第四届美团网络安全高校挑战赛_MTCTF_Crypto_Remeo_复现

原创

M3ng@L 于 2021-12-18 11:03:45 发布 4390 收藏

文章标签： 密码学 python

本文链接：https://blog.csdn.net/qq_51999772/article/details/122009726

Remeo’s Encrypting Machine

题目描述：

```
from Crypto.Util.number import*
from Crypto.Cipher import AES
from secret import msg,password,flag
import socketserver
import signal
assert len(msg) == 32
assert len(password) == 8
# password = "abcd%^&*" 在本地运行的时候随便赋个password即可

def padding(msg):
    return msg + bytes([0 for i in range((16 - len(msg))%16)]) # 这里的bytes()函数运行会报错，我这直接改成str()进行本地测试的
class Task(socketserver.BaseRequestHandler):
    def _recvall(self):
        BUFF_SIZE = 2048
        data = b''
        while True:
            part = self.request.recv(BUFF_SIZE)
            data += part
            if len(part) < BUFF_SIZE:
                break
        return data.strip()

    def send(self, msg, newline=True):
        try:
            if newline:
                msg += b'\n'
            self.request.sendall(msg)
        except:
            pass

    def recv(self):
        return self._recvall()

    def login(self):
        right_num = 0
        while 1:
            self.send(b'[~]Please input your password:')
            str1 = self.recv().strip()[:8]
            true_num = 0
```

```
            for i in range(len(password)):
                if str1[i] != password[i]:
                    login = False
                    self.send(b'False!')
                    break
                else:
                    true_num = i + 1

                if right_num > true_num:
                    continue
                else:
                    right_num = true_num

                if true_num == len(password):
                    login = True
                check = b''
                for i in range(0x2000):
                    check = self.aes.encrypt(padding(check[:-1] + str1[:i+1]))

            if login == True:
                self.send(b"Login Success")
                return True,check[:16]

        return False

    def handle(self):
        signal.alarm(100)
        self.aes = AES.new(padding(password),AES.MODE_ECB)
        _,final_check = self.login()
        if _ == 1:
            assert msg.decode() == final_check.hex()
            self.send(b'Good Morning Master!')
            self.send(flag)

class ThreadedServer(socketserver.ThreadingMixIn, socketserver.TCPServer):
    passu

class ForkedServer(socketserver.ForkingMixIn, socketserver.TCPServer):
    pass

if __name__ == "__main__":
    HOST, PORT = '0.0.0.0', 9999
    print("HOST:POST " + HOST+":" + str(PORT))
    server = ForkedServer((HOST, PORT), Task)
    server.allow_reuse_address = True
    server.serve_forever()
```

分析程序

关键是Task类；其余都是实现连接的代码

```
def handle(self):
 signal.alarm(100)
 self.aes = AES.new(padding(password),AES.MODE_ECB)
 _,final_check = self.login()
  if _ == 1:
            assert msg.decode() == final_check.hex()
            self.send(b'Good Morning Master!')
            self.send(flag)
```

首先开始的是这个函数，里面包含了login函数，跳转去看看login()的定义

看看我们到底传什么东西进去：

```
self.send(b'[~]Please input your password:')
str1 = self.recv().strip()[:8]
```

这里的recv()函数在题目中有定义；实现的功能是从我们输入的字节中接受2048bits的量（之后爆破的过程中比如我们传入"123"，则实际上服务器程序收到的是“123123123”这样由“123”填充到2048字节的字符串），再进行去"\n"且切前8位的片

整个程序就只有这么一个传入点；之后再来分析传入的数值赋给的变量str1

```
for i in range(len(password)):
 if str1[i] != password[i]:
  login = False
        self.send(b'False!')
        break
```

首先逐位判断传入的字符是否与password相同（不安全的比较函数）

如果一旦有一位不同则退出这个比较循环

如果相同的话，则继续之后的；其中有一个AES加密过程，加密结果赋值给check变量

```
for i in range(0x2000):
 check = self.aes.encrypt(padding(check[:-1] + str1[:i+1]))
```

紧接着

```
if login == True:
 self.send(b"Login Success")
 return True,check[:16]
```

如果传入的str1与password完全相同，则整个函数返回True，check的前16位；实际上check的数值，我们管不了，`只要传入的str1与password相同`即可得到flag

那么这个AES的加密过程又有什么用呢？

由于加密相比普通的程序逻辑上要多花一些时间（1s以内）

爆破过程分析

所以可以使用`timing attack`（实际上和sql时间盲注原理差不多）

在一个包含所有可能构成password的字符的字符表中进行逐位爆破（使用string.printable作为这个字符表）

检验相邻的两个爆破时间长短；

我们通过前面的分析可以知道每当一位password与str1相同时，就会进行一次AES加密，这就意味着时间差就会拉大，就更有可能判断对正确的字符

第一层for循环

然后开始爆破：

```
alph = string.printable
password = ""
for n in tqdm(range(8)):  #已知password是8位，所以只需爆破8位即可
    t = 0.0
    temp = ""
    for i in alph[:94]:  # 由于94位之后都是空格或者\t,\n,很显然不是password里面的字符，所以进行切片操作
        if i != "!":   # 这里不知道为什么一旦传入"!"程序就会报错停止运行，所以就干脆先ban掉
            io.recvuntil(b"\n")  # 服务器程序里面提到过每次传给客户机的字符串之后都会加上'\n'，所以以这个为界限进行传输和
交互
            io.send((password + i).encode("utf8"))  #传入的数据必须是字节类型
            start = time()   # 开始时间
            data = io.recvuntil(b"\n")
            end = time()   # 结束时间
            if (end - start) > t:
                temp = i
                t = end - start
    password += temp  # 选出执行程序时间最长的那一位字符
print(password)
```

这样一次爆破就完成了

第二层while循环

但是实际上由于各种不可控因素，每次进行爆破得到的结果都不会相同

所以在爆破位次的基础上我们还需要进行一轮爆破（判断条件是服务器返回的数据中有无b"Success"）

在原来爆破的基础上增加一个while循环

```
alph = string.printable
password = ""
while True:
    for n in tqdm(range(8)):
        t = 0.0
        temp = ""
        for i in alph[:94]:
            if i != "!":
                # print(i,end="")
                io.recvuntil(b"\n")
                io.send((password + i + "0").encode("utf8"))
                start = time()
                data = io.recvuntil(b"\n")
                end = time()
                if (end - start) > t:
                    temp = i
                    t = end - start
                    # print(end - start)
        password += temp
        ending = time()
        if ending - starting > 98:
            middle = 1
            print(password)
            break
        # print(password)
    if middle == 1:
        break
    io.recvuntil(b"\n")
    io.send(password.encode())  # 提交最后一位爆破完成之后生成的password
    data = io.recvuntil(b"\n")
    if b"Success" not in data:
        print(password)
        password = ""
        continue
    else:
        print(password)
        io.interactive()
        break
```

第三层while循环

运行了程序我们就会知道，每当爆破100s之后，服务器程序会自动切断连接

从脚本中可以看到：

```
def handle(self):
 signal.alarm(100)
```

那么怎么让ta实现完全自动化的爆破脚本（不需要每过100s就手动再运行一次程序）

再加一个while循环；在服务器切断连接之前我们重新再次连接服务器程序，这样100s就会重新倒计时

```
alph = string.printable
password = ""
while True:
    io = remote("127.0.0.1","9999")
    starting = time() # 一次连接开始的时间
    middle = 0
    while True:
        for n in tqdm(range(8 - len(password))): # 8 - len(password)的目的是为了在自己切断服务器连接之后上一次爆破的password如果没有完成的话，接着爆破剩下的位数
            t = 0.0
            temp = ""
            for i in alph[:94]:
                if i != "!":
                    # print(i,end="")
                    io.recvuntil(b"\n")
                    io.send((password + i).encode("utf8"))
                    start = time()
                    data = io.recvuntil(b"\n")
                    end = time()
                    if (end - start) > t:
                        temp = i
                        t = end - start
                        # print(end - start)
            password += temp
            ending = time()  # 查看已经连接多长时间了
            if ending - starting > 98:  # 大于98s之后就可以主动切换了，不要太贪心
                middle = 1  # 使用变量传递的形式跳出多个循环
                print(password)
                break
            # print(password)
        if middle == 1:
            break
        io.recvuntil(b"\n")
        io.send(password.encode())
        data = io.recvuntil(b"\n")
        if b"Success" not in data:
            # sum.append(password)
            print(password)
            password = ""
            continue
        else:
            print(password)
            break
    if middle == 1:
        continue
    io.interactive() # 如果是因为password相同而跳出循环就会执行正常的交互，退出机械交互
```

这样就实现了一个完全自动的爆破位次的程序（~~但是我还没爆破出来正确的password是什么…~~）

PS：由于上面的脚本是为了讲清楚脚本部分的逻辑，所以单独运行是不行的，有几个模块还没引入

下面是完整的脚本

代码实现

```python
from pwn import *
from time  import time
import string
from tqdm import tqdm

# context.log_level='debug'

alph = string.printable
password = ""

while True:
    io = remote("127.0.0.1","9999")
    starting = time()
    middle = 0
    while True:
        for n in tqdm(range(8 - len(password))):
            t = 0.0
            temp = ""
            for i in alph[:94]:
                if i != "!":
                    # print(i,end="")
                    io.recvuntil(b"\n")
                    io.send((password + i).encode("utf8"))
                    start = time()
                    data = io.recvuntil(b"\n")
                    end = time()
                    if (end - start) > t:
                        temp = i
                        t = end - start
                        # print(end - start)
            password += temp
            ending = time()
            if ending - starting > 98:
                middle = 1
                print(password)
                break
            # print(password)
        if middle == 1:
            break
        io.recvuntil(b"\n")
        io.send(password.encode())
        data = io.recvuntil(b"\n")
        if b"Success" not in data:
            # sum.append(password)
            print(password)
            password = ""
            continue
        else:
            print(password)
            break
    if middle == 1:
        continue
    io.interactive()
```

参考文章

(8条消息) 20211211 美团CTF2021 Crypto方向&&Pwn方向部分WP_4XWi11的博客-CSDN博客

第二届美团ctf预赛-writeup by WDNMD (qq.com)