击方法】

向包含未知字符串的明文中插入数据，其实也是枚举验算的过程：

1.获取未知字符串的第一位

比输入name为111111111111，

服务器生成未知字符串"hello, 111111111111, your mission's flag is: FLAGXXXXXX"的用ecb加密过的字符串S1给我们，

其中FLAGXXXXXX是我们希望得到的flag，通过观察题目程序可知，此处ECB使用的是16位的，

进而观察可知，16个字符的字符串加密后就变成32个字符的密文了。

就是说"hello, 111111111111, your mission's flag is: X"的最后一个X刚好是第32位，我们首先通过这个32字符的字符串获取一下加密过的encrypto， 也就是64个字符的字符串S2，那么这个S2必然是S1的前64位。

然而我们刚开始并不知道这个X是什么，于是我们枚举这个X，令X为一个ascii字符，

比如向服务器发送"hello, 111111111111, your mission's flag is: A"，获取密文，如果这个密文恰好是S1的前64位，那么A就是flag未知字符串的第一位，于是我们就破解了flag的第一位。

2.获取未知字符串的剩余位

因为A已经是flag的第一位，根据破解的原理，我们要把未知字符放在第32位（此处和第一步一样取一个合理的16的倍数）

那么我们只要把name的名字长度缩小一个字符就可以了

即此时变为 **"hello, 11111111111, your mission's flag is: AX"**

（可以对照第一步的字符**"hello, 111111111111, your mission's flag is: A"**）

依然枚举X即可，以此类推，获取全部的字符串（此处以"}"为结尾标志）

【代码】

主程序代码：

```
#!/usr/bin/env python
# coding=utf-8

import conn
import os

number = 30 + 16
mess = "hello, 111111111111, your mission's flag is: moeflag{"
ans = ""
while (number >= 0):
    found = False
    conn.remote("****YourIp****", 8001)
    conn.reads()
    mess = "1" * number
    conn.sends(mess)
    # print("发送 " + mess)
    initialdata = conn.reads()
    # print("原始 "+initialdata)
    blockdata = initialdata[0:128 + 32]
    # print("前端 \n"+blockdata)
    # print(initialdata)
    # print("----------")
    array = "mqmwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM0123456789{}_-~!@#$%^&*()-=+[]?><'|,./`\

    for i in array:
        conn.reads()
        # print("尝试 "+i)
        mess = "hello, " + "1" * number + ", your mission's flag is: " + ans + i
        conn.sends(mess)
        data = conn.reads()
        # print(data)

        # print data
        if data[0:128 + 32] == blockdata:
            found = True
            print("发现 " + i)
            ans += i
            number = number - 1
            if i == "}":
                number = -1
                print("答案:" + ans)
                os.system("pause")
                # message=message+i
            break
    conn.close()
    if not found :
        print("没有找到~~")
        os.system("pause")
print(ans)
```

组件conn的代码

```
import socket

import os

obj = None


def remote(addr, ip):
    global obj
    obj = socket.socket()
    obj.connect((addr, ip))


def reads():
    global obj
    ret_bytes = obj.recv(1024)
    ret_str = str(ret_bytes, encoding="utf-8")
    #print(ret_str)
    return ret_str


def sends(content):
    global obj
    obj.sendall(bytes(content + "\n", encoding="utf-8"))


def close():
    global obj
    obj.close()
```

注：这两个代码需要放在同级目录下，只要运行main就可以了

【编写调试中的问题】

1.连接socket服务器后，能 收到服务器消息， 但是发送客户端发送数据后，却收不到服务器回复。

见https://blog.csdn.net/u014549283/article/details/81484517

这可能是因为输入的字符串最后没有 换行符，导致服务器认为输入未结束。

2.程序运行到一定程度后没有输出反应：

可能有这么几个原因：

原因1：枚举的字符集合不够大，有些flag中例如“=”这样的数据没有包含进去。也可以采用转ascii，这样更完整，但是运算速度就降低了。

原因2：flag位数太大， 移位以后还不够，可以 分批运行程序，也可以扩大初始name的长度

3.本来也可以用python中的pwn来做，但是Windows下面用pip来安装pwn经常出现错误，linux下，更新了国内源后，即便下载成功，安装 也会出错。