CBC Bit-Flipping Attack Conclusion

#cbc

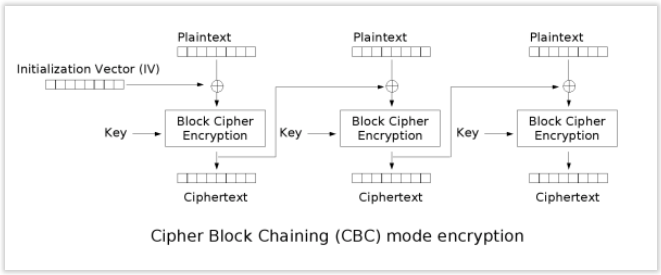
2019/05/23 122 Share

Crypto

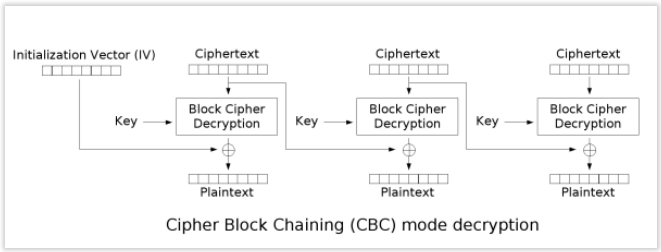
CATALOG

前面通过一个例子说明了分组密码block cipher里的ECB分组模式存在的已知明文攻击，现在看一下CBC分组模式里的CBC Bit-Flipping Attack。

CBC全称Cipher Block Chaining，密码分组链接模式，下面是它的加密和解密过程。



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

从加密过程中可以看到，每个密文块都依赖于它前面所有的明文块，这就很明显地与ECB模式的每个块独立加解密不同。

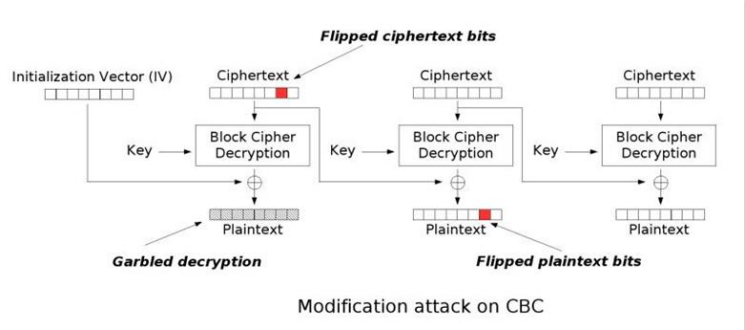
CBC Bit-Flipping Attack在国内又被称为CBC字节翻转攻击，无论是翻转bit还是byte，本质上还是一致的，所以不必纠结中英文的不同。首先要知道该攻击发生在CBC的解密环节上。

要实现该攻击需要一定的条件：

> We will list down all the information one must have access to, in order to initiate this attack:
>
> 1. Cipher text
> 2. Encryption Oracle as E("random string" || payload || "another random string")
>   - Here, in this function, the attacker is allowed to supply input to the encryption function as payload. This function is literally the heart of the attack. All the arguments for the attack will be supplied here.
> 3. Decryption Oracle

从上图可以看到，首先要能控制CBC分组加密的输入，而且要能拿到加密后的密文，并要对密文进行解密。

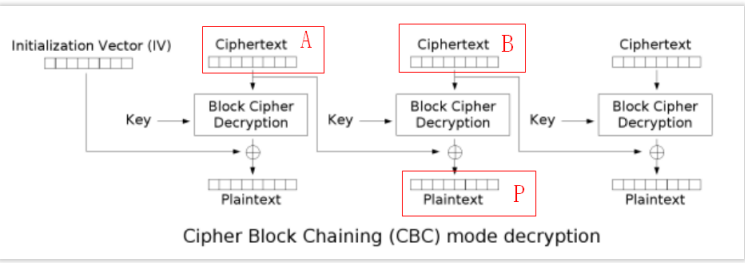
接着看为什么说攻击发生在CBC的解密环节，如下图所示：



Modification attack on CBC

上图可以直观地看到，在解密过程里，通过翻转前一组密文里特定位置的bit，从而达到了翻转下一组明文里特定位置bit的效果。同样的，如果可以修改iv，那么也可以修改第一组解密出的明文内容(https://masterpessimistaa.wordpress.com/2018/07/23/exploiting-cookie-auth-mechanism-ctfzone-ussh-3-0-write-up/)。

进一步分析其原理（参考https://masterpessimistaa.wordpress.com/2017/05/03/cbc-bit-flipping-attack/）：



Cipher Block Chaining (CBC) mode decryption

从上图可以清楚得到：

```
A = P xor BlockCipherDecryption(B)
```

需要注意的是 BlockCipherDecryption(B)是一个常量，因为这里没有修改B

对于分组的第n字节，相应地有：

```
A[n] = P[n] xor BlockCipherDecryption(B[n])
```

变形得到：

```
BlockCipherDecryption(B[n]) = A[n] xor P[n]
```

在式1里，假定我们想要输出的明文P[n]为我们想要的明文，设为P1

在式2里，假定输出的明文P[n]是密文未经过修改得到的真实明文，设为P2

于是有：

```
A[n] = P1 xor A[n] xor P2
```

调整顺序：

```
A[n] = A[n] xor P1 xor P2
```

可见，通过这种方式就可以修改密文达到翻转解密出的明文字节的效果。

示例代码如下：

```
from Crypto.Cipher import AES
from os import urandom

key = urandom(16)
iv = urandom(16)


def padding(string):
    l = len(string)
    blocksize = 16
    padlen = blocksize - (l % blocksize)
    padbyte = hex(padlen).replace("0x", "")
    if len(padbyte) != 2:
        padbyte = "0" + padbyte
    string = string.encode("hex") + padlen*padbyte
    string = string.decode("hex")
```

```
    return string


def encrypt(payload):
    obj = AES.new(key, AES.MODE_CBC, iv)
    for i in xrange(len(payload)):
        if payload[i] == ";" or payload[i] == "=":
            payload = payload.replace(payload[i], "?")
    str1 = "comment1=cooking%20MCs;userdata=" + payload + ";comment2=%20li
    str1 = padding(str1)
    ciphertext = obj.encrypt(str1)
    return ciphertext


def decrypt(ciphertext):
    obj1 = AES.new(key,AES.MODE_CBC,iv)
    plaintext = obj1.decrypt(ciphertext)
    if ";admin=true;" in plaintext:
        print "Logged in as admin"
    else:
        print "You need to be admin to get the access!"

if __name__ == "__main__":
    p = ";admin=true;"
    c = encrypt(p)
    print c
    decrypt(c)
```

重点看加解密函数：

encrypt(payload)函数使用AES-CBC加密，并接收payload作为加密的输入，可见输入可控，并且返回了密文，另外需要注意的是，代码里将payload中出现的";"和"="都替换为"?"再进行padding和加密。

decrypt(ciphertext)函数对密文解密，如果解密出明文里包含了";admin=true;"就认证成功，因为上面加密函数的处理，使得这里的条件似乎永远无法满足。

运行代码，效果如下，符合预期。

```
WnHf[illegible]+[illegible]M
%[illegible]Qx[illegible]a[illegible]/[illegible]\o[illegible]iy-_K[illegible]_M[illegible]W@[illegible]<&[illegible]M [illegible]$o \,b[illegible]n[illegible]=[illegible]=[illegible]a[illegible]A;[illegible]Uj7X(J[illegible]
You need to be admin to get the access!
```

为了绕过代码限制认证成功，这里可以使用CBC Bit-Flipping Attack。

```
from Crypto.Cipher import AES
from os import urandom

key = urandom(16)
iv = urandom(16)


def padding(string):
    l = len(string)
    blocksize = 16
    padlen = blocksize - (l % blocksize)
    padbyte = hex(padlen).replace("0x", "")
    if len(padbyte) != 2:
        padbyte = "0" + padbyte
    string = string.encode("hex") + padlen*padbyte
    string = string.decode("hex")
    return string


def encrypt(payload):
    obj = AES.new(key, AES.MODE_CBC, iv)
    for i in xrange(len(payload)):
        if payload[i] == ";" or payload[i] == "=":
            payload = payload.replace(payload[i], "?")
    str1 = "comment1=cooking%20MCs;userdata=" + payload + ";comment2=%20li
    str1 = padding(str1)
    ciphertext = obj.encrypt(str1)
    return ciphertext


def decrypt(ciphertext):
    obj1 = AES.new(key,AES.MODE_CBC,iv)
    plaintext = obj1.decrypt(ciphertext)
    if ";admin=true;" in plaintext:
        print "Logged in as admin"
    else:
        print "You need to be admin to get the access!"


# Exploit using the Bit Flipping Attack!
```

```
cipher_list = []
payload = ";admin=true;"
ciphertext = encrypt(payload)

i = 0
while i*16 <= len(ciphertext):
    cipher_list.append(ciphertext[i*16: 16 + (i*16)])
    i += 1
cipher_list.remove(cipher_list[6])

attack_on_block = cipher_list[1]
list1 = list(attack_on_block)
list1[0] = chr(ord(list1[0]) ^ ord("?") ^ ord(";"))
list1[6] = chr(ord(list1[6]) ^ ord("?") ^ ord("="))
list1[11] = chr(ord(list1[11]) ^ ord("?") ^ ord(";"))
cipher_list[1] = ''.join(list1)
ciphertext = ''.join(cipher_list)

decrypt(ciphertext)
```

运行后发现成功翻转了密文为预期的;admin=true;

```
root@automne:/Pentest/Crypto/CBC# python cbc3.py
Logged in as admin
```

2019.7.14更新

重写上面的代码，使之易于理解

```
from Crypto.Cipher import AES
from os import urandom

key = urandom(16)
iv = urandom(16)


def padding(string):
    l = len(string)
    blocksize = 16
    padlen = blocksize - (l % blocksize)
    padbyte = hex(padlen).replace("0x", "")
    if len(padbyte) != 2:
        padbyte = "0" + padbyte
    string = string.encode("hex") + padlen*padbyte
    string = string.decode("hex")
    return string


def encrypt(payload):
    obj = AES.new(key, AES.MODE_CBC, iv)
    for i in xrange(len(payload)):
        if payload[i] == ";" or payload[i] == "=":
            payload = payload.replace(payload[i], "?")
    str1 = "comment1=cooking%20MCs;userdata=" + payload + ";comment2=%20li
    str1 = padding(str1)
    ciphertext = obj.encrypt(str1)
    return ciphertext


def decrypt(ciphertext):
    obj1 = AES.new(key,AES.MODE_CBC,iv)
    plaintext = obj1.decrypt(ciphertext)
    print plaintext
    if ";admin=true;" in plaintext:
        print "Logged in as admin"
    else:
        print "You need to be admin to get the access!"


# Exploit using the Bit Flipping Attack!
cipher_list = []
payload = ";admin=true;"
ciphertext = encrypt(payload)
print ciphertext

theBlock = ciphertext[16:32]

newBlock0 = chr(ord(theBlock[0])^ord("?")^ord(";"))
newBlock6 = chr(ord(theBlock[6])^ord("?")^ord("="))
newBlock11 = chr(ord(theBlock[11])^ord("?")^ord(";"))

ciphertext_new = ciphertext[:16]+newBlock0+ciphertext[17:22]+newBlock6+cip

decrypt(ciphertext_new)
```

运行效果：

```
??a9*?? ?y?V??X?S?a?4?~?vm?;?[?]?wt?$??b?X??_?F??)?u?0?????a$???Y?)q0??Ukv?
                                                                        ???K
                                                                          ?M*J??P
comment1=cooking?W????P?jAo¥m};admin=true;;comment2=%20like%20a%20pound%20of%20bacon
```

```
Logged in as admin
```


< Next Post
HSCTF 2019
SuperSecureSystem
WriteUp

Previous Post >
LFSR in Crypto
Conclusion